Samuel Collin Robinson, P.E.

```
Option Explicit
'*************************************************************************************************************************************************
'*************************************************************************************************************************************************
'*************************************************************************************************************************************************
'*************************************************************************************************************************************************
Sub Solve()                                                         'Calls data preparation, specified solver, and chart subroutines.
Dim msg As String, msgh As String                                   'Dimension Warning Message, Message Holder.
Dim ic As Integer, imn As Integer, rc As Integer, rmn As Integer    'Dimension impulse and response counts and minima.
Dim IMS As String                                                   'Dimension impulse distribution selection.
Dim idk As Integer                                                  'Dimension impulse distribution key.
Dim now1 As Double, now2 As Double                                  'Dimension start and stop time to allow diagnostic output of elapsed.
Dim IT As String                                                    'Dimension impulse type.
Dim TSIS As Double                                                  'Dimension time series impulse total.
Dim CPC As Variant                                                  'Dimension nothing to see here.
CPC = "*******"                                                     'Dimension still nothing.
Dim DIC As String                                                   'Dimension workbook status indicator.
Dim sel As Range                                                    'Dimension initial selection.
Dim AT As String                                                    'Dimension Aquifer Type.
Dim i As Long                                                       'Dimension counter.
Dim SB As String, SIT As String                                     'Dimension streambed type parameters.

now1 = Now                                                         'Set start time.
Sheet11.Unprotect Password:=CPC
Set sel = Selection                                                 'Loads initial selection.
DIC = WorksheetFunction.Proper(Sheet11.Cells(2, 2).Value)
Sheet11.Range("E39:g2438").ClearContents
Sheet11.Range("j80:j158").ClearContents
Sheet11.Cells(30, 5).Value = "None!"
Sheet11.Cells(31, 5).Value = "under way"
Sheet11.Cells(6, 22).Value = 0
Sheet11.Cells(10, 20).Value = 0
Sheet11.Cells(20, 20).Value = 0
Sheet11.Protect Password:=CPC

Application.ScreenUpdating = False

Sheet10.Unprotect Password:=CPC                                    'Prepare to clear heads.
Sheet10.Range("c77:az126").ClearContents
Sheet10.Range("c140:d2540").ClearContents
Sheet10.Range("c75:d75").ClearContents
Sheet10.Range("g73:g74").ClearContents
Sheet10.Range("bh73:bh74").ClearContents
Sheet10.Range("g134:g135").ClearContents
Sheet10.Range("c135:d135").ClearContents
Sheet10.Range("bd77:da126").ClearContents
Sheet10.Range("bc73:bc74").ClearContents
Sheet10.Protect Password:=CPC                                      'Restore sheet.

Sheets("Time").Select

If DIC = "Delayed Impact Calculator" Then                          'DIC if - Check that workbook sound.
```

```
  ITSP                                                                  'Executes the subroutine below that prepares the impulse time series input
data.

msg = Sheet11.Cells(30, 5).Value
If msg = "None!" Then                                                    'Msg if.
AT = Sheet3.Cells(9, 8).Value
ic = Sheet3.Cells(7, 15).Value
rc = Sheet3.Cells(10, 15).Value
imn = Sheet3.Cells(7, 16).Value
rmn = Sheet3.Cells(10, 16).Value
IMS = WorksheetFunction.Proper(Sheet8.Cells(7, 6).Value)
SB = WorksheetFunction.Proper(Sheet12.Cells(7, 6).Value)
SIT = WorksheetFunction.Proper(Sheet12.Cells(8, 6).Value)

If Left(IMS, 4) = "Disc" Then
 If Sheet8.Cells(8, 11).Value > 0 Then
  idk = 1
  Else
  idk = 0
 End If
 Else
 idk = 1
End If
If ic >= imn Then                                                     'Impulse cell count if.
 If rc >= rmn Then                                                    'Response cell count if.
 If idk = 1 Then                                                      'Impulse distribution.

    If AT <> "Unconfined" Then
    Dim imm As Integer, jmm As Integer, DECz As Double, AqTz As Double, CTz As Integer  'Checks that aquifer top below ground.
    Dim Ll As Integer, mm As Integer
    Ll = Sheet3.Cells(12, 8).Value
    mm = Sheet3.Cells(15, 8).Value
    For jmm = 1 To mm
    For imm = 1 To Ll
     If WorksheetFunction.Proper(Sheet4.Cells(8, 5).Value) = "Uniform" Then
       AqTz = -1 * Sheet4.Cells(9, 5).Value
       Else
       AqTz = -1 * Sheet4.Cells(23 + jmm, 2 + imm).Value
     End If
     CTz = Sheet3.Cells(26 + jmm, 3 + imm).Value
     If WorksheetFunction.Proper(Sheet5.Cells(8, 5).Value) = "Uniform" Then
       DECz = -1 * Sheet5.Cells(9, 5).Value
       Else
        If CTz = 2 Then
         If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
           DECz = -1 * Sheet5.Cells(16 + jmm, 2 + imm).Value
          Else
           DECz = 0
         End If
         Else
           DECz = -1 * Sheet5.Cells(16 + jmm, 2 + imm).Value
        End If
     End If
     If AqTz > DECz Then
      Sheet11.Unprotect Password:=CPC
       Sheet11.Cells(30, 5).Value = "Posted at J80."
```

```
    Sheet11.Cells(31, 5).Value = "Ended."
    Sheet11.Cells(80, 10) = "Calculation Notes"
    Sheet11.Cells(81, 10).Value = "Aquifer Top above Ground!"
   Sheet11.Protect Password:=CPC
   Application.ScreenUpdating = True
   Exit Sub
  End If
 Next imm
 Next jmm
 If AT = "Delayed" Then
  If WorksheetFunction.Proper(Sheet3.Cells(11, 8).Value) = "Next Step" Then
   Sheet3.Unprotect Password:=CPC
   Sheet3.Cells(11, 8).Value = "Same Step"
   Sheet3.Protect Password:=CPC
  End If
 End If
End If

 If AT <> "Delayed" Then                                    'Executes a subroutine below that writes and solves a system of equations.
  If SB = "Restrictive" Then
   If SIT = "Complete" Then
     IMTC                                                   'With 3d thickness and storage for bank overlap.
    Else
     IMT                                                    'With no bank parameters or overlap.
   End If
   Else
    IMT
  End If
 ElseIf AT = "Delayed" Then
  If SB = "Restrictive" Then
   If SIT = "Complete" Then
     IMTDC                                                  'With 3d thickness and storage for bank overlap, delayed-yield aquifer.
    Else
     IMTD                                                   'With no bank parameters or overlap, delayed-yield aquifer.
   End If
   Else
    IMTD
  End If
 End If

   If Sheet10.Cells(135, 7).Value <> Empty Then
     HTSC                                                   'Executes subroutine below that rescales head output time series ranges.
   End If
     TSC                                                    'Executes subroutine below that sets time series chart range.

Sheet11.Unprotect Password:=CPC
If Sheet11.Cells(6, 22).Value = 1 Then
Sheet11.Cells(31, 5).Value = "Complete!"
ElseIf Sheet11.Cells(6, 22).Value = 2 Then
Sheet11.Cells(31, 5).Value = "Failed."
ElseIf Sheet11.Cells(6, 22).Value = 3 Then
Sheet11.Cells(31, 5).Value = "Ended."
ElseIf Sheet11.Cells(6, 22).Value = 4 Then
Sheet11.Cells(31, 5).Value = "Flawed."
ElseIf Sheet11.Cells(6, 22).Value = 5 Then
Sheet11.Cells(31, 5).Value = "Anomalous."
```

```
  End If
  Sheet11.Protect Password:=CPC

Else                                                    'Impulse distribution else.
 If msg = "None!" Then
 msg = ""
End If
Sheet11.Unprotect Password:=CPC
If Left(msg, 4) = "Post" Then
i = 1
Do While i < 100
If Sheet11.Cells(80 + i, 10) = Empty Then
 Sheet11.Cells(80 + i, 10) = "Zero Impulse Distribution! "
 Sheet11.Cells(80 + 1 + i, 10) = msg
 i = 100
 Else:
 i = i + 1
End If
Loop
ElseIf msg = "" Then
Sheet11.Cells(30, 5).Value = "Zero Impulse Distribution! "
Else
 Sheet11.Cells(30, 5).Value = "Posted at J80."
 Sheet11.Cells(80, 10) = "Calculation Notes"
 i = 1
 Sheet11.Cells(80 + i, 10) = msg
 i = i + 1
 Sheet11.Cells(80 + i, 10) = "Zero Impulse Distribution! "
End If
Sheet11.Cells(31, 5).Value = "Failed."
Sheet11.Protect Password:=CPC
End If                                                  'Impulse distribution end if.
Else                                                    'Responce cell else.
If msg = "None!" Then
 msg = ""
End If
 Sheet11.Unprotect Password:=CPC
If Left(msg, 4) = "Post" Then
i = 1
Do While i < 100
If Sheet11.Cells(80 + i, 10) = Empty Then
 Sheet11.Cells(80 + i, 10) = "A response cell? "
 Sheet11.Cells(80 + 1 + i, 10) = msg
 i = 100
 Else:
 i = i + 1
End If
Loop
ElseIf msg = "" Then
Sheet11.Cells(30, 5).Value = "A response cell? "
Else
 Sheet11.Cells(30, 5).Value = "Posted at J80."
 Sheet11.Cells(80, 10) = "Calculation Notes"
 i = 1
 Sheet11.Cells(80 + i, 10) = msg
 i = i + 1
```

```
  Sheet11.Cells(80 + i, 10) = "A response cell? "
 End If
 Sheet11.Cells(31, 5).Value = "Failed."
 Sheet11.Protect Password:=CPC
 End If                                                    'Response cell end if.
 Else                                                      'Impulse cell else.
  If msg = "None!" Then
  msg = ""
 End If
  Sheet11.Unprotect Password:=CPC
 If Left(msg, 4) = "Post" Then
 i = 1
 Do While i < 100
 If Sheet11.Cells(80 + i, 10) = Empty Then
  Sheet11.Cells(80 + i, 10) = "An impulse cell? "
  Sheet11.Cells(80 + 1 + i, 10) = msg
  i = 100
  Else:
  i = i + 1
 End If
 Loop
 ElseIf msg = "" Then
 Sheet11.Cells(30, 5).Value = "An impulse cell? "
 Else
  Sheet11.Cells(30, 5).Value = "Posted at J80."
  Sheet11.Cells(80, 10) = "Calculation Notes"
  i = 1
  Sheet11.Cells(80 + i, 10) = msg
  i = i + 1
  Sheet11.Cells(80 + i, 10) = "An impulse cell? "
 End If
 Sheet11.Cells(31, 5).Value = "Failed."
 Sheet11.Protect Password:=CPC
End If                                                     'Impulse cell end if.
Else                                                       'Msg Else.
 Sheet11.Unprotect Password:=CPC
 Sheet11.Cells(31, 5).Value = "Failed."
 Sheet11.Protect Password:=CPC
End If                                                     'Msg end if.

TSIS = Sheet11.Cells(6, 20).Value
If TSIS = 0 Then                                            'TSIS if - Check for zero input.
 IT = WorksheetFunction.Proper(Sheet11.Cells(36, 4).Value)
 msgh = Sheet11.Cells(30, 5).Value
 If msg = "None!" Then
  msg = ""
 End If
 If msg <> msgh Then
  If msgh <> "" Then
   If msg <> "" Then
    msg = msgh & ", " & msg
   Else
    msg = msgh
   End If
  End If
 End If
```

```
 Sheet11.Unprotect Password:=CPC
 If Left(msg, 4) = "Post" Then
 i = 1
 Do While i < 100
 If Sheet11.Cells(80 + i, 10) = Empty Then
  Sheet11.Cells(80 + i, 10) = "Zero Net " & IT & ". "
  i = 100
  Else:
  i = i + 1
 End If
 Loop
 ElseIf msg = "None!" Then
 Sheet11.Cells(30, 5).Value = "Zero Net " & IT & ". "
 Else
  Sheet11.Cells(30, 5).Value = "Posted At J80."
  Sheet11.Cells(80, 10) = "Calculation Notes"
  i = 1
  Sheet11.Cells(80 + i, 10) = msg
  i = i + 1
  Sheet11.Cells(80 + i, 10) = "Zero Net " & IT & ". "
 End If
 Sheet11.Range("g39:g2438").ClearContents
 Sheet11.Range("e29").ClearContents
 Sheet11.Protect Password:=CPC
End If                                                  'TSIS end if.

Else                                                    'DIC else.
 Sheet11.Unprotect Password:=CPC
 Sheet11.Range("E39:g2438").ClearContents
 Sheet11.Cells(30, 5).Value = "Update workbook."
 Sheet11.Cells(31, 5).Value = "Failed."
 Sheet11.Protect Password:=CPC
End If                                                  'DIC end if.

 now2 = Now
 Sheet11.Unprotect Password:=CPC
 Sheets("Time").Select
 Sheet11.Cells(28, 5).Value = now2 - now1                     'DIAGNOSTIC - Output time elapsed during sub-routine.
 sel.Select
 Sheet11.Protect Password:=CPC

Application.ScreenUpdating = True

End Sub
'*************************************************************************************************************************************************
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
'*************************************************************************************************************************************************
Sub ITSP()                                                  'Subroutine that prepares the impulse time series input data.
Dim St As String, PU As String, msg As String               'Dimension Schedule Type, Period Units, warning message.
Dim no As Integer, o As Integer, p As Integer               'Dimension number of periods, loop counters.
Dim IT(1 To 365) As Double                                  'Dimension first year impulse by period.
Dim CPC As Variant
Dim Qm() As Double
```

```
CPC = “*******”

St = WorksheetFunction.Proper(Sheet11.Cells(12, 5).Value)                          'Load parameters.
PU = WorksheetFunction.Proper(Sheet11.Cells(14, 5).Value)
no = Sheet11.Cells(13, 5).Value
Sheet11.Unprotect Password:=CPC
msg = "None!"

ReDim Qm(1 To no)

' -                                                                 'Variable series inputs left as-are.

If St = "Single Impulse" Then                                      'Single Impulse case.
 Sheet11.Range("d40:d2438").Value = 0
End If

If St = "Uniform Series" Then                                      'Uniform Series case.
 IT(1) = Sheet11.Cells(39, 4).Value
 Sheet11.Range("d40:d2438").Value = IT(1)
End If

If St = "Annual Pattern" Then                                      'Annual Pattern case.
 If PU = "Months" Then                                             'Check for adequate simulation span for rolling impact factors.
  If no < 36 Then
   msg = "Add Periods!"
  Else
  o = 1
  Do While o < 13
   IT(o) = Sheet11.Cells(38 + o, 4).Value                          'Load first year impulse totals by period.
   o = o + 1
  Loop
  p = 12 + 1
  Do While p < no + 1
   o = 1
   Do While o < 13
    Sheet11.Cells(38 + p, 4).Value = IT(o)                         'Cycle first year impulse totals to fill remainder of time series.
    o = o + 1
    p = p + 1
   Loop
  Loop
  End If
  ElseIf PU = "Days" Then
   If no < 1095 Then
    msg = "Add Periods!"
   Else
   o = 1
   Do While o < 366
    IT(o) = Sheet11.Cells(38 + o, 4).Value
    o = o + 1
   Loop
   p = 366
   Do While p < no + 1
    o = 1
    Do While o < 366
     Sheet11.Cells(38 + p, 4).Value = IT(o)                        'Cycle first year impulse totals to fill remainder of time series.
     o = o + 1
```

```
     p = p + 1
    Loop
   Loop
  End If
 ElseIf PU = "Hours" Then
  msg = "Period Units!"
 End If
End If

o = 1
Do While o < no + 1
    Qm(o) = Sheet11.Cells(38 + o, 4).Value
    If Qm(o) = Empty Then
     Qm(o) = 0                              'Display 0 for impulse values left empty.
    End If
o = o + 1
Loop
p = 38 + no
Sheet11.Range("d39:" & "d" & p).Value = WorksheetFunction.Transpose(Qm())

Sheet11.Cells(30, 5).Value = msg
Sheet11.Protect Password:=CPC

End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub IMT()                          'Solution of implicit representation.

Dim dt As Double                      'Dimension calculation time step.
Dim dx() As Double, dy() As Double    'Dimension increments.
Dim k() As Double                     'Dimension hydraulic conductivity.
Dim Sy() As Double, Ss() As Double    'Dimension storage properties.
Dim hoo() As Double, ho() As Double   'Dimension initial saturated thickness, transmitting thickness.
Dim AqT() As Double, AqB() As Double  'Dimension aquifer top, bottom.
Dim Hn() As Double, Hno() As Double   'Dimension initial head, original for step, single index vector.
Dim Hni() As Double, Hnp() As Double  'Dimension single index initial head.
Dim Qd() As Double, Qm() As Double    'Dimension net flow impulse distribution, impulse time series.
Dim Qdmx As Double, Qdmn As Double    'Dimension impulse distribution max and min.
Dim Qdmna  As Double                  'Dimension distribution absolute value min.
Dim Qc() As Double, QCP() As Double   'Dimension impulse for calculation.
Dim AA() As Double, BB() As Double    'Dimension flow coefficients: A, B.
Dim CC() As Double, DD() As Double    'Dimension flow coefficients.
Dim EE() As Double, b() As Double     'Dimension volume coefficients.
Dim SE() As Double, SEC() As Double   'Dimension matrix to hold system of equations, redimensioned below with element limits.
Dim Hnn() As Double, Hnnp() As Double 'Dimension next time step differential head, head to plot.
Dim Qnno() As Double                  'Dimension next time step flow output, to be determined for response cells, converting calculated head by
Storage Coefficient, etc.
Dim RSP() As Double, RSPA() As Double 'Dimension Response, Zone Response.
Dim RSPO() As Double, RSPT As Double  'Dimension Response Output, Response Total.
Dim RSPOA() As Double, RSPTA As Double 'Dimension Response Output, Response Total, both for Zone.
Dim i As Integer                      'Dimension matrix row index.
Dim im As Integer, jm As Integer      'Dimension model row, column indices.
Dim L As Integer, m As Integer, r As Integer 'Model: cells on one side, cells on the other side; system rows.
```

```
Dim ITS As String, IMS As String      'Dimension Impulse Type Specification, Impulse Magnitude Specification.
Dim CT() As Integer, CTi() As Integer 'Dimension Cell Type.
Dim IA As Double, IAP As Double       'Dimension Input Area effects.
Dim qs As Double, qrs As Double       'Dimension Flow Sign, Flow Response Sign.
Dim PD As String, ic As String        'Dimension Period Denomination, Impulse Character.
Dim u As Long, n As Long, nc As Long  'Dimension units per period, number of periods, number of sub-periods.
Dim nn As Long, nh As Long            'Dimension number of proxy periods, number of period to plot heads.
Dim no As Long, nm As Long            'Dimension original number of periods, number of of period in undivided increments.
Dim o As Long, deno As Long           'Dimension period counter, characteristic impulse denominator.
Dim ATC As String                     'Dimension Aquifer Thickness Character.
Dim UCF As Double                     'Dimension Unit Conversion Factor.
Dim zone() As String                  'Dimension response Zone.
Dim msg As String, msgc As String     'Dimension warning messages.
Dim msgt As String, msgti As String   'Dimension warning messages.
Dim msgbg As String                   'Dimension warning messages.
Dim msgtt As String, msgh As String   'Dimension warning messages.
Dim msgtp As String, msgmt As String  'Dimension warning messages.
Dim msgim As String, msgim2 As String 'Dimension warning messages.
Dim msgpb As String, msgep As String  'Dimension warning messages.
Dim msgtg As String                   'Dimension warning messages.
Dim MSTA As Double                    'Dimension minimum saturated thickness allowed.
Dim DEC() As Double                   'Dimension depth of earth cover.
Dim IMPST As Double, FCR As Double    'Dimension impulse total, final cumulative ratio.
Dim SO As String                      'Dimension Secondary Output type.
Dim SOV() As Variant                  'Dimension Secondary Output variable.
Dim CUMI As Double, CUMR As Double    'Dimension cumulative impulse and response, respectively, through any time step.
Dim CUMRn() As Double                 'Dimension cumulative response at each time step.
Dim tt As Long                        'Dimension secondary output period index for impact factor calculation.
Dim CPC As Variant                    'Dimension some dummy text here.
CPC = "*******"                       'Dimension nothing to see here again.
Dim RSPOFM As Integer                 'Dimension response output flag to catch initial head imbalance.
Dim RSPOFM2 As Integer                'Dimension stream head imbalance flag.
Dim count2 As Double, count13 As Double 'Dimension storage and area weighted counters for cell types.
Dim headsum2 As Double                'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headave2 As Double                'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headsum2s As Double               'Dimension stream head sum.
Dim headsum13 As Double               'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headave13 As Double               'Dimension initial storage and area weighted head sums and averages by cell type.
Dim SB As String, SIT As String       'Dimesion relative permeability class of streambed, streambed incision type.
Dim WR() As Double                    'Dimension streambed width ratio.
Dim kpp() As Double, bpp() As Double  'Dimension discrete streambed conductivity, thickness.
Dim bppi() As Double, widp() As Double 'Dimension bed thickness single index, dummy width.
Dim widpp() As Double, lenp() As Double 'Dimension streambed width, length.
Dim hrb As Variant, hrr As Variant    'Dimension output cell range extents.
Dim epse As Integer                   'Dimension iteration closure threshold exponent.
Dim ThickCon As String                'Dimension thickness contribution option...update timing.
Dim IMPSMX As Double, IMPSMN As Double 'Dimension impulse max and min.
Dim NegHead As Integer                'Dimension negative head flag.
Dim NegHeadNM As Integer              'Dimension neghead error period.
Dim AqBi() As Double, AqTi() As Double 'Dimension single-index variables for bottom and top.
Dim AvTp As String                    'Dimension transmission characteristic average type.
Dim Bed() As Double, BEDi() As Double 'Dimension discrete bed elevation.
Dim SHD() As Double                   'Dimension stream head.
Dim TipW() As Integer                 'Dimension thickness input parameter watch variable.
Dim hx() As Double, eps As Double     'Dimension successive approximation head vector, convergence threshold for MICCG.
Dim Ax() As Double, rr() As Double    'Dimension MICCG solver variables.
```

```
Dim vv() As Double, zz() As Double    'Dimension MICCG variables.
Dim UT() As Double, DU() As Double    'Dimension MICCG solver preconditioner matrices.
Dim UM() As Double                    'Dimension MICCG upper preconditioner matrix.
Dim pp() As Double, Ap() As Double    'Dimension MICCG solver variables.
Dim nip As Double, dip As Double      'Dimension MICCG solver variables.
Dim alpha As Double, beta As Double   'Dimension MICCG variables.
Dim xp() As Double, maxerr As Double  'Dimension MICCG solver variables.
Dim maxerrp As Double                 'Dimension prior max error.
Dim maxdelta As Double                'Dimension maximum head estimate change.
Dim resord As Integer                 'Dimension residual order.
Dim msgres As String                  'Dimension residual order message.
Dim kitr As Integer, kitri As Integer 'Dimension iteration trackers.
Dim kitrtr As Integer, kitro As Integer 'Dimension iteration trackers.
Dim GTC As Integer, GTCR As Integer   'Dimension GoTo codes.
Dim St As String                      'Dimension schedule type.
Dim Trans() As Integer                'Dimension pressurization transition flag.
Dim TransB() As Integer               'Dimension bed pressurization transition flag.
Dim transp() As Integer               'Dimension prior pressurization transition flag.
Dim TransBp() As Integer              'Dimension prior pressurization transition flag.
Dim TransT As Integer                 'Dimension pressurization transition aggregate flag.
Dim Transnm() As Integer              'Dimension period transition.
Dim Transpnm() As Integer             'Dimension prior period transition.
Dim IFT As Double                     'Dimension impact factor total.
Dim IGTC As Integer                   'Dimension impulse distribution GoTo code.
Dim dH As Double, Qt As Double        'Dimension Newton solver variables for impulse distribution.
Dim Qtt As Double, Qf As Double       'Dimension impulse distribution variables.
Dim ObF As Double, ObFF As Double     'Dimension Newton solver variables for impulse distribution.
Dim dFdu As Double                    'Dimension Newton solver variable, prior Qttn.
Dim dS As Double                      'Dimension change in storage for global budget.
Dim BD As Double, BLT As Double       'Dimension budget difference, budget largest term.
Dim PermFact() As Double              'Dimension permissive bed coefficient factor.
Dim tttb As Double                    'Dimension value holder to prevent duplication.
Dim HTS() As Double                   'Dimension head time series output.
Dim HTII As Integer, HTIJ As Integer  'Dimension head time series location im index, jm index.
Dim UMS() As Double, UMA() As Double  'Dimension Ultimate Matric Suction below streambed, aquifer top.
Dim UMAi() As Double, UMSi() As Double 'Dimension single index variables.
Dim Qttp As Double                    'Dimension previous impulse.
Dim Transpn() As Integer              'Dimension previous transition.
Dim flaw As Integer                   'Dimension flaw flag.
Dim Lons() As Integer, subsflag As Integer 'Dimension long side tracker, routine substitution flag.
Dim Lon1 As Integer, Lon2 As Integer  'Dimension long side roation trackers.
Dim msgu As String, msgumat As String 'Dimension matric suction messages.
Dim BedMat() As Integer               'Dimension streambed head condition flag.
Dim msgclt As String, msgwp As String 'Dimension Complete override, stream width messages.
Dim bppb() As Double                  'Dimension dummy variable for bank thickness.
Dim ph As Integer                     'Dimension stream orientation placeholder.
Dim delta As Double                   'Dimension MICCG upper preconditioner matrix diagonal scale factor to prevent negatives.
Dim PFSflag As Integer                'Dimension PermFact scale flag.
Dim MaxTerm As Double                 'Dimension maximum augmented matrix entry value.
Dim CTP() As Integer                  'Dimension neighboring stream cell type for complete permissive bed cases.
Dim BedCP() As Double                 'Dimension neighboring stream cell bed level for complete permissive bed cases.
Dim CTPflg As Integer                 'Dimension CTP flag for cases with relatively small channel dimensions.
Dim msgctp As String                  'Dimension CTP message.
Dim SEB As String                     'Dimension stream end bank option.
Dim msgrza As String                  'Dimension response zone ambiguity message.
Dim zone1 As Integer                  'Dimension single response zone indicator.
```

```
Dim Qb As Double                      'Dimension domain step impulse for budget block.

L = Sheet3.Cells(12, 8).Value         'Load number of cells on one side of model.
m = Sheet3.Cells(15, 8).Value         'Load number of cells on other side of model.
r = L * m                             'Number of rows in equation matrix.

no = Sheet11.Cells(13, 5).Value       'Load number of periods.
nc = Sheet11.Cells(16, 5).Value       'Load number of sub-period steps.
nn = no * nc                          'Set number of total steps.

ReDim dx(1 To L), dy(1 To m)          'Set custom array dimensions.
ReDim k(1 To m, 1 To L)
ReDim hoo(1 To m, 1 To L), ho(1 To m, 1 To L)
ReDim Hno(1 To m, 1 To L), Hn(1 To m, 1 To L)
ReDim Hnp(1 To m, 1 To L)
ReDim Qd(1 To m, 1 To L), Qm(1 To no)
ReDim Hnn(1 To m, 1 To L), Hnnp(1 To m, 1 To L)
ReDim Qnno(1 To m, 1 To L)
ReDim RSP(1 To nn), RSPA(1 To nn)
ReDim RSPO(1 To no)
ReDim RSPOA(1 To no)
ReDim CT(1 To m, 1 To L)
ReDim zone(1 To m, 1 To L)
ReDim DEC(1 To m, 1 To L)
ReDim Sy(1 To m, 1 To L), Ss(1 To m, 1 To L)
ReDim SOV(1 To no)
ReDim CUMRn(0 To no)
ReDim kpp(1 To m, 1 To L), bpp(1 To m, 1 To L), bppi(1 To r)
ReDim widp(1 To m, 1 To L), widpp(1 To m, 1 To L)
ReDim lenp(1 To m, 1 To L)
ReDim AqBi(1 To r), AqTi(1 To r)
ReDim Bed(1 To m, 1 To L), BEDi(1 To r), SHD(1 To m, 1 To L)
ReDim AqT(1 To m, 1 To L), AqB(1 To m, 1 To L)
ReDim WR(1 To m, 1 To L)
ReDim TipW(1 To 5)
ReDim Transpnm(1 To m, 1 To L)
ReDim Hni(1 To r), CTi(1 To r)
ReDim Qc(1 To m, 1 To L), QCP(1 To m, 1 To L)
ReDim HTS(0 To no)
ReDim UMS(1 To m, 1 To L), UMA(1 To m, 1 To L)
ReDim UMAi(1 To r), UMSi(1 To r)
ReDim PermFact(1 To m, 1 To L), Lons(1 To m, 1 To L)
ReDim Transpn(1 To r)
ReDim bppb(1 To m, 1 To L)
ReDim CTP(1 To m, 1 To L), BedCP(1 To m, 1 To L)

'
'INPUT
'-----------------------------------------------------------------------
RSPOFM = 1                          'Initialize flags all clear.
RSPOFM2 = 1
subsflag = 0

St = WorksheetFunction.Proper(Sheet11.Cells(12, 5).Value) 'Load schedule type.
SB = WorksheetFunction.Proper(Sheet12.Cells(7, 6).Value)  'Load relative permeability class of streambed.
SIT = WorksheetFunction.Proper(Sheet12.Cells(8, 6).Value) 'Load whether streambed incision represented.
```

```
SO = WorksheetFunction.Proper(Sheet11.Cells(19, 5).Value)      'Load secondary output type.
ITS = WorksheetFunction.Proper(Sheet3.Cells(7, 8).Value)       'Load Impulse Type Specification.
IMS = WorksheetFunction.Proper(Sheet8.Cells(7, 6).Value)       'Load Impulse Magnitude Specification.
ic = WorksheetFunction.Proper(Sheet11.Cells(17, 5).Value)      'Load Impulse Character.
ATC = WorksheetFunction.Proper(Sheet3.Cells(9, 8).Value)       'Load Aquifer Thickness Character.
AvTp = WorksheetFunction.Proper(Sheet3.Cells(10, 8).Value)     'Load characteristic average type.
ThickCon = WorksheetFunction.Proper(Sheet3.Cells(11, 8).Value)  'Load Thickness Contribution...Update Type.
HTII = Sheet11.Cells(24, 5).Value                               'Load head time series output location i index.
HTIJ = Sheet11.Cells(25, 5).Value                               'Load head time series output location j index.
flaw = 1                                                        'Initialize flaw flag.
SEB = WorksheetFunction.Proper(Sheet12.Cells(12, 14).Value)     'Load Stream End Bank Option.

epse = Sheet11.Cells(10, 5).Value    'Load closure threshold exponent.
eps = 1 * 10 ^ (-epse)               'Set convergence threshold value.

msg = Sheet11.Cells(30, 5).Value      'Load any warning message.

If ITS = "Well Pumping" Then          'Set flow signs for practical conventions.
 qs = 1
 qrs = -1
 Else
 qs = -1
 qrs = 1
End If
If WorksheetFunction.Proper(Sheet3.Cells(8, 8).Value) = "Acre-Feet" Then   'Set Unit Conversion Factor.
 UCF = 43560
 Else
 UCF = 1
End If
If WorksheetFunction.Proper(Sheet4.Cells(4, 4).Value) = "A" Then  'Load Min Sat Thick Allow.
 MSTA = Sheet4.Cells(15, 5).Value
 Else:
 MSTA = 0
End If

PD = WorksheetFunction.Proper(Sheet11.Cells(14, 5).Value) 'Load time unit denomination.
u = Sheet11.Cells(15, 5).Value       'Load units per period.
nh = Sheet11.Cells(23, 5).Value      'Load period to plot heads.
If PD = "Days" Then                  'Set time step.
 dt = u / nc
 ElseIf PD = "Months" Then
 dt = u * 365.25 / 12 / nc
 ElseIf PD = "Hours" Then
 dt = u / nc / 24
End If
If ic = "Steady" Then
 deno = nc
 Else
 deno = 1
End If

If WorksheetFunction.Proper(Sheet3.Cells(13, 8).Value) = "Uniform" Then  'Load x-axis increment sizes for uniform case.
im = 1
Do While im < L + 1
 dx(im) = Sheet3.Cells(14, 8).Value
 im = im + 1
```

```
Loop
Else
im = 1
Do While im < L + 1
 dx(im) = Sheet3.Cells(25, 3 + im).Value   'Load x-axis increment sizes for discrete case.
  im = im + 1
Loop
End If

If WorksheetFunction.Proper(Sheet3.Cells(16, 8).Value) = "Uniform" Then  'Load y-axis increment sizes for uniform case.
jm = 1
Do While jm < m + 1
 dy(jm) = Sheet3.Cells(17, 8).Value
 jm = jm + 1
Loop
Else
jm = 1
Do While jm < m + 1
 dy(jm) = Sheet3.Cells(26 + jm, 2).Value   'Load y-axis increment sizes for discrete case.
 jm = jm + 1
Loop
End If

n = 1
Do While n < no + 1
 Qm(n) = Sheet11.Cells(38 + n, 4).Value  'Load impulse magnitude time series.
 If n = 1 Then
 IMPSMX = Qm(1)
 IMPSMN = Qm(1)
 End If
 If IMPSMX < Qm(n) Then
  IMPSMX = Qm(n)
 End If
 If IMPSMN > Qm(n) Then
  IMPSMN = Qm(n)
 End If
 n = n + 1
Loop

For im = 1 To L
For jm = 1 To m
   CT(jm, im) = Sheet3.Cells(26 + jm, 3 + im).Value
Next jm
Next im

 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1                          'Parameterization block.
    PermFact(jm, im) = 1
    If CT(jm, im) = 2 Then
     If dy(jm) > dx(im) Then
      lenp(jm, im) = dy(jm)                     'Load stream lengths (longer of cell dimensions).
      widp(jm, im) = dx(im)
      Lons(jm, im) = 1
```

```
    Else
    lenp(jm, im) = dx(im)
    widp(jm, im) = dy(jm)
    Lons(jm, im) = 2
   End If
  Call Lon12(Lon1, Lon2, ph, jm, im, CT(), L, m)
  If Lon1 <> Lon2 Then
    If Lon1 > Lon2 Then
      Lons(jm, im) = 1
       lenp(jm, im) = dy(jm)
       widp(jm, im) = dx(im)
     Else
      Lons(jm, im) = 2
       lenp(jm, im) = dx(im)
       widp(jm, im) = dy(jm)
    End If
   End If
  End If
If ATC = "Confined" Then
 If WorksheetFunction.Proper(Sheet4.Cells(8, 5).Value) = "Uniform" Then
  AqT(jm, im) = -1 * Sheet4.Cells(9, 5).Value
  Else
  AqT(jm, im) = -1 * Sheet4.Cells(23 + jm, 2 + im).Value
 End If
End If
 If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
  AqB(jm, im) = -1 * Sheet4.Cells(12, 5).Value
  Else
  AqB(jm, im) = -1 * Sheet4.Cells(82 + jm, 2 + im).Value
 End If
   If WorksheetFunction.Proper(Sheet5.Cells(8, 5).Value) = "Uniform" Then
     DEC(jm, im) = -1 * Sheet5.Cells(9, 5).Value
     Else
      If CT(jm, im) = 2 Then
       If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
         DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
        Else
         DEC(jm, im) = 0
       End If
       Else
         DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
      End If
   End If
 If WorksheetFunction.Proper(Sheet6.Cells(8, 6).Value) = "Uniform" Then
  k(jm, im) = Sheet6.Cells(9, 6).Value
  Else
  k(jm, im) = Sheet6.Cells(16 + jm, 2 + im).Value
 End If
 If Sheet9.Cells(7, 6).Value = 1 Then
  zone(jm, im) = 1
  Else
  zone(jm, im) = Sheet9.Cells(15 + jm, 2 + im).Value
 End If
 If WorksheetFunction.Proper(Sheet7.Cells(8, 7).Value) = "Uniform" Then
  Sy(jm, im) = Sheet7.Cells(9, 7).Value
  Else
```

```
   Sy(jm, im) = Sheet7.Cells(18 + jm, 2 + im).Value
  End If
  If WorksheetFunction.Proper(Sheet7.Cells(10, 7).Value) = "Uniform" Then
   Ss(jm, im) = Sheet7.Cells(11, 7).Value
   Else
   Ss(jm, im) = Sheet7.Cells(76 + jm, 2 + im).Value
  End If
  If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) = "Uniform" Then
   Hn(jm, im) = -1 * Sheet10.Cells(9, 5).Value
   Else
   Hn(jm, im) = -1 * Sheet10.Cells(16 + jm, 2 + im).Value
  End If
  Hno(jm, im) = Hn(jm, im)
  Hnp(jm, im) = Hn(jm, im)
If WorksheetFunction.Proper(Sheet4.Cells(11, 14).Value) <> "None" Then
 If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
  UMA(jm, im) = Sheet4.Cells(12, 14).Value                    'Load Ultimate Matric Suction in Aquifer.
  Else
  UMA(jm, im) = Sheet4.Cells(200 + jm, 2 + im).Value
 End If
End If
  If CT(jm, im) = 2 Then
    If SB = "Restrictive" Then
     If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
      kpp(jm, im) = Sheet12.Cells(13, 6).Value
      bpp(jm, im) = Sheet12.Cells(14, 6).Value
      Else
      kpp(jm, im) = Sheet12.Cells(27 + jm, 2 + im).Value
      bpp(jm, im) = Sheet12.Cells(87 + jm, 2 + im).Value
     End If
     Else
     bpp(jm, im) = 0
    End If
    If SB <> "Permissive" Then
    If SIT <> "None" Then
    If WorksheetFunction.Proper(Sheet12.Cells(9, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       SHD(jm, im) = -1 * Sheet12.Cells(15, 6).Value
       Else
       SHD(jm, im) = -1 * Sheet12.Cells(147 + jm, 2 + im).Value
      End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
    If SIT = "None" Then
       Bed(jm, im) = SHD(jm, im)
       ElseIf SIT <> "None" Then
        If WorksheetFunction.Proper(Sheet12.Cells(10, 6).Value) = "Specified" Then
         If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
          Bed(jm, im) = -1 * Sheet12.Cells(16, 6).Value
```

```
          Else
          Bed(jm, im) = -1 * Sheet12.Cells(207 + jm, 2 + im).Value
         End If
         Else
         Bed(jm, im) = SHD(jm, im)
        End If
       End If
           If Bed(jm, im) > SHD(jm, im) Then
            Bed(jm, im) = SHD(jm, im)
            RSPOFM2 = 2
           End If
     If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       widpp(jm, im) = Sheet12.Cells(17, 6).Value
       Else
       widpp(jm, im) = Sheet12.Cells(267 + jm, 2 + im).Value
      End If
      Else
      widpp(jm, im) = widp(jm, im)
     End If
  If AqB(jm, im) >= Bed(jm, im) - bpp(jm, im) Then
  TipW(1) = 3
  End If
If SB = "Restrictive" Then
If SIT <> "None" Then
If WorksheetFunction.Proper(Sheet12.Cells(7, 14).Value) = "Simple" Then
If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
  UMS(jm, im) = Sheet12.Cells(8, 14).Value                      'Load Ultimate Matric Suction Below Bed.
  Else
  UMS(jm, im) = Sheet12.Cells(327 + jm, 2 + im).Value
End If
End If
 If UMA(jm, im) > UMS(jm, im) Then
   UMS(jm, im) = UMA(jm, im)
   msgu = "Matric Suction Value(s) Altered! "
 End If
End If
End If
   If Bed(jm, im) - bpp(jm, im) - AqB(jm, im) <= UMS(jm, im) Then
      UMS(jm, im) = Bed(jm, im) - bpp(jm, im) - AqB(jm, im)
   End If
  If widpp(jm, im) = widp(jm, im) Then
     UMA(jm, im) = UMS(jm, im)
  End If
   If SIT = "None" Then
     WR(jm, im) = 0
    ElseIf SIT <> "None" Then
     WR(jm, im) = widpp(jm, im) / widp(jm, im)
     If WR(jm, im) > 1 Then
      WR(jm, im) = 1
      widpp(jm, im) = widp(jm, im)
      msgwp = "Stream Width Limited to Cell Width. "
     End If
   End If
  ElseIf CT(jm, im) <> 2 Then
   WR(jm, im) = 0
```

```
   End If
   If CT(jm, im) <> 4 Then
    ho(jm, im) = hofunc(Hn(jm, im), AqT(jm, im), AqB(jm, im), UMA(jm, im), ATC, UMS(jm, im), Bed(jm, im), bpp(jm, im), WR(jm, im)) 'Saturated
thickness calculated by calling function to sort through conditions.
     If AqB(jm, im) + ho(jm, im) > DEC(jm, im) Then
       ho(jm, im) = DEC(jm, im) - AqB(jm, im)
     End If
    hoo(jm, im) = ho(jm, im)
   End If
  If CT(jm, im) = 1 Then
  If Right(IMS, 4) = "Head" Then
   If IMS = "Discrete, Head" Then
     Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
     ElseIf IMS = "Uniform, Head" Then
     Qd(jm, im) = 1
   End If
   ElseIf Right(IMS, 4) = "lume" Then
    If IMS = "Discrete, Volume" Then
     Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
     ElseIf IMS = "Uniform, Volume" Then
     Qd(jm, im) = 1
   End If
   End If
   If Qdmx = 0 Then
      Qdmx = Qd(jm, im)
      End If
      If Qdmn = 0 Then
      Qdmn = Qd(jm, im)
      End If
      If Qdmna = 0 Then
      Qdmna = Abs(Qd(jm, im))
      End If
      If Qd(jm, im) > Qdmx Then
      Qdmx = Qd(jm, im)
      End If
      If Qd(jm, im) < Qdmn Then
      Qdmn = Qd(jm, im)
      End If
      If Qd(jm, im) <> 0 Then
      If Abs(Qd(jm, im)) < Qdmna Then
      Qdmna = Abs(Qd(jm, im))
      End If
   End If
  End If
    If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
     If CT(jm, im) = 2 Then
     headsum2 = headsum2 + Hn(jm, im) * dx(im) * dy(jm) * Sfunc(2, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), Ss(jm,
im), Sy(jm, im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im))
     count2 = count2 + dx(im) * dy(jm) * Sfunc(2, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), Ss(jm, im), Sy(jm, im),
AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im))
     headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
     End If
     If CT(jm, im) Mod 2 = 1 Then
     headsum13 = headsum13 + Hn(jm, im) * dx(im) * dy(jm) * Sfunc(1, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im),
Ss(jm, im), Sy(jm, im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im))
```

```
     count13 = count13 + dx(im) * dy(jm) * Sfunc(1, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), Ss(jm, im), Sy(jm,
im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im))
     End If
   Else
     If CT(jm, im) = 2 Then
     headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
     End If
   End If
   If CT(jm, im) <> 4 Then
   If DEC(jm, im) > 0 Then
   TipW(1) = 1
   End If
   If Hn(jm, im) > DEC(jm, im) Then
   TipW(2) = 1
   End If
   If ATC = "Confined" Then
    If AqT(jm, im) >= Hn(jm, im) Then
    TipW(3) = 1
    End If
    If AqB(jm, im) >= AqT(jm, im) Then
    TipW(4) = 1
    End If
   Else
    If AqB(jm, im) >= Hn(jm, im) Then
    TipW(5) = 1
    End If
   End If
   End If
    AqBi(i) = AqB(jm, im)               'Fill single index variables.
    AqTi(i) = AqT(jm, im)
    BEDi(i) = Bed(jm, im)
    CTi(i) = CT(jm, im)
    bppi(i) = bpp(jm, im)
    UMAi(i) = UMA(jm, im)
    UMSi(i) = UMS(jm, im)
   i = i + 1
   im = im + 1
  Loop
  jm = jm + 1
 Loop

Call CTPload(CTPflg, m, L, CTP(), CT(), SB, SIT, _
             Lons(), SEB, flaw, msgctp, widpp(), widp(), _
             SHD(), AqB(), Bed(), BedCP(), zone(), msgrza, zone1)

 If HTII > 0 Then
    HTS(0) = -1 * Hn(HTIJ, HTII)
 End If

 If Qdmx > 0 Then
  If Qdmn < 0 Then
   msgim = "Mixed Impulse. "
  End If
 End If

If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
```

```
headave2 = headsum2 / count2
headave13 = headsum13 / count13
 If Abs(headave2 - headave13) > 0.000001 Then            'Check for initial head balance.
    RSPOFM = 2
 End If
End If
If Abs(headsum2s) > 0.000001 Then
   RSPOFM = 4
End If

If TipW(1) + TipW(2) + TipW(4) + TipW(5) = 0 Then                                'Condition routine on valid thickness input, end if near end of
subroutine.

'
'TIME STEP COMPUTATION LOOP
'---------------------------------------------------------------------
NegHead = 1

n = 1                              'Initialize time step counter.
o = 1
nm = 1
Do While n < nn + 1                'Loop for time steps!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

GTC = 0                            'Initialize codes for each time step.
kitro = 0
kitrtr = 0
TransT = 0
delta = 0
PFSflag = 0
MaxTerm = 0

 If n > 1 Then
 Qttp = Qtt
   For i = 1 To r
    Transpn(i) = Trans(i)
   Next i
 End If
ReDim Trans(1 To r), TransB(1 To r)
ReDim transp(1 To r), TransBp(1 To r)
If o = 1 Then
 ReDim Transnm(1 To m, 1 To L)
End If

If o > 1 Then                                  'IMPULSE DISTRIBUTION BLOCK.+++++++++++++++
  If ic = "Instantaneous" Then
    ReDim Qc(1 To m, 1 To L)
    dH = 0
    Qtt = 0
    kitr = 0
    Qb = 0
  End If
    GoTo Line8
End If
If ic = "Steady" Then
   Qb = -1 * UCF * Qm(nm) / nc
   Qm(nm) = Qm(nm) / nc
```

```
  Else
   Qb = -1 * UCF * Qm(nm)
End If
'||||
dH = 0
Qf = 0
  IA = 0
  IAP = 0
    jm = 1
    Do While jm < m + 1                                'TALLEY RELATIVE IMPULSE MAGNITUDE WEIGHTING.
      im = 1
      Do While im < L + 1
        If CT(jm, im) = 1 Then                           'CT if - impulse cells.
         If Right(IMS, 4) = "lume" Then                      'IMS if - Volumetric impulse distribution case.
          IA = IA + Qd(jm, im)
          If Qd(jm, im) > 0 Then                         'Alternate denominator for zero net impulse case, by volume.
            IAP = IAP + Qd(jm, im)
          End If                                         'Denominator end if.
          ElseIf Right(IMS, 4) = "Head" Then                 'IMS ElseIf.
          IA = IA + (Qd(jm, im) * dx(im) * dy(jm) * Sfunc(1, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), Ss(jm, im),
Sy(jm, im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im)))
          If Qd(jm, im) > 0 Then                         'Alternate denominator for zero net impulse case.
            IAP = IAP + (Qd(jm, im) * dx(im) * dy(jm) * Sfunc(1, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), Ss(jm,
im), Sy(jm, im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im)))
          End If                                         'Denominator end if.
        End If                                               'IMS end if.
        End If                                           'CT end if
        im = im + 1                                      'Increment to loop.
      Loop
      jm = jm + 1
    Loop
If IMS = "Uniform, Head" Then                  'IMS if.++++++++++++++++++++++++++++++++++
    Qf = UCF * qs * Qm(nm)
    dH = -1 * Qf * Qdmna / IA
     GTC = 1
     GoTo Line4
Line1:   '=========================
ElseIf IMS = "Uniform, Volume" Then            'IMS elseif.++++++++++++++++++++++++++++++
   jm = 1                                                 'Initialize loops, separate to allow prior loops to total denominator, AI.
   Do While jm < m + 1
   im = 1                                             'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   Do While im < L + 1
   If CT(jm, im) = 1 Then                             'CT if - only impulse cells.
    Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IA 'Fill impulse volume.
   End If                                             'CT end if
   im = im + 1                                        'Increment to loop.
   Loop
   jm = jm + 1
  Loop
ElseIf IMS = "Discrete, Head" Then             'IMS elseif.++++++++++++++++++++++++++++++
  If msgim = "Mixed Impulse. " Then
     msgti = "Discrete Impulse Distribution by Head Requires Like Signs Throughout Grid. Distribute by Volume Instead. "
     GoTo Line50
  End If
    If IA = 0 Then
     msgti = "Discrete Impulse Distribution by Head Requires Non-zero Weighting. "
```

```
     GoTo Line50
    End If
    If IA > 0 Then                                          'IA If.
         Qf = UCF * qs * Qm(nm)
         dH = -1 * Qf * Qdmna / IA
       ElseIf IA < 0 Then                                   'IA ElseIf.
         Qf = UCF * qs * Qm(nm) * (-1)
         dH = -1 * Qf * Qdmna / IA
    End If                                                  'IA end if.
     GTC = 2
     GoTo Line4
Line2:   '=========================
ElseIf IMS = "Discrete, Volume" Then           'IMS elseif.++++++++++++++++++++++++++++++
   jm = 1                                                    'Initialize loops, separate to allow prior loops to total denominator, AI.
   Do While jm < m + 1
   im = 1                                                  'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   Do While im < L + 1
   If CT(jm, im) = 1 Then                                  'CT if - only impulse cells.
    If IA <> 0 Then                                        'IA If.
    Qc(jm, im) = UCF * qs * Qm(nm) * Abs(Qd(jm, im)) / IA 'Fill impulse volume.
    Else
      If IAP > 0 Then                                        'IAP if.
      msgim2 = "10"               'Zero NET impulse case.
      Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IAP
      Else
      msgim2 = "11"                     'Zero impulse case.
      GoTo Line50
      End If                                                 'End IAP end if.
    End If                                                   'IA end if.
   End If                                                  'CT end if
   im = im + 1                                             'Increment to loop.
   Loop
   jm = jm + 1
  Loop
End If                                         'IMS end if.
'||||
  If ic = "Instantaneous" Then                 'ic if.++++++++++++++++++++++++++++++++++
  Qtt = 0
  i = 1
  jm = 1                                                     'Initialize loops, separate to allow prior loops to total denominator, AI.
  Do While jm < m + 1
   im = 1                                                 'CONVERT INSTANTANEOUS IMPULSE TO HEAD.
   Do While im < L + 1
   If CT(jm, im) = 1 Then                                  'CT if - only impulse cells.
          Qtt = Qtt + Qc(jm, im)                           'For Domain Volumetric Budget Later.
          If ATC = "Confined" Then                         'ATC if - confined.
           If Hn(jm, im) >= AqT(jm, im) - UMA(jm, im) Then
            If Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im))) < AqT(jm, im) - UMA(jm, im) Then
'Depressurization.
             Hn(jm, im) = (AqT(jm, im) - UMA(jm, im)) - (Qc(jm, im) - (Hn(jm, im) - (AqT(jm, im) - UMA(jm, im))) * dx(im) * dy(jm) * Ss(jm, im) *
(AqT(jm, im) - AqB(jm, im))) / (dx(im) * dy(jm) * Sy(jm, im))
    If msgti = "" Then
    msgti = "Impulse Distribution Through Confinement Transition. "
    End If
           Else
            Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im))) 'Pressurized.
```

```
          End If
          ElseIf Hn(jm, im) < AqT(jm, im) - UMA(jm, im) Then
           If Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Sy(jm, im)) > AqT(jm, im) - UMA(jm, im) Then 'Pressurization.
             Hn(jm, im) = (AqT(jm, im) - UMA(jm, im)) - (Qc(jm, im) - (Hn(jm, im) - (AqT(jm, im) - UMA(jm, im))) * dx(im) * dy(jm) * Sy(jm, im)) /
(dx(im) * dy(jm) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im)))
    If msgti = "" Then
    msgti = "Impulse Distribution Through Confinement Transition. "
    End If
            Else
             Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Sy(jm, im)) 'Depressurized.
           End If
          End If
         Else                                        'ATC else - unconfined.
          Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Sy(jm, im))
        End If                                       'ATC end if.
        If Hn(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
         If ATC = "Unconfined" Then
          GoTo Line3
         Else
           If ITS = "Deep Percolation" Then
Line3:   '==========================
             msgc = "Inundated Ground! "
             msgti = "Impulse Exceeds Aquifer Capacity. "
             flaw = 3
             NegHead = 6
             NegHeadNM = nm + 38
             GoTo Line50
           End If
         End If
        End If
        Qc(jm, im) = 0                               'Annul converted flow (instantaneous).
      End If                                         'CT end if.
      i = i + 1
      im = im + 1                                        'Increment to loop.
      Loop
    jm = jm + 1
    Loop
    ElseIf ic = "Steady" Then                    'ic elseif.++++++++++++++++++++++++++++++
    Qtt = 0
    For jm = 1 To m                                    'CONVERT STEADY IMPULSE TO FLOW.
    For im = 1 To L
      If CT(jm, im) = 1 Then
        Qtt = Qtt + Qc(jm, im)                       'For Domain Volumetric Budget Later.
        Qc(jm, im) = Qc(jm, im) / (dt)               'Convert to volume per time.
      End If
   Next im
   Next jm
  End If                                         'ic end if.
'||||
If ic = "Steady" Then                            'Restore Qm for later use.
    Qm(nm) = Qm(nm) * nc
End If
GoTo Line8                                       'Direct past solver.++++++++++++++++++++++
'||||>
Line4:          '==========================
'START NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION OVER GRID BY HEAD]]]]]]]]]]
```

```
   kitr = 0                                        'Initialize iteration counter.
   IGTC = 0                                        'Initialize impulse go to code.
   ObF = 1                                         'Initialize objective function.
   Do While Abs(ObF) > 0.0000001        'Solver loop.
   If kitr < 500 Then                             'Iteration limit check
     kitr = kitr + 1                              'Increment iteration counter.
Line5:          '==========================
   Qt = 0                                          'Initialize virtual volume tally.
   '((((>
   i = 1
   jm = 1                                          'Prepare for calculation loop.
   Do While jm < m + 1
   im = 1
   Do While im < L + 1                             'Calculation loop.
   If CT(jm, im) = 1 Then                          'CT if - cell type check.
   If ATC = "Confined" Then                        'ATC if - confined.
           If ic = "Steady" Then                   'ic if.
           If Hn(jm, im) >= AqT(jm, im) - UMA(jm, im) Then
             Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im)) * dx(im) * dy(jm)
           ElseIf Hn(jm, im) < AqT(jm, im) - UMA(jm, im) Then
             Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * Sy(jm, im) * dx(im) * dy(jm)
           End If
           Else                                    'ic else - instantaneous.
           If Hn(jm, im) >= AqT(jm, im) - UMA(jm, im) Then
            If Hn(jm, im) + (Qd(jm, im) * dH / Qdmna) < AqT(jm, im) - UMA(jm, im) Then 'Depressurization.
             Qc(jm, im) = ((Hn(jm, im) - (AqT(jm, im) - UMA(jm, im))) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im)) + ((AqT(jm, im) - UMA(jm, im)) -
(Hn(jm, im) + (Qd(jm, im) * dH / Qdmna))) * Sy(jm, im)) * dx(im) * dy(jm)
            Else                                      'Pressurized.
             Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im)) * dx(im) * dy(jm)
            End If
           ElseIf Hn(jm, im) < AqT(jm, im) - UMA(jm, im) Then
            If Hn(jm, im) + (Qd(jm, im) * dH / Qdmna) > AqT(jm, im) - UMA(jm, im) Then 'Pressurization.
             Qc(jm, im) = ((Hn(jm, im) - (AqT(jm, im) - UMA(jm, im))) * Sy(jm, im) + ((AqT(jm, im) - UMA(jm, im)) - (Hn(jm, im) + (Qd(jm, im) * dH /
Qdmna))) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im))) * dx(im) * dy(jm)
            Else                                      'Depressurized.
             Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * Sy(jm, im) * dx(im) * dy(jm)
            End If
           End If
           End If                                 'ic end if.
         Else                                   'ATC else - unconfined.
            Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * Sy(jm, im) * dx(im) * dy(jm)
     End If                                          'ATC end if.
     If IA <> 0 Then
     Qt = Qt + Qc(jm, im)                        'For distribution by Newton Solver.
     Else
     If Qd(jm, im) > 0 Then
     Qt = Qt + Qc(jm, im)
     End If
     End If
  End If                                        'CT end if.
   i = i + 1
   im = im + 1
   Loop
   jm = jm + 1
   Loop                                            'Close calculation loop.
   '((((>
```

```
   If IGTC = 6 Then
   IGTC = 0
   GoTo Line6
   End If
   '((((>
   ObF = Qf - Qt                              'Compute objective function [F(n)].
  If Abs(ObF) <= 0.0000001 Then
  GoTo Line7
  End If
  dH = (1 + 0.00000001) * dH                   'Increment head increment.
   IGTC = 6
   GoTo Line5
Line6:   '==========================
  ObFF = Qf - Qt                             'Recompute objective function [F(n+1)].
  dFdu = (ObFF - ObF) / (dH - dH / (1 + 0.00000001)) 'Calculate derivative.
  dH = dH / (1 + 0.00000001)                 'Restore head increment.
  dH = dH - ObF / dFdu                       'Compute next estimate for head increment.
Line7:   '==========================
  Else                                                'Intervene to end iteration if not converging in a reasonable number of cycles.
   NegHead = 4
   NegHeadNM = nm + 38
   GoTo Line50
  End If                                              'Iteration counter end if.
  Loop                                       'Close solver loop.
'END NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION]]]]]]]]]]
If GTC = 1 Then
 GoTo Line1
ElseIf GTC = 2 Then
 GoTo Line2
End If                                  'END IMPULSE DISTRIBUTION BLOCK.++++++++++

Line8: '==========================

ReDim hx(1 To r)
i = 1                                           'Re-initialize head vectors.
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1
If CT(jm, im) <> 4 Then
  If Hn(jm, im) <= AqB(jm, im) Then                    'Check that head not beneath aquifer.
    NegHead = 2
    NegHeadNM = nm + 38
    GoTo Line50
  Else                                                'Thickness update.
ho(jm, im) = hofunc(Hn(jm, im), AqT(jm, im), AqB(jm, im), UMA(jm, im), ATC, UMS(jm, im), Bed(jm, im), bpp(jm, im), WR(jm, im))
     If AqB(jm, im) + ho(jm, im) > DEC(jm, im) Then
       ho(jm, im) = DEC(jm, im) - AqB(jm, im)
     End If
  End If
End If
Hni(i) = Hn(jm, im)                                   'Fill vectors.
hx(i) = Hn(jm, im)
    im = im + 1                                       'Increment to loop.
    i = i + 1
  Loop
```

```
  jm = jm + 1
 Loop

Line9: '=========================       'Line label for point of return, if pressurization transition in first set of iterations, and for outside
iteration.
'
'FILL PRIMARY COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------------------------
ReDim AA(1 To m, 1 To L), BB(1 To m, 1 To L), CC(1 To m, 1 To L)  'Redimension A, B, C to reset all to empty.
ReDim DD(1 To m, 1 To L), EE(1 To m, 1 To L), b(1 To m, 1 To L)   'Redimension D, E, b to reset all to empty.
ReDim BedMat(1 To m, 1 To L)
 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1                        'Calculate coefficient values.
   If CT(jm, im) <> 4 Then
    If jm - 1 > 0 Then
    If CT(jm - 1, im) < 4 Then
      AA(jm, im) = khbar(SIT, AvTp, k(jm - 1, im), ho(jm - 1, im), dy(jm - 1), WR(jm - 1, im), hx(i - L), UMS(jm - 1, im), Bed(jm - 1, im), bpp(jm -
1, im), AqB(jm - 1, im), _
                         k(jm, im), ho(jm, im), dy(jm), WR(jm, im), hx(i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dx(im), UMA(jm - 1,
im), UMA(jm, im), ATC, _
                         AqT(jm - 1, im), AqT(jm, im), Lons(jm - 1, im), Lons(jm, im), CT(jm - 1, im), CT(jm, im), 2, SB)
    End If
   End If
   If im - 1 > 0 Then
    If CT(jm, im - 1) < 4 Then
      BB(jm, im) = khbar(SIT, AvTp, k(jm, im - 1), ho(jm, im - 1), dx(im - 1), WR(jm, im - 1), hx(i - 1), UMS(jm, im - 1), Bed(jm, im - 1), bpp(jm,
im - 1), AqB(jm, im - 1), _
                         k(jm, im), ho(jm, im), dx(im), WR(jm, im), hx(i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dy(jm), UMA(jm, im -
1), UMA(jm, im), ATC, _
                         AqT(jm, im - 1), AqT(jm, im), Lons(jm, im - 1), Lons(jm, im), CT(jm, im - 1), CT(jm, im), 1, SB)
    End If
   End If
   If jm + 1 < m + 1 Then
    If CT(jm + 1, im) < 4 Then
      CC(jm, im) = khbar(SIT, AvTp, k(jm + 1, im), ho(jm + 1, im), dy(jm + 1), WR(jm + 1, im), hx(i + L), UMS(jm + 1, im), Bed(jm + 1, im), bpp(jm +
1, im), AqB(jm + 1, im), _
                         k(jm, im), ho(jm, im), dy(jm), WR(jm, im), hx(i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dx(im), UMA(jm + 1,
im), UMA(jm, im), ATC, _
                         AqT(jm + 1, im), AqT(jm, im), Lons(jm + 1, im), Lons(jm, im), CT(jm + 1, im), CT(jm, im), 2, SB)
    End If
   End If
   If im + 1 < L + 1 Then
    If CT(jm, im + 1) < 4 Then
      DD(jm, im) = khbar(SIT, AvTp, k(jm, im + 1), ho(jm, im + 1), dx(im + 1), WR(jm, im + 1), hx(i + 1), UMS(jm, im + 1), Bed(jm, im + 1), bpp(jm,
im + 1), AqB(jm, im + 1), _
                         k(jm, im), ho(jm, im), dx(im), WR(jm, im), hx(i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dy(jm), UMA(jm, im +
1), UMA(jm, im), ATC, _
                         AqT(jm, im + 1), AqT(jm, im), Lons(jm, im + 1), Lons(jm, im), CT(jm, im + 1), CT(jm, im), 1, SB)
     End If
   End If
  End If
  EE(jm, im) = EE(jm, im) + FuncE(AA(jm, im), BB(jm, im), CC(jm, im), DD(jm, im), Trans(i), TransB(i), CT(jm, im), dx(im), dy(jm), dt, WR(jm, im),
ATC, AqT(jm, im), AqB(jm, im), Ss(jm, im), Sy(jm, im), SIT, Bed(jm, im), bpp(jm, im), hx(i), SB, Hn(jm, im), UMA(jm, im), UMS(jm, im))
```

```
  b(jm, im) = b(jm, im) + FuncB(Qc(jm, im), Trans(i), TransB(i), CT(jm, im), dx(im), dy(jm), dt, WR(jm, im), ATC, Hn(jm, im), AqT(jm, im), AqB(jm,
im), Ss(jm, im), Sy(jm, im), SIT, Bed(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im))
   If CT(jm, im) = 2 Then
   If SB = "Restrictive" Then
    If hx(i) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      EE(jm, im) = EE(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) / bpp(jm, im)                          'Includes full bed flux term.
      b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * SHD(jm, im) / bpp(jm, im)             'Includes full bed flux term.
     ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) / bpp(jm, im)
'Bed flux limited (drawn down below bottom).
     End If
     End If
     End If
    i = i + 1
    im = im + 1
   Loop
   jm = jm + 1
 Loop

 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1
   If CTP(jm, im) = 4 Then
    GoTo line9e
   ElseIf CTP(jm, im) = 2 Then
   If SB = "Permissive" Then
line9e:      '=========================
    AA(jm, im) = 0
    BB(jm, im) = 0
    CC(jm, im) = 0
    DD(jm, im) = 0
    EE(jm, im) = PermFact(jm, im) * dx(im) * dy(jm) / dt                    'Factor scales coefficients for procedure stability without affecting
result.
    b(jm, im) = -EE(jm, im) * Hnp(jm, im)
   End If
   End If
    If PFSflag < 1 Then
     If MaxTerm < Abs(EE(jm, im)) Then
      MaxTerm = Abs(EE(jm, im))
     End If
    End If
    i = i + 1
    im = im + 1
   Loop
   jm = jm + 1
 Loop

If PFSflag < 1 Then
   PFSflag = 1
End If

'
'WRITE AUGMENTED MATRIX BY POPULATING COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------------------------
```

```
ReDim SE(1 To r, 1 To 5), SEC(1 To r), UM(1 To r, 1 To 3), UT(1 To r, 1 To 3), DU(1 To r, 1 To 3)       'Redimension matrices to obviate memory
errors(row limit, column limit).
i = 1                                'Initialize system of equations row index, also index for model cell identification serial.
For jm = 1 To m                      'Initialize model space row index.  Note: convention for matrix indeces is different than model.  Model columns
are im, Matrix columns are j.
  For im = 1 To L                           'Initialize and reset model space column index.

   SE(i, 3) = -EE(jm, im)         'Coefficient for cell's own next-step head.
   SEC(i) = b(jm, im)             'Constant or intercept value for row.
   If m > 1 Then
   If L > 1 Then
   If jm + 1 < m + 1 Then
    SE(i, 5) = CC(jm, im)          'Coefficient for next-step head in model cell beneath, matrix place L to the right.
   End If
   If jm - 1 > 0 Then
    SE(i, 1) = AA(jm, im)          'Coefficient for next-step head in model cell above, matrix place L to the left.
   End If
   End If
   End If
   If L > 1 Then
    If im + 1 < L + 1 Then
     SE(i, 4) = DD(jm, im)          'Coefficient for next-step head in model cell to right, matrix place one to the right.
    End If
    If im - 1 > 0 Then
     SE(i, 2) = BB(jm, im)          'Coefficient for next-step head in model cell to the left, matrix place one to the left.
    End If
   Else
    If jm + 1 < m + 1 Then
     SE(i, 4) = CC(jm, im)          'Coefficient for next-step head in model cell beneath, matrix place one to the right.
    End If
    If jm - 1 > 0 Then
     SE(i, 2) = AA(jm, im)          'Coefficient for next-step head in model cell above, matrix place one to the left.
    End If
   End If

   SE(i, 1) = -1 * SE(i, 1)
   SE(i, 2) = -1 * SE(i, 2)
   SE(i, 3) = -1 * SE(i, 3)
   SE(i, 4) = -1 * SE(i, 4)
   SE(i, 5) = -1 * SE(i, 5)
   SEC(i) = -1 * SEC(i)


    If CTP(jm, im) = 4 Then
     GoTo Line9f
    ElseIf CTP(jm, im) = 2 Then
    If SB = "Permissive" Then
Line9f:      '=========================
     If PFSflag < 2 Then
      If Abs(SE(i, 3)) < 0.75 * MaxTerm Then
       PermFact(jm, im) = PermFact(jm, im) * WorksheetFunction.RoundUp(0.75 * MaxTerm / SE(i, 3), 0) 'Increment factor to recompute coefficient for
mathematical stability.
       subsflag = 1
      End If
     End If
    End If
```

```
     End If

   i = i + 1
  Next im
Next jm
    If subsflag = 1 Then
     PFSflag = 2
     subsflag = 0
     GoTo Line9
    End If

Line9g:      '=========================
 For i = 1 To r
  UM(i, 1) = (1 + delta) * SE(i, 3)            'Modified incomplete Cholesky factorization upper conditioner matrix.
 Next i                                             'Initial loop sets all uii values to aii, unless delta non-zero.
 For i = 1 To r
  If i - 1 > 0 Then
   UM(i, 1) = UM(i, 1) - (SE(i - 1, 4) ^ 2) / UM(i - 1, 1)                                    'Xi, Hill pg. 10.
   UM(i, 1) = UM(i, 1) - SE(i - 1, 4) * SE(i - 1, 5) / UM(i - 1, 1)                           'Phi.
     If i + L - 1 <= r Then
       UM(i + L - 1, 1) = UM(i + L - 1, 1) - SE(i - 1, 4) * SE(i - 1, 5) / UM(i - 1, 1)
     End If
   If i - L > 0 Then
    UM(i, 1) = UM(i, 1) - (SE(i - L, 5) ^ 2) / UM(i - L, 1)                                   'Tau.
   End If
  End If
  If UM(i, 1) < 0.001 Then
   If delta < 1000000 Then
     delta = 1.5 * delta + 0.001
     GoTo Line9g
   End If
  End If
  If i < r Then
   UM(i, 2) = SE(i, 4)
  End If
  If i <= r - L Then
   UM(i, 3) = SE(i, 5)
  End If
  DU(i, 1) = 1 'UM(i, 1) * (1 / UM(i, 1)) 'Diagonal matrix of 1/UM(i,1) times UM.
  DU(i, 2) = UM(i, 2) / UM(i, 1)
  DU(i, 3) = UM(i, 3) / UM(i, 1)
 Next i

For i = 1 To r                          'Transpose of UM carried out for condensed format.
UT(i, 3) = UM(i, 1)
If i - 1 > 0 Then
UT(i, 2) = UM(i - 1, 2)
End If
If i - L > 0 Then
UT(i, 1) = UM(i - L, 3)
End If
Next i

'SIMULTANEOUS SOLUTION OF SYSTEM OF EQUATIONS
'-----------------------------------------------------------------------------------------
'Modified Incomplete Cholesky Preconditioned CONJUGATE GRADIENT METHOD (MICCG)!!
```

```
'SEE "Preconditioned Conjugate-Gradient 2 (PCG2), A Computer Program for Solving Ground-Water flow Equations"
'by Mary C. Hill, USGS Water Resources Investigations Report 90-4048. 1990.
Line10: '==========================       'Line label for top of solver.
ReDim Preserve SE(1 To r, 1 To 5), SEC(1 To r), hx(1 To r)
kitri = 0                      'Initialize iteration counter.
maxerrp = 1E+99                'Initialize prior iteration closure.
maxerr = 1                     'Initialize iteration closure.
GTCR = 0                       'Initialize GoTo Code for Residual Rounding.
Line11: '==========================
'Initialize Ax, rr, zz.
'-----------------quick homemade array multiplication
ReDim Ax(1 To r)
For i = 1 To r
Ax(i) = SE(i, 3) * hx(i)
If m > 1 Then
If L > 1 Then
  If i - L > 0 Then
   Ax(i) = Ax(i) + SE(i, 1) * hx(i - L)
  End If
  If i + L < r + 1 Then
   Ax(i) = Ax(i) + SE(i, 5) * hx(i + L)
  End If
End If
End If
 If i - 1 > 0 Then
  Ax(i) = Ax(i) + SE(i, 2) * hx(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 4) * hx(i + 1)
 End If
Next i
'-----------------
ReDim Preserve Ax(1 To r)
ReDim rr(1 To r)
For i = 1 To r          'method step
rr(i) = SEC(i) - Ax(i)
Next i
          If GTCR = 2 Then
            GoTo Line14
          End If
Do While maxerr > eps                        '#############Top of Method loop.
If kitri < 1000 Then                           'Iteration limit check.
Line11b: '==========================
ReDim Preserve UT(1 To r, 1 To 3), rr(1 To r)
ReDim vv(1 To r)
For i = 1 To r          'forward sub
vv(i) = rr(i)
If L > 1 Then
 If m > 1 Then
  If i - L > 0 Then
   vv(i) = vv(i) - UT(i, 1) * vv(i - L)
  End If
 End If
End If
If i - 1 > 0 Then
vv(i) = vv(i) - UT(i, 2) * vv(i - 1)
```

```
End If
vv(i) = vv(i) / UT(i, 3)
Next i
ReDim Preserve DU(1 To r, 1 To 3), vv(1 To r)
ReDim zz(1 To r)
i = r
Do While i > 0           'back sub
zz(i) = vv(i)
If L > 1 Then
 If m > 1 Then
  If i + L < r + 1 Then
    zz(i) = zz(i) - DU(i, 3) * zz(i + L)
  End If
 End If
End If
If i + 1 < r + 1 Then
zz(i) = zz(i) - DU(i, 2) * zz(i + 1)
End If
'zz(i) = zz(i) / DU(i, 1)  DU(i,1)=1, so no need to divide.
i = i - 1
Loop
ReDim Preserve zz(1 To r)
          If GTCR = 1 Then
            GoTo Line12
          End If
If kitri = 0 Then
Line12: '==========================
 ReDim pp(1 To r)
 nip = 0
 For i = 1 To r
 pp(i) = zz(i)
 nip = nip + zz(i) * rr(i)
 Next i
           If GTCR = 1 Then
            ReDim Preserve pp(1 To r)
            GoTo Line13
           End If
ElseIf kitri > 0 Then
 dip = nip
 nip = 0
 For i = 1 To r
 nip = nip + zz(i) * rr(i)
 Next i
 beta = nip / dip
 For i = 1 To r
 pp(i) = zz(i) + beta * pp(i)
 Next i
End If
ReDim Preserve pp(1 To r)
kitri = kitri + 1         'Update iteration counter.
'-----------------array multiplication
Line13: '==========================
ReDim Preserve SE(1 To r, 1 To 5)
ReDim Ap(1 To r)
For i = 1 To r
Ap(i) = SE(i, 3) * pp(i)
```

```
If m > 1 Then
If L > 1 Then
  If i - L > 0 Then
   Ap(i) = Ap(i) + SE(i, 1) * pp(i - L)
  End If
  If i + L < r + 1 Then
   Ap(i) = Ap(i) + SE(i, 5) * pp(i + L)
  End If
End If
End If
 If i - 1 > 0 Then
  Ap(i) = Ap(i) + SE(i, 2) * pp(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 4) * pp(i + 1)
 End If
Next i
ReDim Preserve Ap(1 To r)
'-----------------
dip = 0                    'Remainder of method steps
For i = 1 To r             'For-Next gives internal products.
If Abs(nip) < 1E+100 Then      'Confirm stability.
dip = dip + pp(i) * Ap(i)
Else
 NegHead = 3
 NegHeadNM = nm + 38       'Intervene if not stable.
 GoTo Line50
End If
Next i
If nip <> 0 Then           'Zero residual check.
alpha = nip / dip
ReDim xp(1 To r)
For i = 1 To r             'Calculate next estimates.
xp(i) = hx(i)
hx(i) = hx(i) + alpha * pp(i)
rr(i) = rr(i) - alpha * Ap(i)
Next i
ReDim Preserve rr(1 To r), xp(1 To r), hx(1 To r)
           If GTCR = 1 Then
            GTCR = 2
            GoTo Line11
           ElseIf GTCR = 2 Then
            GoTo Line11
           End If
Line14: '==========================
Else               'Zero residual upshot.
 ReDim xp(1 To r)
 For i = 1 To r
  xp(i) = hx(i)
  hx(i) = hx(i)
  rr(i) = rr(i)
 Next i
End If             'Zero residual end if.

maxerr = 0
maxdelta = 0
```

```
For i = 1 To r                 'Calculate closure difference.
  If Abs(hx(i) - xp(i)) > maxerr Then
    maxerr = Abs(hx(i) - xp(i))
  End If
  If Abs(hx(i) - xp(i)) > maxdelta Then
    maxdelta = Abs(hx(i) - xp(i))
  End If
  If Abs(rr(i)) > maxerr Then
    maxerr = Abs(rr(i))
  End If
Next i
If maxdelta < 0.00000000001 Then
 If maxerr > eps Then
   If kitro + kitrtr < 100 Then
     maxerr = 0.5 * eps
     If kitri = 1 Then
      kitri = 2
     End If
    Else
     resord = -1 * WorksheetFunction.RoundDown(WorksheetFunction.Log10(maxerr), 0)
     eps = 1 * 10 ^ (-resord)
     msgres = "Residual Order = " & resord & ". "
   End If
 End If
End If

    If maxerrp <= maxerr Then                  'Watches for rounding error; if error evident, recalculates residual, instead of using erroneous
values.
    If GTCR = 0 Then
        GTCR = 1
        For i = 1 To r
            hx(i) = xp(i)
        Next i
        GoTo Line11
      ElseIf kitri > 50 Then
        subsflag = 76
    End If
    End If

Line15: '==========================

TransT = 0
i = 1
For jm = 1 To m
For im = 1 To L
Trans(i) = 0
TransB(i) = 0
If CTi(i) <> 4 Then
    If ATC = "Confined" Then
     If Hni(i) >= AqTi(i) - UMAi(i) Then
      If hx(i) < AqTi(i) - UMAi(i) Then
       Trans(i) = 1                    'Depressurization code.
      End If
      ElseIf Hni(i) < AqTi(i) - UMAi(i) Then
      If hx(i) >= AqTi(i) - UMAi(i) Then
       Trans(i) = 2                    'Pressurization code.
```

```
      End If
     End If
     If SB = "Permissive" Then
     If CTP(jm, im) = 2 Then
          Trans(i) = 0
     End If
     End If
    End If
    If CTi(i) = 2 Then
     tttb = TTB(SIT, ATC, AqTi(i), BEDi(i), bppi(i), UMSi(i), UMAi(i))
     If (Hni(i) + UMSi(i)) >= tttb Then
      If (hx(i) + UMSi(i)) < tttb Then
       TransB(i) = 1
      End If
      ElseIf (Hni(i) + UMSi(i)) < tttb Then
      If (hx(i) + UMSi(i)) >= tttb Then
       TransB(i) = 2
      End If
     End If
     If SB = "Permissive" Then
          TransB(i) = 0
     End If
    End If
    If Trans(i) <> transp(i) Then
     TransT = 1
    End If
    If TransB(i) <> TransBp(i) Then
     TransT = 1
    End If
End If
i = i + 1
Next im
Next jm
If subsflag = 1 Then
  subsflag = 0
  GoTo Line48
End If
If subsflag = 76 Then
  subsflag = 0
  GoTo Line16
End If
If subsflag = 77 Then
  subsflag = 0
  GoTo Line16b
End If

If kitro + kitrtr < 200 Then
If kitri > 100 Then
Line15a: '==========================

  If TransT = 1 Then
   GoTo Line16
  End If

i = 1
For jm = 1 To m
```

```
For im = 1 To L
       If CT(jm, im) <> 4 Then
       If hx(i) <= AqBi(i) Then
         GoTo Line16
       End If
       End If
i = i + 1
Next im
Next jm

End If               'kitri.
Else
If kitri > 4 Then
GoTo Line15a
End If
End If

maxerrp = maxerr  'Set prior maxerr.

Else                                              'Intervene to end iteration if not converging in a reasonable number of cycles.
 NegHead = 3
 NegHeadNM = nm + 38
 GoTo Line50
End If                                            'Iteration counter end if.

Loop                                        '##############Bottom of method loop.

Line16: '==========================

i = 1
 For jm = 1 To m
  For im = 1 To L
        If CTi(i) Mod 2 = 1 Then
        If hx(i) <= AqBi(i) Then             'Check for dewatering.
          If kitro + kitrtr > 500 Then
           NegHead = 2
           NegHeadNM = nm + 38
           GoTo Line50
          Else
           If ATC = "Confined" Then
            If (Hni(i) - AqBi(i)) > 0.2 * (AqTi(i) - AqBi(i)) Then
              hx(i) = AqBi(i) + 0.2 * (AqTi(i) - AqBi(i))
             Else
              hx(i) = Hni(i)
            End If
           Else 'Unconfined
            If (Hni(i) - AqBi(i)) > 0.2 * (Hnp(jm, im) - AqBi(i)) Then
              hx(i) = AqBi(i) + 0.2 * (Hnp(jm, im) - AqBi(i))
             Else
              hx(i) = Hni(i)
            End If
           End If
           maxerr = 2 * eps
          End If
        End If
        End If
```

```
      i = i + 1
 Next im
Next jm
If maxerr = 2 * eps Then
  subsflag = 77
  GoTo Line15
End If

Line16b: '==========================
For i = 1 To r
    transp(i) = Trans(i)          'Set prior transition code for next cycle.
    TransBp(i) = TransB(i)
Next i

If ThickCon <> "Next Step" Then
ReDim ho(1 To m, 1 To L)
i = 1
jm = 1
 Do While jm < m + 1
 im = 1
  Do While im < L + 1
  If CT(jm, im) <> 4 Then
ho(jm, im) = hofunc(hx(i), AqT(jm, im), AqB(jm, im), UMA(jm, im), ATC, UMS(jm, im), Bed(jm, im), bpp(jm, im), WR(jm, im)) 'Thickness update.
     If AqB(jm, im) + ho(jm, im) > DEC(jm, im) Then
       ho(jm, im) = DEC(jm, im) - AqB(jm, im)
     End If
  End If
   i = i + 1
   im = im + 1
  Loop
  jm = jm + 1
 Loop
 End If

Line17: '==========================

If TransT > 0 Then
  If kitrtr < 1000 Then
     kitrtr = kitrtr + 1
     GoTo Line9
    Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
ElseIf maxerr > eps Then
  If kitro < 1000 Then
     kitro = kitro + 1
     GoTo Line9
    Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
End If
```

```
If GTC = 40 Then
GoTo Line40
End If
'---------------------------------------------------------------------------------------------
'END SIMULTANEOUS SOLUTION by MICCG!!
'
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Recursive portions of solution follow initial solution for time step.
If ThickCon = "Next Step" Then       'ThickCon if.
Else                                 'ThickCon else, "Same Step" update timing option...the recommended setting.
  '##########Begin outside iteration (recursive solution loop) for aquifer thickness.
Line31: '==========================

Do While kitri > 1   '----------------Top of outside iteration loop, continues until convergence on first inner iteration after thickness update.
If kitro < 1000 Then                    'Outside iteration limit check if.
kitro = kitro + 1
  GTC = 40
  GoTo Line9    'Recompute coefficients from new head estimates, from there, repeat simultaneous solution by MICCG.
Line40: '==========================
 Else                                                     'Intervene to end outside iteration if not converging in a reasonable number of cycles.
  NegHead = 3
  NegHeadNM = nm + 38
  GoTo Line50
 End If                                                   'Outside iteration counter end if.
Loop               '----------------Bottom of outside iteration loop.

End If                              'ThickCon end if.
   '##########End outside iteration for aquifer thickness.
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Last of recursion.

Line45: '==========================
 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1
    Hnn(jm, im) = hx(i)                         'Convert next head solution from system domain vector to model domain array.
    Hni(i) = Hno(jm, im)                        'Revert single index vector to step-original head for storage check below.
     If ho(jm, im) = DEC(jm, im) - AqB(jm, im) Then
       If msgc = "" Then
       msgc = "Moist Surface! "
       End If
       flaw = 4
     End If
    If Trans(i) > Transnm(jm, im) Then
     Transnm(jm, im) = Trans(i)
    End If
              If Transpn(i) = 1 Then
               If Trans(i) = 2 Then
                If ic = "Instantaneous" Then
                 GoTo Line46
                End If
               End If
              ElseIf Transpn(i) = 2 Then
               If Trans(i) = 1 Then
                If ic = "Instantaneous" Then   'ic if
Line46: '==========================
```

```
                msgtg = "Status Oscillation (Instantaneous Impulse Arrival). "
              End If                          'ic end if.
             End If
            End If
    If CT(jm, im) = 1 Then
         If ATC = "Confined" Then
           If ITS = "Deep Percolation" Then
             If Hnn(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
              GoTo Line47
             End If
           End If
          End If
        If Transnm(jm, im) > 0 Then
         If Right(IMS, 4) = "Head" Then
          If ic = "Steady" Then
           If msgti = "" Then
            msgti = "Impulse Distribution Through Confinement Transition. "
           End If
          End If
         End If
           If nm > 1 Then
           If Transpnm(jm, im) > 0 Then
           If qs * Qm(nm) > qs * Qm(nm - 1) Then
            If Qc(jm, im) < QCP(jm, im) Then
              msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
            End If
            ElseIf qs * Qm(nm) <= qs * Qm(nm - 1) Then
            If Qc(jm, im) > QCP(jm, im) Then
              msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
            End If
           End If
           End If
         End If
        End If
    End If
    If ATC = "Unconfined" Then
    If CTi(i) <> 4 Then
    If Hnn(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
Line47: '==========================
             msgc = "Inundated Ground! "
             msgti = "Impulse Exceeds Aquifer Capacity. "
             flaw = 3
             NegHead = 6
             NegHeadNM = nm + 38
             GoTo Line50
    End If
    End If
    End If
   If CT(jm, im) <> 4 Then
     If MSTA <> 0 Then
        If ho(jm, im) < hoo(jm, im) * MSTA / 100 Then                                   'Check minimum allowed saturated thickness.
          msgmt = "Min Thickness! "                                                          'Update warning message for minimum allowed
saturated thickness.
        End If                                                                            'End if.
     End If
     If TransB(i) = 1 Then
```

```
       msgt = "Confinement Change. "
       ElseIf TransB(i) = 2 Then
       msgt = "Confinement Change. "
       End If
       If msgt = "" Then
       If Trans(i) = 1 Then
       msgt = "Confinement Change. "
       ElseIf Trans(i) = 2 Then
       msgt = "Confinement Change. "
     End If
     End If
    End If
    If o = nc Then
     QCP(jm, im) = Qc(jm, im)
     Transpnm(jm, im) = Transnm(jm, im)
    End If
    im = im + 1
    i = i + 1
   Loop
   jm = jm + 1
 Loop

'
'OUTPUT  (starts before end of time step loop)
'-----------------------------------------------------------------
RSP(n) = 0                          'Prepare for response flow calculation.....................................................
RSPA(n) = 0
 ReDim Qnno(1 To m, 1 To L)
 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1              'Solved heads allow explicit calculation of flow through streambed.
    If CTP(jm, im) = 2 Then             'Restrictive bed flow by Darcy's law.
      If SB = "Restrictive" Then
        If SIT <> "None" Then
        If hx(i) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
          Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnn(jm, im) - SHD(jm, im)) / bpp(jm, im)    'Full bed flux
term.
         ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
          Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im) - UMS(jm, im)) /
bpp(jm, im)      'Bed flux limited by gradient to bottom of bed.
          msgbg = "Bed Flux Limited by Desaturation. "
        End If
        Else 'SIT
        If hx(i) >= (Bed(jm, im) - bpp(jm, im)) Then
          Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnn(jm, im) - SHD(jm, im)) / bpp(jm, im)    'Full bed flux
term.
         ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im)) Then
          Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im)) / bpp(jm, im)
'Bed flux limited by gradient to bottom of bed.
          msgbg = "Bed Flux Limited by Desaturation. "
        End If
        End If 'SIT
       ElseIf SB = "Permissive" Then
        Qnno(jm, im) = 0
```

```
        If jm - 1 > 0 Then              'Permissive bed flow is cell flow, by Darcy's law, no change in storage.
         If CTP(jm - 1, im) Mod 2 = 1 Then  'Not class 2 or 4 (remainder of division is 1).
          Qnno(jm, im) = Qnno(jm, im) + CC(jm - 1, im) * (Hnn(jm - 1, im) - Hnn(jm, im))
         End If
        End If
        If im - 1 > 0 Then
         If CTP(jm, im - 1) Mod 2 = 1 Then
          Qnno(jm, im) = Qnno(jm, im) + DD(jm, im - 1) * (Hnn(jm, im - 1) - Hnn(jm, im))
         End If
        End If
        If jm + 1 < m + 1 Then
         If CTP(jm + 1, im) Mod 2 = 1 Then
          Qnno(jm, im) = Qnno(jm, im) + AA(jm + 1, im) * (Hnn(jm + 1, im) - Hnn(jm, im))
         End If
        End If
        If im + 1 < L + 1 Then
         If CTP(jm, im + 1) Mod 2 = 1 Then
          Qnno(jm, im) = Qnno(jm, im) + BB(jm, im + 1) * (Hnn(jm, im + 1) - Hnn(jm, im))
         End If
        End If
       End If
     Qnno(jm, im) = qrs * dt * Qnno(jm, im) / UCF  'Flow given correct sign, multiplied by dt to convert to units of volume, converted to acre-feet
if applicable.
    RSP(n) = RSP(n) + Qnno(jm, im) 'Aggregate.
    If zone(jm, im) = 1 Then       'Differentiate.
     RSPA(n) = RSPA(n) + Qnno(jm, im) '.....................................................................................
    End If
   End If
    im = im + 1
    i = i + 1
   Loop
   jm = jm + 1
 Loop

subsflag = 1                        'Independent domain volume budget block.
GoTo Line15                           'Check balance to confirm solution threshold adequacy.
Line48: '==========================
Qtt = qrs * Qtt                     'Volume into aquifer during step.
RSP(n) = (-1) * UCF * RSP(n)        'Volume out.
dS = 0
i = 1
jm = 1
Do While jm < m + 1
  im = 1
  Do While im < L + 1               'Change in storage.
    dS = dS + FuncdV(Trans(i), TransB(i), WR(jm, im), ATC, (Hno(jm, im) + UMAi(i)), (Hnn(jm, im) + UMAi(i)), AqT(jm, im), AqB(jm, im), Ss(jm, im),
Sy(jm, im), SIT, Bed(jm, im), CT(jm, im), bpp(jm, im), SB, dx(im), dy(jm), UMA(jm, im), UMS(jm, im))
    im = im + 1
    i = i + 1
  Loop
  jm = jm + 1
Loop
dS = qs * dS
BD = Abs(Qb - (dS + RSP(n)))  'Budget difference: volume in vs. change in storage plus volume out.
BLT = BLTF(Qb, dS, RSP(n), ic, Qm(nm), UCF, nc)
If BLT > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then 'Minimum scale for evaluation.
```

```
If BD > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then
If 100 * BD / BLT > 0.1 Then 'Difference divided by largest term, converted to percentage, compared to acceptance level...1/10 of 1%.
 If msgc <> "Moist Surface! " Then
   NegHeadNM = nm + 38
   NegHead = 5
   If Abs(IMPSMX) + Abs(IMPSMN) > 0.00001 Then
     msgep = "Budget Gap. "
    Else
     msgep = "No Volume Budget. "
   End If
   GoTo Line50
  End If
 End If
 End If
 End If
RSP(n) = RSP(n) / ((-1) * UCF)
Qtt = Qtt / qrs

   If HTII > 0 Then
    If o = nc Then
     HTS(nm) = -1 * Hnn(HTIJ, HTII)
    End If
   End If
If nm = nh Then
  If o = nc Then
   im = 1
   jm = 1
   Do While im < L + 1             'Stash head to plot.
    Do While jm < m + 1
      Hnnp(jm, im) = -1 * Hnn(jm, im)
     jm = jm + 1
    Loop
    jm = 1
    im = im + 1
   Loop
  End If
 End If

n = n + 1                        'TIME STEP PROGRESSION.
o = o + 1
If o = nc + 1 Then
 o = 1
 nm = nm + 1
End If

im = 1
jm = 1
 Do While im < L + 1
  Do While jm < m + 1
   Hn(jm, im) = Hnn(jm, im)      'UPDATE HEAD.
   Hno(jm, im) = Hn(jm, im)
   jm = jm + 1
  Loop
  jm = 1
  im = im + 1
 Loop
```

```
Loop                              'Time step loop!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                  'END OF TIME STEP COMPUTATION LOOP (continue output section)
Line50: '==========================

 Sheet10.Unprotect Password:=CPC
 Sheet10.Cells(133, 2).Value = "Head Time Series at Node"
 If HTII > 0 Then
   Sheet10.Cells(134, 7).Value = HTII
   Sheet10.Cells(135, 7).Value = HTIJ
    If NegHead > 1 Then
        hrb = NegHeadNM - 38 + 140 - 1
     Else
        hrb = no + 140
    End If
    Sheet10.Range("c140:" & "c" & hrb).Value = WorksheetFunction.Transpose(HTS())        'Output head time series.
   Else
   Sheet10.Cells(135, 3).Value = "To plot an output head time series, please specify, on the Time tab, non-zero node indeces for the desired
location.  Thank you!"
 End If
     Sheet10.Cells(72, 55).Value = ""
     Sheet10.Cells(72, 2).Value = "Head Surface Output"
If nh > 0 Then
  Sheet10.Cells(73, 7).Value = nh
If nh < no + 1 Then
hrb = 76 + m
hrr = crf(L)
Sheet10.Range("c77:" & hrr & hrb).Value = Hnnp()        'Output head to plot.
Else
GoTo Line60
End If
Else
Line60: '==========================
Sheet10.Cells(75, 3).Value = "To plot head, please specify, on the Time tab, a period number for plotting that is greater than zero and less than or
equal to the number of simulation periods.  Thank you!"
End If
  If NegHead > 1 Then
  If NegHeadNM - 38 - 1 < nh Then
  hrb = 76 + m
  hrr = crf(L)
  Sheet10.Range("c77:" & hrr & hrb).ClearContents
  End If
  End If
 Sheet10.Protect Password:=CPC

RSPT = 0
RSPTA = 0
IMPST = 0
n = 1
nm = 1
Do While n < nn + 1                                     'Compute Impulse and Response Outputs and Totals.
 o = 1
 RSPO(nm) = 0
 RSPOA(nm) = 0
 Do While o < nc + 1
  RSPO(nm) = RSPO(nm) + RSP(n)
```

```
  RSPOA(nm) = RSPOA(nm) + RSPA(n)
  o = o + 1
  n = n + 1
 Loop
 IMPST = IMPST + Qm(nm)
 RSPT = RSPT + RSPO(nm)
 RSPTA = RSPTA + RSPOA(nm)
 nm = nm + 1
Loop
  If IMPSMX > 0 Then
   If IMPSMN < 0 Then
    IMPSMX = 999999999.123457
   End If
  End If

If IMPSMX <> 999999999.123457 Then
If IMPST <> 0 Then                    'SECONDARY OUTPUT BLOCK
If SO = "Truncated Response" Then     'Physical residual distributed evenly for single impulse scenario.
 n = 1
 Do While n < no + 1
  SOV(n) = RSPO(n) + (IMPST - RSPT) / no
  n = n + 1
 Loop
End If
If SO = "Cumulative Ratio" Then       'Ratio of cumulative response to impulse through each time step.
 CUMI = 0
 CUMR = 0
 n = 1
 Do While n < no + 1
  CUMI = CUMI + Qm(n)
  CUMR = CUMR + RSPO(n)
  If CUMI <> 0 Then
  SOV(n) = CUMR / CUMI
  Else
  SOV(n) = "-"
  End If
  n = n + 1
 Loop
End If
If SO = "Response Ratio" Then         'Ratio of response for time step to impulse total.
 n = 1
 Do While n < no + 1
  SOV(n) = RSPO(n) / IMPST
  n = n + 1
 Loop
End If
If SO = "Period Ratio" Then         'Ratio of response for time step to impulse for time step.
 n = 1
 Do While n < no + 1
  If Qm(n) <> 0 Then
  SOV(n) = RSPO(n) / Qm(n)
  Else
  SOV(n) = "-"
  End If
  n = n + 1
 Loop
```

```
End If
If SO = "Impact Factor" Then         'Fraction of annual response on symmetrical rolling basis.
 If St = "Annual Pattern" Then
 If NegHead = 1 Then
 IFT = 0
 CUMRn(0) = 0
 n = 1
 Do While n < no + 1
  CUMRn(n) = CUMRn(n - 1) + RSPO(n)
  n = n + 1
 Loop
 If PD = "Days" Then
  tt = 365 * WorksheetFunction.RoundDown(no / 365, 0) - ((2 * 365) - 1)
  n = 1
  Do While n < no + 1
   If n < 365 + 1 Then
    SOV(n) = RSPO(tt + n - 1) / (CUMRn(tt + n - 1 + (365 - 1) / 2) - CUMRn(tt + n - 1 - 1 - (365 - 1) / 2))
    IFT = IFT + SOV(n)
   Else
    SOV(n) = "-"
   End If
   n = n + 1
  Loop
  For n = 1 To 365
   SOV(n) = SOV(n) / IFT
  Next n
 Else
  tt = 12 * WorksheetFunction.RoundDown(no / 12, 0) - ((2 * 12) - 1)
  n = 1
  Do While n < no + 1
   If n < 12 + 1 Then
    SOV(n) = RSPO(tt + n - 1) / (((CUMRn(tt + n - 1 + 5) - CUMRn(tt + n - 1 - 7)) + (CUMRn(tt + n - 1 + 6) - CUMRn(tt + n - 1 - 6))) / 2)
    IFT = IFT + SOV(n)
   Else
    SOV(n) = "-"
   End If
   n = n + 1
  Loop
  For n = 1 To 12
   SOV(n) = SOV(n) / IFT
  Next n
 End If                       'PD end if.
 End If                       'NegHead end if.
 End If                       'ST end if.
End If                        'SO end if.
FCR = RSPT / IMPST
End If                        'IMPST end if.
End If                        'IMPSMX end if.

Sheet11.Unprotect Password:=CPC

Sheet11.Cells(20, 20).Value = CTPflg

hrb = 38 + no
Sheet11.Cells(6, 22).Value = flaw
Sheet11.Range("e29").ClearContents
```

```
If msgim2 = "10" Then                           'Message prep.
msgim2 = "Impulses Net 0! "
End If
If msgim2 = "11" Then
msgim2 = "Impulse Tab 0s! "
hrb = 38
Sheet11.Cells(6, 22).Value = 2
End If
If RSPOFM2 = 2 Then
msgh = msgh & "Bed Level Reset to Stream Head. "
End If
If msgti = "Discrete Impulse Distribution by Head Requires Non-zero Weighting. " Then
hrb = 38
Sheet11.Cells(6, 22).Value = 2
End If
If msgti = "Discrete Impulse Distribution by Head Requires Like Signs Throughout Grid. Distribute by Volume Instead. " Then
hrb = 38
Sheet11.Cells(6, 22).Value = 2
End If
If msgep <> "" Then
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
 If msgep = "Budget Gap. " Then
   Sheet11.Cells(10, 10).Value = epse
  Else
   Sheet11.Cells(10, 10).ClearContents
 End If
Else
Sheet11.Cells(10, 10).ClearContents
End If
If msgpb <> "" Then
msgh = msgpb & msgh
End If
If RSPOFM = 2 Then
msgh = msgh & "Head Equalization. "
ElseIf RSPOFM = 4 Then
msgh = msgh & "Stream Head Equalization. "
End If
If NegHead = 2 Then
msgtt = "Dewatered Cell(s)! "
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
msgres = ""
End If
Sheet11.Cells(10, 11).Value = 11
If NegHead = 3 Then
msgtt = "Iteration Limit Exceeded! "
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
Sheet11.Cells(10, 11).Value = epse
msgres = ""
End If
If NegHead = 6 Then
hrb = NegHeadNM - 1
End If
If msgres <> "" Then
```

```
Sheet11.Cells(10, 11).Value = epse
End If
If NegHead = 4 Then
msgtt = "Impulse Distribution Iteration Limit Exceeded! "
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
End If

If hrb > 38 Then
Sheet11.Range("e39:" & "e" & hrb).Value = WorksheetFunction.Transpose(RSPOA())     'Output response time series, Zone 1, then total, then secondary
output.
Sheet11.Range("f39:" & "f" & hrb).Value = WorksheetFunction.Transpose(RSPO())
 If IMPSMX <> 999999999.123457 Then
   If hrb = 38 + no Then
    If RSPOFM = 1 Then
      Sheet11.Range("g39:" & "g" & hrb).Value = WorksheetFunction.Transpose(SOV())
      Sheet11.Cells(29, 5).Value = FCR                                          'Output final computed cumulative ratio of response to impulse.
    End If
   End If
  Else
  msgh = "Impulse Alternation. "
 End If
End If

Else                                      'Thickness input else.
Sheet11.Unprotect Password:=CPC
   msgh = msghfun(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1))
   If msgh22(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1)) = 2 Then
   Sheet11.Cells(6, 22).Value = 2
   End If
End If                                    'End thickness input if.
   If TipW(3) = 1 Then
   msgtp = "Aquifer Top at or above Initial Piezometric Surface. "
   End If

If msgtt <> "" Then
  If msg = "None!" Then                 'Final message prep.
    msg = msgtt
    Else
    msg = msgtt & msg
  End If
End If
If msgc <> "" Then
  If msg = "None!" Then
    msg = msgc
    Else
    msg = msg & msgc
  End If
End If
If msgt <> "" Then
  If msg = "None!" Then
    msg = msgt
    Else
    msg = msg & msgt
  End If
End If
```

```
If msgh <> "" Then
  If msg = "None!" Then
    msg = msgh
    Else
    msg = msgh & msg
  End If
End If
If msgim <> "" Then
  If msg = "None!" Then
    msg = msgim
    Else
    msg = msg & msgim
  End If
End If
If msgep <> "" Then
  If msg = "None!" Then
    msg = msgep
    Else
    msg = msg & msgep
  End If
End If

If Len(msg & msgim2 & msgbg & msgti & msgmt & msgtg & msgu & msgumat & msgclt & msgwp & msgres & msgctp & msgrza & msgtt) > 18 Then
Sheet11.Cells(30, 5).Value = "Posted at J80."
Sheet11.Range("j80:j158").ClearContents
Sheet11.Cells(80, 10).Value = "Calculation Notes"
i = 0
If msgc <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgc  'Output message.
End If
If msgtp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtp
End If
If msgh <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgh
End If
If msgim <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgim
End If
If msgim2 <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgim2
End If
If msgt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgt
End If
If msgti <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgti
End If
If msgtg <> "" Then
```

```
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtg
End If
If msgbg <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgbg
End If
If msgmt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgmt
End If
If msgtt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtt
End If
If msgep <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgep
End If
If msgu <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgu
End If
If msgumat <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgumat
End If
If msgclt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgclt
End If
If msgwp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgwp
End If
If msgres <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgres
End If
If msgctp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgctp
End If
If msgrza <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgrza
End If
Else
Sheet11.Range("j80:j158").ClearContents
Sheet11.Cells(30, 5).Value = msg & msgim2 & msgbg & msgti & msgtg & msgmt & msgtt & msgu & msgumat & msgclt & msgwp & msgres & msgctp & msgrza
End If

Sheet11.Protect Password:=CPC

End Sub
'*************************************************************************************************************************************************
```

```
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************
Sub IMTD()                          'Solution of implicit representation Adapted For 2-layer delayed-yield aquifer.

Dim dt As Double                      'Dimension calculation time step.
Dim dx() As Double, dy() As Double    'Dimension increments.
Dim k() As Double                     'Dimension hydraulic conductivity.
Dim Sy() As Double, Ss() As Double    'Dimension storage properties.
Dim hoo() As Double, ho() As Double   'Dimension initial saturated thickness, transmitting thickness.
Dim AqT() As Double, AqB() As Double  'Dimension aquifer top, bottom.
Dim Hn() As Double, Hno() As Double   'Dimension initial head, original for step.
Dim Hni() As Double                   'Dimension single index initial head.
Dim Hnp() As Double, Hnpat() As Double 'Dimension permanent initial heads.
Dim Qd() As Double, Qm() As Double    'Dimension net flow impulse distribution, impulse time series.
Dim Qdmx As Double, Qdmn As Double    'Dimension impulse distribution max and min.
Dim Qdmna As Double                   'Dimension impulse min absolute value.
Dim Qc() As Double, QCP() As Double   'Dimension impulse for calculation.
Dim AA() As Double, BB() As Double    'Dimension flow coefficients.
Dim CC() As Double, DD() As Double    'Dimension flow coefficients.
Dim EE() As Double, b() As Double     'Dimension volume coefficients.
Dim SE() As Double, SEC() As Double   'Dimension matrix to hold system of equations, redimensioned below with element limits.
Dim Hnn() As Double, Hnnp() As Double 'Dimension next time step differential head, head to plot.
Dim Hnnpat() As Double                'Dimension head to plot.
Dim Qnno() As Double                  'Dimension next time step flow output, to be determined for response cells, converting calculated head by
Storage Coefficient, etc.
Dim RSP() As Double, RSPA() As Double 'Dimension Response, Zone Response.
Dim RSPO() As Double, RSPT As Double  'Dimension Response Output, Response Total.
Dim RSPOA() As Double, RSPTA As Double 'Dimension Response Output, Response Total, both for Zone.
Dim i As Integer, im As Integer       'Dimension matrix row index, model column index.
Dim jm As Integer, L As Integer       'Dimension model row index, number of model columns.
Dim m As Integer, r As Integer        'Dimension number of model columns, matrix rows.
Dim ITS As String, IMS As String      'Dimension Impulse Type Specification, Impulse Magnitude Specification.
Dim CT() As Integer, CTi() As Integer 'Dimension Cell Type.
Dim IA As Double, IAP As Double       'Dimension Input Area effects.
Dim qs As Double, qrs As Double       'Dimension Flow Sign, Flow Response Sign.
Dim PD As String, ic As String        'Dimension Period Denomination, Impulse Character.
Dim u As Long, n As Long, nc As Long  'Dimension units per period, number of periods, number of sub-periods.
Dim nn As Long, nh As Long            'Dimension total number of steps, period number to plot head.
Dim no As Long, nm As Long, o As Long 'Dimension original number of periods, number of of period in undivided increments, period counter.
Dim deno As Long                      'Dimension denomitor for characteristic impulse calculation.
Dim ATC As String                     'Dimension Aquifer Thickness Character.
Dim UCF As Double                     'Dimension Unit Conversion Factor.
Dim zone() As String                  'Dimension response Zone.
Dim msg As String, msgc As String     'Dimension warning messages.
Dim msgt As String, msgti As String   'Dimension warning messages.
Dim msgbg As String                   'Dimension warning messages.
Dim msgtt As String, msgh As String   'Dimension warning messages.
Dim msgtp As String, msgmt As String  'Dimension warning messages.
Dim msgim As String, msgim2 As String 'Dimension warning messages.
Dim msgpb As String, msgep As String  'Dimension warning messages.
Dim MSTA As Double                    'Dimension minimum saturated thickness allowed.
Dim DEC() As Double                   'Dimension depth of earth cover.
Dim IMPST As Double                   'Dimension impulse total.
Dim CPC As Variant                    'Dimension some dummy text here.
```

```
CPC = "*******"                       'Dimension nothing to see here again.
Dim RSPOFM As Integer                 'Dimension response output flag to catch initial head imbalance.
Dim RSPOFM2 As Integer                'Dimension stream head imbalance flag.
Dim count2 As Double, count13 As Double 'Dimension storage and area weighted counters for cell types.
Dim headsum2 As Double                'Dimension initial storage and area weighted head sums and averages by cell type, stream.
Dim headave2 As Double                'Dimension initial averages and sums.
Dim headsum2s As Double               'Dimension initial averages and sums.
Dim headsum13 As Double               'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headave13 As Double               'Dimension initial averages and sums.
Dim SB As String, SIT As String       'Dimesion relative permeability class of streambed, streambed incision type.
Dim WR() As Double                    'Dimension streambed width ratio.
Dim kpp() As Double, bpp() As Double  'Dimension discrete streambed conductivity, thickness.
Dim bppi() As Double, widp() As Double 'Dimension discrete streambed single index thickness, dummy width.
Dim widpp() As Double, lenp() As Double 'Dimension discrete streambed width, length.
Dim hrb As Variant, hrr As Variant    'Dimension output cell range extents.
Dim hrr2 As Variant                   'Dimension cell range extent.
Dim epse As Integer                   'Dimension iteration closure threshold exponent.
Dim IMPSMX As Double, IMPSMN As Double 'Dimension impulse max and min.
Dim NegHead As Integer                'Dimension negative head flag.
Dim NegHeadNM As Integer              'dimension error period.
Dim AqBi() As Double, AqTi() As Double 'Dimension single-index variables for bottom and top.
Dim AvTp As String                    'Dimension transmission characteristic average type.
Dim Bed() As Double, BEDi() As Double 'Dimension discrete bed elevation.
Dim SHD() As Double                   'Dimension stream head.
Dim TipW() As Integer                 'Dimension thickness input parameter watch variable.
Dim hx() As Double                    'Dimension successive approximation head vector.
Dim eps As Double                     'Dimension convergence threshold for MICCG.
Dim Ax() As Double, rr() As Double    'Dimension MICCG solver variables.
Dim vv() As Double, zz() As Double    'Dimension MICCG solver variables.
Dim UT() As Double, DU() As Double    'Dimension MICCG solver preconditioner matrices.
Dim UM() As Double                    'Dimension MICCG solver preconditioner matrix.
Dim pp() As Double, Ap() As Double    'Dimension MICCG solver variables.
Dim nip As Double, dip As Double      'Dimension MICCG solver variables.
Dim alpha As Double, beta As Double   'Dimension MICCG solver variables.
Dim xp() As Double, maxerr As Double  'Dimension MICCG solver variables.
Dim maxerrp As Double                 'Dimension prior max error.
Dim kitr As Integer, kitri As Integer 'Dimension iteration trackers.
Dim kitrtr As Integer, kitro As Integer 'Dimension iteration trackers.
Dim GTC As Integer, GTCR As Integer   'Dimension GoTo codes.
Dim St As String                      'Dimension schedule type.
Dim Trans() As Integer                'Dimension pressurization transition flags.
Dim TransB() As Integer               'Dimension bed transition flag.
Dim transp() As Integer               'Dimension prior pressurization transition flags.
Dim TransBp() As Integer              'Dimension prior bed transition flag.
Dim TransT As Integer                 'Dimension pressurization transition aggregate flag.
Dim Transnm() As Integer              'Dimension period transition.
Dim Transpnm() As Integer             'Dimension prior period transition.
Dim IGTC As Integer                   'Dimension impulse distribution GoTo code.
Dim dH As Double, Qt As Double        'Dimension Newton solver variables for impulse distribution.
Dim Qtt As Double, Qf As Double       'Dimension impulse distribution variables.
Dim ObF As Double, ObFF As Double     'Dimension Newton solver variables for impulse distribution.
Dim dFdu As Double, Qttp As Double    'Dimension impulse distribution variable, prior Qtt.
Dim dS As Double                      'Dimension change in storage for global budget.
Dim BD As Double, BLT As Double       'Dimension budget difference, budget largest term.
Dim PermFact() As Double              'Dimension permissive bed coefficient factor.
Dim tttb As Double                    'Dimension value holder to prevent duplication.
```

```
Dim HTS() As Double, HTS2() As Double 'Dimension head time series output.
Dim HTII As Integer, HTIJ As Integer  'Dimension head time series location im index, jm index.
Dim kv() As Double, kh() As Double    'Dimension vertical, horizontal conductivity of aquitard.
Dim kvq() As Double                   'Dimension vertical conductivity of aquifer.
Dim Sigma() As Double, Sigs() As Double 'Dimension specific yield, specific storage of aquitard.
Dim Hnat() As Double, Hnoat() As Double 'Dimension initial head of aquitard.
Dim Hnnat() As Double, hoat() As Double 'Dimension solved head, saturated thickness of aquitard.
Dim Hniat() As Double                 'Dimension single index intial aquitard head.
Dim AAt() As Double, BBt() As Double  'Dimension primary coefficients for aquitard equations.
Dim CCt() As Double, DDt() As Double  'Dimension aquitard flow coefficients.
Dim EET() As Double, bbtt() As Double 'Dimension aquitard coefficients.
Dim GG() As Double, GGT() As Double   'Dimension vertical flow coefficients for flow between layers.
Dim msgat As String, msgres As String 'Dimension aquitard messages.
Dim WRAT() As Double                  'Dimension aquitard width ratio for stream cells.
Dim PermFlag() As Integer             'Dimension permissivefactor flag, residual order.
Dim resord As Integer                 'Dimension residual order.
Dim maxdelta As Double                'Dimension maximum change in head estimates.
Dim LK As Integer                     'Dimension leakance flag.
Dim UMS() As Double, UMA() As Double  'Dimension Ultimate Matric Suction below streambed, aquifer top.
Dim UMAT() As Double                  'Dimension Critical Matric Suction in aquitard, previous impulse.
Dim UMAi() As Double, UMATi() As Double 'Dimension single index variables.
Dim UMSi() As Double                  'Dimension single index suction beneath stream bed.
Dim subsflag As Integer               'Dimension substitution routine flag.
Dim Transpn() As Integer              'Dimension previous step transition.
Dim msgtg As String                   'Dimension g instability message.
Dim kvtb() As Double                  'Dimension average vertical conductivity.
Dim flaw As Integer                   'Dimension flaw flag.
Dim AqTa As Double, AqtB As Double    'Dimension aquifer top place holders.
Dim AqTc As Double, AqTd As Double    'Dimension aquifer top place holders.
Dim Lons() As Integer                  'Dimension long side indicator.
Dim Lon1 As Integer, Lon2 As Integer  'Dimension long side rotation trackers.
Dim msgu As String, msgumat As String 'Dimension matric suction messages.
Dim xpo() As Double                   'Dimension prior outside iteration hx estimates.
Dim cset() As Double, csbt() As Double 'Dimension channel side e and b coefficient, aquitard.
Dim BedMat() As Integer                'Dimension channel bed head condition flag.
Dim TransC() As Long                   'Dimension pressurization oscillation counter.
Dim Gflag() As Integer                 'Dimension pressurization oscillation flag.
Dim GCT() As Integer, GCA() As Integer 'Dimension total and above counters for G flag.
Dim qnoc() As Double                   'Dimension channel side e and b coefficient, aquifer.
Dim msgclt As String, msgwp As String  'Dimension Complete override, stream width messages.
Dim rpc() As Integer                   'Dimension repressurization counter.
Dim bppb() As Double                   'Dimension dummy variable for bank thickness.
Dim Dewflag() As Integer               'Dimension aquitard dewatering flag.
Dim dvtc() As Double                   'Dimension volume increment placeholder for depressurization.
Dim CAS() As Double                   'Dimension cell area times nominal storage.
Dim vfhd As String, vfch As String    'Dimension vertical flow head dissipation options.
Dim Knocker As String                 'Dimension wet cell face seepage option.
Dim trigger As Integer                'Dimension aquitard head adjustment iteration trigger.
Dim GTCRT As Integer                  'Dimension trigger goto code.
Dim KOC As Integer                    'Dimension Kickout Code.
Dim msgsep As String                  'Dimension head separation message.
Dim ATDF As Integer                   'Dimension aquitard head surface dewatering flag.
Dim delta As Double                   'Dimension MICCG upper preconditioner matrix diagonal scale factor to prevent negatives.
Dim PFSflag As Integer                'Dimension PermFact scale flag.
Dim MaxTerm As Double                 'Dimension maximum augmented matrix entry value for aquifer rows.
Dim MaxTermT As Double                'Dimension maximum augmented matrix entry value for aquitard rows.
```

```
Dim CTP() As Integer                  'Dimension neighboring stream cell type for complete permissive bed cases.
Dim BedCP() As Double                 'Dimension neighboring stream cell Bed elevation for complete permissive bed cases.
Dim CTPflg As Integer                 'Dimension CTP flag for cases with relatively small channel dimensions.
Dim msgctp As String                  'Dimension CTP message.
Dim SEB As String                     'Dimension stream end bank option.
Dim msgrza As String                  'Dimension response zone ambiguity message.
Dim zone1 As Integer                  'Dimenson single response zone indicator.
Dim Qb As Double                      'Dimension domain step impulse for budget block.

Line0:   '==========================

ATDF = 0

L = Sheet3.Cells(12, 8).Value         'Load number of cells on one side of model.
m = Sheet3.Cells(15, 8).Value         'Load number of cells on other side of model.
r = L * m * 2                         'Number of rows in equation matrix.

no = Sheet11.Cells(13, 5).Value       'Load number of periods.
nc = Sheet11.Cells(16, 5).Value       'Load number of sub-period steps.
nn = no * nc                          'Set number of total steps.

ReDim dx(1 To L), dy(1 To m)           'Set custom array dimensions.
ReDim k(1 To m, 1 To L)
ReDim hoo(1 To m, 1 To L), ho(1 To m, 1 To L)
ReDim Hno(1 To m, 1 To L), Hn(1 To m, 1 To L)
ReDim Hnp(1 To m, 1 To L), Hnpat(1 To m, 1 To L)
ReDim Qd(1 To m, 1 To L)
ReDim Qm(1 To no)
ReDim Hnn(1 To m, 1 To L), Hnnp(1 To m, 1 To L), Hnnpat(1 To m, 1 To L)
ReDim Qnno(1 To m, 1 To L)
ReDim RSP(1 To nn), RSPA(1 To nn)
ReDim RSPO(1 To no)
ReDim RSPOA(1 To no)
ReDim CT(1 To m, 1 To L)
ReDim zone(1 To m, 1 To L)
ReDim DEC(1 To m, 1 To L)
ReDim Sy(1 To m, 1 To L), Ss(1 To m, 1 To L)
ReDim kpp(1 To m, 1 To L), bpp(1 To m, 1 To L), bppi(1 To r / 2)
ReDim widp(1 To m, 1 To L), widpp(1 To m, 1 To L)
ReDim lenp(1 To m, 1 To L)
ReDim AqBi(1 To r / 2), AqTi(1 To r / 2)
ReDim Bed(1 To m, 1 To L), BEDi(1 To r / 2), SHD(1 To m, 1 To L)
ReDim AqT(1 To m, 1 To L), AqB(1 To m, 1 To L)
ReDim WR(1 To m, 1 To L)
ReDim TipW(1 To 5)
ReDim Transpnm(1 To m, 1 To L)
ReDim Hni(r / 2 + 1 To r)
ReDim CTi(1 To r / 2)
ReDim Qc(1 To m, 1 To L), QCP(1 To m, 1 To L)
ReDim HTS(0 To no), HTS2(0 To no)
ReDim kv(1 To m, 1 To L), kh(1 To m, 1 To L), kvq(1 To m, 1 To L)
ReDim Sigma(1 To m, 1 To L), Sigs(1 To m, 1 To L)
ReDim Hnat(1 To m, 1 To L), Hnoat(1 To m, 1 To L)
ReDim Hnnat(1 To m, 1 To L), hoat(1 To m, 1 To L)
ReDim Hniat(1 To r / 2)
ReDim WRAT(1 To m, 1 To L)
```

```
ReDim Transpn(1 To r)
ReDim UMS(1 To m, 1 To L), UMA(1 To m, 1 To L)
ReDim UMAT(1 To m, 1 To L)
ReDim UMAi(r / 2 + 1 To r), UMATi(1 To r / 2), UMSi(1 To r / 2)
ReDim PermFact(1 To m, 1 To L, 1 To 2), Lons(1 To m, 1 To L)
ReDim bppb(1 To m, 1 To L)
ReDim CTP(1 To m, 1 To L)
ReDim BedCP(1 To m, 1 To L)

'
'INPUT
'---------------------------------------------------------------------
RSPOFM = 1                          'Initialize flags all clear.
RSPOFM2 = 1

Call pload(St, SB, SIT, ITS, IMS, ic, ATC, AvTp, HTII, HTIJ, flaw)
vfch = WorksheetFunction.Proper(Sheet6.Cells(8, 37).Value)
vfhd = WorksheetFunction.Proper(Sheet6.Cells(10, 37).Value)
Knocker = WorksheetFunction.Proper(Sheet6.Cells(9, 30).Value)
SEB = WorksheetFunction.Proper(Sheet12.Cells(12, 14).Value)     'Load Stream End Bank Option.

epse = Sheet11.Cells(10, 5).Value    'Load closure threshold exponent.
eps = 1 * 10 ^ (-epse)                    'Set convergence threshold value.

msg = Sheet11.Cells(30, 5).Value      'Load warning message, if any.

If ITS = "Well Pumping" Then          'Set flow signs for practical conventions.
 qs = 1
 qrs = -1
 Else
 qs = -1
 qrs = 1
End If
If WorksheetFunction.Proper(Sheet3.Cells(8, 8).Value) = "Acre-Feet" Then   'Set Unit Conversion Factor.
 UCF = 43560
 Else
 UCF = 1
End If
If WorksheetFunction.Proper(Sheet4.Cells(4, 4).Value) = "A" Then  'Load Min Sat Thick Allow.
 MSTA = Sheet4.Cells(15, 5).Value
End If

PD = WorksheetFunction.Proper(Sheet11.Cells(14, 5).Value) 'Load time unit denomination.
u = Sheet11.Cells(15, 5).Value       'Load units per period.
nh = Sheet11.Cells(23, 5).Value      'Load period to plot heads.
If PD = "Days" Then                  'Set time step.
 dt = u / nc
 ElseIf PD = "Months" Then
 dt = u * 365.25 / 12 / nc
 ElseIf PD = "Hours" Then
 dt = u / nc / 24
End If
If ic = "Steady" Then
 deno = nc
 Else
 deno = 1
```

```
End If

For im = 1 To L
For jm = 1 To m
   CT(jm, im) = Sheet3.Cells(26 + jm, 3 + im).Value
Next jm
Next im
                                        'Load initial parameter values.
Call Load(RSPOFM2, SB, SIT, IMS, im, L, jm, m, msgwp, _
          dx(), dy(), CT(), widp(), widpp(), lenp(), Lons(), AqT(), _
          kv(), kh(), kvq(), LK, Sigma(), Sigs(), AqB(), DEC(), _
          k(), zone(), Sy(), Ss(), Hn(), Hno(), Hnp(), Hnat(), _
          Hnoat(), Hnpat(), UMA(), UMAT(), kpp(), bpp(), _
          SHD(), Bed(), msgpb, msgh, TipW(), UMS(), WR(), _
          WRAT(), Qd(), Qdmx, Qdmn, Qdmna, headsum2, count2, _
          headsum2s, headsum13, count13, msgu, Lon1, Lon2, ATC)

                                        'Preliminary processing of parameters.
Call Prelim(jm, m, im, L, CT(), Bed(), bpp(), _
            AqT(), DEC(), TipW(), Hn(), Hnat(), AqB(), AqBi(), _
            AqTi(), BEDi(), CTi(), bppi(), UMAi(), UMATi(), UMSi(), _
            ho(), hoo(), UMA(), UMAT(), UMS(), ATC, r, PermFact(), widp(), _
            widpp(), SB, SIT, msgsep, flaw, WR())

Call CTPload(CTPflg, m, L, CTP(), CT(), SB, SIT, _
             Lons(), SEB, flaw, msgctp, widpp(), widp(), _
             SHD(), AqT(), Bed(), BedCP(), zone(), msgrza, zone1)

For nm = 1 To no
 Qm(nm) = Sheet11.Cells(38 + nm, 4).Value  'Load impulse magnitude time series.
 If nm = 1 Then
 IMPSMX = Qm(1)
 IMPSMN = Qm(1)
 End If
 If IMPSMX < Qm(nm) Then
  IMPSMX = Qm(nm)
 End If
 If IMPSMN > Qm(nm) Then
  IMPSMN = Qm(nm)
 End If
Next nm

 If HTII > 0 Then
         HTS(0) = -1 * Hn(HTIJ, HTII)
         HTS2(0) = -1 * Hnat(HTIJ, HTII)
 End If

 If Qdmx > 0 Then
  If Qdmn < 0 Then
   msgim = "Mixed Impulse. "
  End If
 End If

If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
headave2 = headsum2 / count2
headave13 = headsum13 / count13
```

```
 If Abs(headave2 - headave13) > 0.000001 Then            'Check for initial head balance.
  msgh = msgh & "Head Equalization. "
  RSPOFM = 2
 End If
End If
If Abs(headsum2s) > 0.000001 Then
 msgh = msgh & "Stream Head Equalization. "
 RSPOFM = 4
End If

If TipW(1) + TipW(2) + TipW(4) + TipW(5) = 0 Then                                'Condition routine on valid thickness input, end if near end of
subroutine.

'
'TIME STEP COMPUTATION LOOP
'-----------------------------------------------------------------------
NegHead = 1

  n = 1                              'Initialize time step counter.
  o = 1
  nm = 1

Do While n < nn + 1                'Loop for time steps!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

GTC = 0                             'Initialize codes for each time step.
kitro = 0
kitrtr = 0
TransT = 0
delta = 0
PFSflag = 0
MaxTerm = 0
MaxTermT = 0

If n > 1 Then
   Qttp = Qtt
   For i = 1 To r
    Transpn(i) = Trans(i)
   Next i
End If
ReDim xpo(1 To r)
ReDim rpc(1 To m, 1 To L)
ReDim AvTpT(1 To m, 1 To L, 1 To 4)
ReDim AvTpA(1 To m, 1 To L, 1 To 4)
ReDim Trans(1 To r), TransB(1 To r)
ReDim transp(1 To r), TransBp(1 To r)
ReDim TransC(1 To r), Dewflag(1 To r / 2)
ReDim Gflag(1 To m, 1 To L)
ReDim GCT(1 To m, 1 To L)
ReDim GCA(1 To m, 1 To L)
ReDim CAS(1 To r / 2)
ReDim kvtb(1 To r / 2)
ReDim dvtc(1 To r / 2)
ReDim Preserve Hn(1 To m, 1 To L)
ReDim Preserve Hnat(1 To m, 1 To L)
If o = 1 Then
 ReDim Transnm(1 To m, 1 To L)
```

```
End If

'IMPULSE DISTRIBUTION BLOCK.++++++++++++++
Call IDB(o, ic, Qc(), dH, Qtt, Qm(), Qf, IA, _
        IAP, jm, im, m, L, i, CT(), IMS, _
        Qd(), ITS, dx(), dy(), Sigma(), ATC, SIT, WR(), _
        Hn(), AqT(), Bed(), Ss(), Sy(), AqB(), bpp(), SB, _
        UMA(), UMS(), GTC, msgim, msgti, UCF, qs, Qdmna, _
        msgim2, Hnat(), msgc, NegHead, NegHeadNM, IGTC, ObF, _
        Qt, ObFF, dFdu, msgtt, subsflag, DEC(), nm, nc, dt, flaw, Qb)
If subsflag = 8 Then
 subsflag = 0
 GoTo Line8
 ElseIf subsflag = 50 Then
 subsflag = 0
 GoTo Line50
End If

Line8: '==========================

ReDim hx(1 To r)
i = 1                                          'Re-initialize head vectors.
 For jm = 1 To m
  For im = 1 To L
Hni(i + r / 2) = Hn(jm, im)                                    'Fill vectors.
Hniat(i) = Hnat(jm, im)
hx(i + r / 2) = Hn(jm, im)
hx(i) = Hnat(jm, im)
xpo(i) = hx(i)
xpo(i + r / 2) = hx(i + r / 2)
If CT(jm, im) <> 4 Then
  If Hn(jm, im) <= AqB(jm, im) Then                          'Check that head not beneath aquifer.
Line8b: '==========================
    NegHead = 2
    NegHeadNM = nm + 38
    GoTo Line50
  End If
  If Hnat(jm, im) <= AqB(jm, im) Then
  GoTo Line8b
  End If
End If
    CAS(i) = dx(im) * dy(jm) * SfuncAT(CT(jm, im), SIT, WRAT(jm, im), (AqT(jm, im) + 0.01), AqT(jm, im), Bed(jm, im), Sigs(jm, im), Sigma(jm, im),
bpp(jm, im), SB, UMS(jm, im), UMAT(jm, im))
    kvtb(i) = fkvtb(SIT, kv(jm, im), Bed(jm, im), bpp(jm, im), AqT(jm, im), kvq(jm, im), LK, _
                    dx(im), dy(jm), CT(jm, im), widpp(jm, im), lenp(jm, im), widp(jm, im), SB)
    i = i + 1
 Next im
Next jm
subsflag = 1
GoTo Line17                             'Thickness update.

Line9: '==========================      'Line label for point of return, if pressurization transition in first set of iterations, and for outside
iteration.
'
'FILL PRIMARY COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------------------------
```

```
ReDim AA(1 To m, 1 To L), BB(1 To m, 1 To L), CC(1 To m, 1 To L)  'Redimension A, B, C to reset all to empty.
ReDim DD(1 To m, 1 To L), EE(1 To m, 1 To L), b(1 To m, 1 To L)   'Redimension D, E, b to reset all to empty.
ReDim AAt(1 To m, 1 To L), BBt(1 To m, 1 To L), CCt(1 To m, 1 To L)
ReDim DDt(1 To m, 1 To L), EET(1 To m, 1 To L), bbtt(1 To m, 1 To L)
ReDim GG(1 To m, 1 To L), GGT(1 To m, 1 To L)
ReDim Preserve CT(1 To m, 1 To L), xpo(1 To r)
ReDim BedMat(1 To m, 1 To L)
ReDim PermFlag(1 To m, 1 To L)
ReDim qnoc(1 To m, 1 To L, 1 To 4)
 i = 1
 For jm = 1 To m
  For im = 1 To L                      'Calculate coefficient values.
   If CT(jm, im) <> 4 Then
    dvtc(i) = FuncdVat(Trans(i), TransB(i), WRAT(jm, im), ATC, (Hnat(jm, im) + UMAT(jm, im)), (AqT(jm, im) - UMA(jm, im) + UMAT(jm, im)), AqT(jm,
im), Sigs(jm, im), Sigma(jm, im), SIT, _
                        Bed(jm, im), CT(jm, im), bpp(jm, im), SB, dx(im), dy(jm), UMAT(jm, im), UMS(jm, im), UMA(jm, im))
    If jm - 1 > 0 Then
     If CT(jm - 1, im) < 4 Then
      AA(jm, im) = khbar(SIT, AvTp, k(jm - 1, im), ho(jm - 1, im), dy(jm - 1), WR(jm - 1, im), hx(r / 2 + i - L), UMS(jm - 1, im), Bed(jm - 1, im),
bpp(jm - 1, im), AqB(jm - 1, im), _
                         k(jm, im), ho(jm, im), dy(jm), WR(jm, im), hx(r / 2 + i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dx(im), UMA(jm
- 1, im), UMA(jm, im), ATC, _
                         AqT(jm - 1, im), AqT(jm, im), Lons(jm - 1, im), Lons(jm, im), CT(jm - 1, im), CT(jm, im), 2, SB)
      AAt(jm, im) = khbarat(SIT, SB, AvTp, kh(jm - 1, im), hoat(jm - 1, im), dy(jm - 1), Lons(jm - 1, im), WRAT(jm - 1, im), Bed(jm - 1, im), _
                            bpp(jm - 1, im), AqT(jm - 1, im), hx(i - L), UMS(jm - 1, im), kh(jm, im), hoat(jm, im), dy(jm), Lons(jm, im), WRAT(jm,
im), Bed(jm, im), _
                            bpp(jm, im), AqT(jm, im), hx(i), UMS(jm, im), 2, dx(im), UMAT(jm - 1, im), UMAT(jm, im), CT(jm - 1, im), CT(jm, im),
DEC(jm - 1, im), DEC(jm, im), _
                            rpc, AqB(jm - 1, im), AqB(jm, im), hx(i - L + r / 2), hx(i + r / 2), Hn(jm - 1, im), Hn(jm, im), UMA(jm, im))
                qnoc(jm, im, 1) = QnoCside(SIT, AvTp, AqT(jm, im), AqT(jm - 1, im), hx(i - L), hoat(jm - 1, im), UMAT(jm, im), kh(jm, im), _
                                           kh(jm - 1, im), dx(im), dy(jm), dy(jm - 1), WRAT(jm, im), WRAT(jm - 1, im), Bed(jm, im), _
                                           Bed(jm - 1, im), bpp(jm, im), bpp(jm - 1, im), Lons(jm, im), Lons(jm - 1, im), 2, _
                                           hoat(jm, im), SHD(jm - 1, im), SHD(jm, im), UMS(jm - 1, im), UMS(jm, im), DEC(jm, im), SB, _
                                           UMA(jm, im), UMA(jm - 1, im), hx(i), UMAT(jm - 1, im), Knocker, CT(jm - 1, im), CT(jm, im), DEC(jm - 1,
im), DEC(jm, im), _
                                           rpc, AqB(jm - 1, im), AqB(jm, im), hx(i - L + r / 2), hx(i + r / 2), Hn(jm - 1, im), Hn(jm, im))
                  b(jm, im) = b(jm, im) - qnoc(jm, im, 1)
                  bbtt(jm - 1, im) = bbtt(jm - 1, im) + qnoc(jm, im, 1)
    End If
   End If
   If im - 1 > 0 Then
    If CT(jm, im - 1) < 4 Then
      BB(jm, im) = khbar(SIT, AvTp, k(jm, im - 1), ho(jm, im - 1), dx(im - 1), WR(jm, im - 1), hx(r / 2 + i - 1), UMS(jm, im - 1), Bed(jm, im - 1),
bpp(jm, im - 1), AqB(jm, im - 1), _
                         k(jm, im), ho(jm, im), dx(im), WR(jm, im), hx(r / 2 + i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dy(jm),
UMA(jm, im - 1), UMA(jm, im), ATC, _
                         AqT(jm, im - 1), AqT(jm, im), Lons(jm, im - 1), Lons(jm, im), CT(jm, im - 1), CT(jm, im), 1, SB)
      BBt(jm, im) = khbarat(SIT, SB, AvTp, kh(jm, im - 1), hoat(jm, im - 1), dx(im - 1), Lons(jm, im - 1), WRAT(jm, im - 1), Bed(jm, im - 1), _
                            bpp(jm, im - 1), AqT(jm, im - 1), hx(i - 1), UMS(jm, im - 1), kh(jm, im), hoat(jm, im), dx(im), Lons(jm, im), WRAT(jm,
im), Bed(jm, im), _
                            bpp(jm, im), AqT(jm, im), hx(i), UMS(jm, im), 1, dy(jm), UMAT(jm, im - 1), UMAT(jm, im), CT(jm, im - 1), CT(jm, im),
DEC(jm, im - 1), DEC(jm, im), _
                            rpc, AqB(jm, im - 1), AqB(jm, im), hx(i - 1 + r / 2), hx(i + r / 2), Hn(jm, im - 1), Hn(jm, im), UMA(jm, im))
                qnoc(jm, im, 2) = QnoCside(SIT, AvTp, AqT(jm, im), AqT(jm, im - 1), hx(i - 1), hoat(jm, im - 1), UMAT(jm, im), kh(jm, im), _
                                           kh(jm, im - 1), dy(jm), dx(im), dx(im - 1), WRAT(jm, im), WRAT(jm, im - 1), Bed(jm, im), _
                                           Bed(jm, im - 1), bpp(jm, im), bpp(jm, im - 1), Lons(jm, im), Lons(jm, im - 1), 1, _
```

```
                                        hoat(jm, im), SHD(jm, im - 1), SHD(jm, im), UMS(jm, im - 1), UMS(jm, im), DEC(jm, im), SB, _
                                        UMA(jm, im), UMA(jm, im - 1), hx(i), UMAT(jm, im - 1), Knocker, CT(jm, im - 1), CT(jm, im), DEC(jm, im -
1), DEC(jm, im), _
                                        rpc, AqB(jm, im - 1), AqB(jm, im), hx(i - 1 + r / 2), hx(i + r / 2), Hn(jm, im - 1), Hn(jm, im))
                b(jm, im) = b(jm, im) - qnoc(jm, im, 2)
                bbtt(jm, im - 1) = bbtt(jm, im - 1) + qnoc(jm, im, 2)
    End If
   End If
   If jm + 1 < m + 1 Then
    If CT(jm + 1, im) < 4 Then
      CC(jm, im) = khbar(SIT, AvTp, k(jm + 1, im), ho(jm + 1, im), dy(jm + 1), WR(jm + 1, im), hx(r / 2 + i + L), UMS(jm + 1, im), Bed(jm + 1, im),
bpp(jm + 1, im), AqB(jm + 1, im), _
                         k(jm, im), ho(jm, im), dy(jm), WR(jm, im), hx(r / 2 + i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dx(im), UMA(jm
+ 1, im), UMA(jm, im), ATC, _
                         AqT(jm + 1, im), AqT(jm, im), Lons(jm + 1, im), Lons(jm, im), CT(jm + 1, im), CT(jm, im), 2, SB)
      CCt(jm, im) = khbarat(SIT, SB, AvTp, kh(jm + 1, im), hoat(jm + 1, im), dy(jm + 1), Lons(jm + 1, im), WRAT(jm + 1, im), Bed(jm + 1, im), _
                            bpp(jm + 1, im), AqT(jm + 1, im), hx(i + L), UMS(jm + 1, im), kh(jm, im), hoat(jm, im), dy(jm), Lons(jm, im), WRAT(jm,
im), Bed(jm, im), _
                            bpp(jm, im), AqT(jm, im), hx(i), UMS(jm, im), 2, dx(im), UMAT(jm + 1, im), UMAT(jm, im), CT(jm + 1, im), CT(jm, im),
DEC(jm + 1, im), DEC(jm, im), _
                            rpc, AqB(jm + 1, im), AqB(jm, im), hx(i + L + r / 2), hx(i + r / 2), Hn(jm + 1, im), Hn(jm, im), UMA(jm, im))
                qnoc(jm, im, 3) = QnoCside(SIT, AvTp, AqT(jm, im), AqT(jm + 1, im), hx(i + L), hoat(jm + 1, im), UMAT(jm, im), kh(jm, im), _
                                        kh(jm + 1, im), dx(im), dy(jm), dy(jm + 1), WRAT(jm, im), WRAT(jm + 1, im), Bed(jm, im), _
                                        Bed(jm + 1, im), bpp(jm, im), bpp(jm + 1, im), Lons(jm, im), Lons(jm + 1, im), 2, _
                                        hoat(jm, im), SHD(jm + 1, im), SHD(jm, im), UMS(jm + 1, im), UMS(jm, im), DEC(jm, im), SB, _
                                        UMA(jm, im), UMA(jm + 1, im), hx(i), UMAT(jm + 1, im), Knocker, CT(jm + 1, im), CT(jm, im), DEC(jm + 1,
im), DEC(jm, im), _
                                        rpc, AqB(jm + 1, im), AqB(jm, im), hx(i + L + r / 2), hx(i + r / 2), Hn(jm + 1, im), Hn(jm, im))
                b(jm, im) = b(jm, im) - qnoc(jm, im, 3)
                bbtt(jm + 1, im) = bbtt(jm + 1, im) + qnoc(jm, im, 3)
    End If
   End If
   If im + 1 < L + 1 Then
    If CT(jm, im + 1) < 4 Then
      DD(jm, im) = khbar(SIT, AvTp, k(jm, im + 1), ho(jm, im + 1), dx(im + 1), WR(jm, im + 1), hx(r / 2 + i + 1), UMS(jm, im + 1), Bed(jm, im + 1),
bpp(jm, im + 1), AqB(jm, im + 1), _
                         k(jm, im), ho(jm, im), dx(im), WR(jm, im), hx(r / 2 + i), UMS(jm, im), Bed(jm, im), bpp(jm, im), AqB(jm, im), dy(jm),
UMA(jm, im + 1), UMA(jm, im), ATC, _
                         AqT(jm, im + 1), AqT(jm, im), Lons(jm, im + 1), Lons(jm, im), CT(jm, im + 1), CT(jm, im), 1, SB)
      DDt(jm, im) = khbarat(SIT, SB, AvTp, kh(jm, im + 1), hoat(jm, im + 1), dx(im + 1), Lons(jm, im + 1), WRAT(jm, im + 1), Bed(jm, im + 1), _
                            bpp(jm, im + 1), AqT(jm, im + 1), hx(i + 1), UMS(jm, im + 1), kh(jm, im), hoat(jm, im), dx(im), Lons(jm, im), WRAT(jm,
im), Bed(jm, im), _
                            bpp(jm, im), AqT(jm, im), hx(i), UMS(jm, im), 1, dy(jm), UMAT(jm, im + 1), UMAT(jm, im), CT(jm, im + 1), CT(jm, im),
DEC(jm, im + 1), DEC(jm, im), _
                            rpc, AqB(jm, im + 1), AqB(jm, im), hx(i + 1 + r / 2), hx(i + r / 2), Hn(jm, im + 1), Hn(jm, im), UMA(jm, im))
                qnoc(jm, im, 4) = QnoCside(SIT, AvTp, AqT(jm, im), AqT(jm, im + 1), hx(i + 1), hoat(jm, im + 1), UMAT(jm, im), kh(jm, im), _
                                        kh(jm, im + 1), dy(jm), dx(im), dx(im + 1), WRAT(jm, im), WRAT(jm, im + 1), Bed(jm, im), _
                                        Bed(jm, im + 1), bpp(jm, im), bpp(jm, im + 1), Lons(jm, im), Lons(jm, im + 1), 1, _
                                        hoat(jm, im), SHD(jm, im + 1), SHD(jm, im), UMS(jm, im + 1), UMS(jm, im), DEC(jm, im), SB, _
                                        UMA(jm, im), UMA(jm, im + 1), hx(i), UMAT(jm, im + 1), Knocker, CT(jm, im + 1), CT(jm, im), DEC(jm, im +
1), DEC(jm, im), _
                                        rpc, AqB(jm, im + 1), AqB(jm, im), hx(i + 1 + r / 2), hx(i + r / 2), Hn(jm, im + 1), Hn(jm, im))
                b(jm, im) = b(jm, im) - qnoc(jm, im, 4)
                bbtt(jm, im + 1) = bbtt(jm, im + 1) + qnoc(jm, im, 4)
     End If
   End If
```

```
   If Gflag(jm, im) < 1 Then
      GG(jm, im) = FGG(rpc(jm, im), kv(jm, im), dx(im), dy(jm), CT(jm, im), SB, Bed(jm, im), bpp(jm, im), AqT(jm, im), widpp(jm, im), widp(jm, im),
lenp(jm, im), AqB(jm, im), kvq(jm, im), AvTp, _
                       hx(i + r / 2), LK, SIT, hx(i), Hn(jm, im), UMA(jm, im), UMS(jm, im), GCT(jm, im), GCA(jm, im), UMAT(jm, im), vfhd, vfch)
                  If Gflag(jm, im) = -1 Then
                    GCT(jm, im) = GCT(jm, im) + 1
                    If hx(i + r / 2) >= AqT(jm, im) - UMA(jm, im) Then
                     GCA(jm, im) = GCA(jm, im) + 1
                    End If
                  End If
    ElseIf Gflag(jm, im) = 1 Then
      GG(jm, im) = FGG(rpc(jm, im), kv(jm, im), dx(im), dy(jm), CT(jm, im), SB, Bed(jm, im), bpp(jm, im), AqT(jm, im), widpp(jm, im), widp(jm, im),
lenp(jm, im), AqB(jm, im), kvq(jm, im), AvTp, _
                       797979, LK, SIT, hx(i), Hn(jm, im), UMA(jm, im), UMS(jm, im), GCT(jm, im), GCA(jm, im), UMAT(jm, im), vfhd, vfch)
   End If
    GGT(jm, im) = GG(jm, im)
  End If                              'CT = 4 end if.
  EE(jm, im) = EE(jm, im) + GG(jm, im) + FuncE(AA(jm, im), BB(jm, im), CC(jm, im), DD(jm, im), Trans(i + r / 2), TransB(i + r / 2), CT(jm, im),
dx(im), dy(jm), dt, WR(jm, im), ATC, AqT(jm, im), AqB(jm, im), Ss(jm, im), Sy(jm, im), SIT, Bed(jm, im), bpp(jm, im), hx(i + r / 2), SB, Hn(jm, im),
UMA(jm, im), UMS(jm, im))
  EET(jm, im) = EET(jm, im) + GGT(jm, im) + FuncET(AAt(jm, im), BBt(jm, im), CCt(jm, im), DDt(jm, im), Trans(i), TransB(i), CT(jm, im), dx(im),
dy(jm), dt, WRAT(jm, im), ATC, AqT(jm, im), Sigs(jm, im), Sigma(jm, im), SIT, Bed(jm, im), bpp(jm, im), hx(i), SB, Hnat(jm, im), UMS(jm, im),
UMAT(jm, im))
  b(jm, im) = b(jm, im) + FuncBB(Qc(jm, im), Trans(i + r / 2), TransB(i + r / 2), CT(jm, im), dx(im), dy(jm), dt, WR(jm, im), ATC, Hn(jm, im),
AqT(jm, im), AqB(jm, im), Ss(jm, im), Sy(jm, im), SIT, Bed(jm, im), bpp(jm, im), SB, hx(i + r / 2), ITS, UMA(jm, im), UMS(jm, im))
  bbtt(jm, im) = bbtt(jm, im) + FuncBT(Qc(jm, im), Trans(i), TransB(i), CT(jm, im), dx(im), dy(jm), dt, WRAT(jm, im), ATC, Hnat(jm, im), AqT(jm, im),
AqB(jm, im), Sigs(jm, im), Sigma(jm, im), SIT, Bed(jm, im), bpp(jm, im), SB, hx(i), ITS, UMAT(jm, im), UMS(jm, im), hoat(jm, im), UMA(jm, im))
  If bbtt(jm, im) = 9.99901110999 Then
       NegHead = 6
       NegHeadNM = nm + 38
        msgti = "Impulse Exceeds Aquitard Capacity. "
        flaw = 3
       GoTo Line50
  End If
  If hx(i + r / 2) <= AqTi(i) - UMA(jm, im) Then                                  'Aquitard flux limited (aquifer head drawn down below top).
    If hx(i) > AqTi(i) - UMA(jm, im) Then
         bbtt(jm, im) = bbtt(jm, im) - GG(jm, im) * ((AqT(jm, im) - UMA(jm, im)) - hx(i + r / 2))
         b(jm, im) = b(jm, im) + GG(jm, im) * ((AqT(jm, im) - UMA(jm, im)) - hx(i + r / 2))
    End If
  End If
   If CT(jm, im) = 2 Then
   If SB = "Restrictive" Then
     If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
      If hx(i) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        EET(jm, im) = EET(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) / bpp(jm, im)
        bbtt(jm, im) = bbtt(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * SHD(jm, im) / bpp(jm, im)
        BedMat(jm, im) = 1
       ElseIf hx(i) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        If hx(i) > AqT(jm, im) - UMA(jm, im) Then
          bbtt(jm, im) = bbtt(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) /
bpp(jm, im)
          BedMat(jm, im) = 2
         ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
          b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) / bpp(jm,
im)
          BedMat(jm, im) = 5
```

```
        End If
       End If
      ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
      If hx(i + r / 2) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        EE(jm, im) = EE(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) / bpp(jm, im)                          'Includes full bed flux term.
        b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * SHD(jm, im) / bpp(jm, im)             'Includes full bed flux term.
        BedMat(jm, im) = 3
       ElseIf hx(i + r / 2) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) / bpp(jm, im)
'Bed flux limited (drawn down below bottom).
        BedMat(jm, im) = 4
       End If
     End If
   End If                          'SB end if.
   End If                          'CT end if.
    i = i + 1
  Next im
Next jm
i = 1
 For jm = 1 To m
  For im = 1 To L
   If Trans(i) = 1 Then                                              'Aquitard flux limited (aquitard dewatered).
         Dewflag(i) = 1
         PermFlag(jm, im) = 1
         AAt(jm, im) = 0
         BBt(jm, im) = 0
         CCt(jm, im) = 0
         DDt(jm, im) = 0
         GGT(jm, im) = 0
         GG(jm, im) = 0
         EET(jm, im) = PermFact(jm, im, 1) * dx(im) * dy(jm) / dt
         bbtt(jm, im) = -EET(jm, im) * (AqT(jm, im) - UMA(jm, im))
         If ITS = "Deep Percolation" Then
          If CT(jm, im) = 1 Then
             If Qc(jm, im) < 0 Then                                     'Water infiltrates to layer.
               b(jm, im) = b(jm, im) + Qc(jm, im)
               b(jm, im) = b(jm, im) + dvtc(i) / dt
              ElseIf Qc(jm, im) = 0 Then                                   'Exfiltration case (>) residual not credited to lower layer.
               b(jm, im) = b(jm, im) + dvtc(i) / dt
             End If
            ElseIf CT(jm, im) <> 1 Then
             b(jm, im) = b(jm, im) + dvtc(i) / dt
           End If
          Else
            b(jm, im) = b(jm, im) + dvtc(i) / dt
         End If
   End If
   If CTP(jm, im) = 4 Then
    GoTo Line9d
   ElseIf CTP(jm, im) = 2 Then
   If SB = "Permissive" Then
Line9d:     '=========================
     AAt(jm, im) = 0
     BBt(jm, im) = 0
     CCt(jm, im) = 0
     DDt(jm, im) = 0
```

```
     GGT(jm, im) = 0
     EET(jm, im) = PermFact(jm, im, 1) * dx(im) * dy(jm) / dt
     bbtt(jm, im) = -EET(jm, im) * Hnpat(jm, im)
     PermFlag(jm, im) = 1
     If subsflag = 1 Then
      subsflag = 0
      GoTo line9e
     End If
   If CTP(jm, im) = 4 Then
    GoTo Line9dd
   ElseIf BedCP(jm, im) <= AqT(jm, im) Then
Line9dd:     '==========================
     AA(jm, im) = 0
     BB(jm, im) = 0
     CC(jm, im) = 0
     DD(jm, im) = 0
     GG(jm, im) = 0
     EE(jm, im) = PermFact(jm, im, 2) * dx(im) * dy(jm) / dt                    'Factor scales coefficients for procedure stability without
affecting result.
     b(jm, im) = -EE(jm, im) * Hnp(jm, im)
    ElseIf Bed(jm, im) > AqT(jm, im) Then
    End If
   End If                                  'SB end if.
   End If                                  'CT end if.
    If EET(jm, im) = 0 Then
      subsflag = 1
      GoTo Line9d
line9e:     '==========================
      bbtt(jm, im) = -EET(jm, im) * Hnat(jm, im)
      GoTo Line9f
    End If
    If AAt(jm, im) + BBt(jm, im) + CCt(jm, im) + DDt(jm, im) + GGT(jm, im) = 0 Then
     If Trans(i) <> 1 Then
              Dewflag(i) = 2
              If ITS = "Deep Percolation" Then
               If Qc(jm, im) < 0 Then                                   'Water infiltrates to layer.
                 If Qc(jm, im) >= -1 * kvtb(i) Then                     'At negative unit hydraulic gradient, set limiting condition, (percolation
not keeping aquitard saturated).  [Bed flux not checked, as implicit, since it does not jump around like impulse can.]
                  b(jm, im) = b(jm, im) + Qc(jm, im)                        'Lands at PermFlag.
                 Else
                  GoTo Line9f
                End If
               ElseIf Qc(jm, im) = 0 Then                                   'Lands at PermFlag.
               ElseIf Qc(jm, im) > 0 Then
                 GoTo Line9f
              End If
             End If
             PermFlag(jm, im) = 1
             EET(jm, im) = PermFact(jm, im, 1) * dx(im) * dy(jm) / dt
               If hx(i) <= AqTi(i) - UMA(jm, im) Then
                 bbtt(jm, im) = -EET(jm, im) * (AqTi(i) - UMA(jm, im))
                Else
                 bbtt(jm, im) = -EET(jm, im) * Hnat(jm, im)
               End If
     End If
    End If
```

```
Line9f:     '=========================
    If PFSflag < 1 Then
     If MaxTerm < Abs(EE(jm, im)) Then
      MaxTerm = Abs(EE(jm, im))
     End If
     If MaxTermT < Abs(EET(jm, im)) Then
      MaxTermT = Abs(EET(jm, im))
     End If
    End If
    i = i + 1
  Next im
Next jm

If PFSflag < 1 Then
   PFSflag = 1
End If

'
'WRITE AUGMENTED MATRIX BY POPULATING COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------------------
ReDim SE(1 To r, 1 To 7), SEC(1 To r), UM(1 To r, 1 To 4), UT(1 To r, 1 To 4), DU(1 To r, 1 To 4) 'Redimension matrices to obviate memory errors(row
limit, column limit).
i = 1                               'Initialize system of equations row index, also index for model cell identification serial.
For jm = 1 To m                             'Initialize model space row index.  Note: convention for matrix indeces is different than model.  Model
columns are im, Matrix columns are j.
  For im = 1 To L                           'Initialize and reset model space column index.
   SE(r / 2 + i, 4) = EE(jm, im)         'Coefficient for cell's own next-step head.
   SE(i, 4) = EET(jm, im)
   SEC(r / 2 + i) = -1 * b(jm, im)             'Constant or intercept value for row.
   SEC(i) = -1 * bbtt(jm, im)
   If m > 1 Then
   If L > 1 Then
   If jm + 1 < m + 1 Then
    SE(r / 2 + i, 6) = -1 * CC(jm, im)           'Coefficient for next-step head in model cell beneath, matrix place L to the right.
    SE(i, 6) = -1 * CCt(jm, im)       'Aquitard cell overlying main model.
   End If
   If jm - 1 > 0 Then
    SE(r / 2 + i, 2) = -1 * AA(jm, im)           'Coefficient for next-step head in model cell above, matrix place L to the left.
    SE(i, 2) = -1 * AAt(jm, im)
   End If
   End If
   End If
   If L > 1 Then
    If im + 1 < L + 1 Then
     SE(r / 2 + i, 5) = -1 * DD(jm, im)           'Coefficient for next-step head in model cell to right, matrix place one to the right.
     SE(i, 5) = -1 * DDt(jm, im)
    End If
    If im - 1 > 0 Then
     SE(r / 2 + i, 3) = -1 * BB(jm, im)           'Coefficient for next-step head in model cell to the left, matrix place one to the left.
     SE(i, 3) = -1 * BBt(jm, im)
    End If
   Else
    If jm + 1 < m + 1 Then
     SE(r / 2 + i, 5) = -1 * CC(jm, im)           'Coefficient for next-step head in model cell beneath, matrix place one to the right.
     SE(i, 5) = -1 * CCt(jm, im)
    End If
```

```
    If jm - 1 > 0 Then
     SE(r / 2 + i, 3) = -1 * AA(jm, im)           'Coefficient for next-step head in model cell above, matrix place one to the left.
     SE(i, 3) = -1 * AAt(jm, im)
    End If
   End If
     SE(r / 2 + i, 1) = -1 * GG(jm, im)
     SE(i, 7) = -1 * GGT(jm, im)
      If PermFlag(jm, im) = 1 Then
       If PFSflag < 2 Then
        If SE(i, 4) < 0.75 * MaxTermT Then
         PermFact(jm, im, 1) = PermFact(jm, im, 1) * WorksheetFunction.RoundUp(0.75 * MaxTermT / SE(i, 4), 0) 'Increment factor to recompute
coefficient for mathematical stability.
         subsflag = 1
        End If
        If CTP(jm, im) = 4 Then
         GoTo line9ff
        ElseIf SB = "Permissive" Then
        If BedCP(jm, im) <= AqT(jm, im) Then
line9ff:     '=========================
         If SE(i + r / 2, 4) < 0.75 * MaxTerm Then
          PermFact(jm, im, 2) = PermFact(jm, im, 2) * WorksheetFunction.RoundUp(0.75 * MaxTerm / SE(i + r / 2, 4), 0) 'Increment factor to recompute
coefficient for mathematical stability.
          subsflag = 1
         End If
        End If
        End If
       End If
     End If
   i = i + 1
  Next im
Next jm
If subsflag = 1 Then
 PFSflag = 2
 subsflag = 0
 GoTo Line9
End If

Line9g:     '=========================
For i = 1 To r
 UM(i, 1) = (1 + delta) * SE(i, 4)           'Modified incomplete Cholesky factorization upper conditioner matrix.
Next i                                            'Initial loop sets all uii values to aii, unless delta non-zero.
For i = 1 To r
 If i - 1 > 0 Then
  UM(i, 1) = UM(i, 1) - (SE(i - 1, 5) ^ 2) / UM(i - 1, 1)                                    'Xi, Hill pg. 10.
  UM(i, 1) = UM(i, 1) - SE(i - 1, 5) * SE(i - 1, 7) / UM(i - 1, 1)                           'Theta.
     If i + r / 2 - 1 <= r Then
       UM(i + r / 2 - 1, 1) = UM(i + r / 2 - 1, 1) - SE(i - 1, 5) * SE(i - 1, 7) / UM(i - 1, 1)
     End If
  UM(i, 1) = UM(i, 1) - SE(i - 1, 5) * SE(i - 1, 6) / UM(i - 1, 1)                           'Phi.
     If i + L - 1 <= r Then
       UM(i + L - 1, 1) = UM(i + L - 1, 1) - SE(i - 1, 5) * SE(i - 1, 6) / UM(i - 1, 1)
     End If
  If i - L > 0 Then
   UM(i, 1) = UM(i, 1) - (SE(i - L, 6) ^ 2) / UM(i - L, 1)                                   'Tau.
   UM(i, 1) = UM(i, 1) - SE(i - L, 6) * SE(i - L, 7) / UM(i - L, 1)                          'Psi.
     If i + r / 2 - L <= r Then
```

```
      UM(i + r / 2 - L, 1) = UM(i + r / 2 - L, 1) - SE(i - L, 6) * SE(i - L, 7) / UM(i - L, 1)
    End If
  End If
  If i - r / 2 > 0 Then
   UM(i, 1) = UM(i, 1) - (SE(i - r / 2, 7) ^ 2) / UM(i - r / 2, 1)                          'Gamma.
  End If
 End If
 If UM(i, 1) < 0.001 Then
   If delta < 1000000 Then
     delta = 1.5 * delta + 0.001
     GoTo Line9g
   End If
 End If
 If i < r Then
  UM(i, 2) = SE(i, 5)
 End If
 If i <= r - L Then
  UM(i, 3) = SE(i, 6)
 End If
 If i <= r - r / 2 Then
  UM(i, 4) = SE(i, 7)
 End If
 DU(i, 1) = 1 'UM(i, 1) * (1 / UM(i, 1)) 'Diagonal matrix of 1/UM(i,1) times UM.
  DU(i, 2) = UM(i, 2) / UM(i, 1)
  DU(i, 3) = UM(i, 3) / UM(i, 1)
  DU(i, 4) = UM(i, 4) / UM(i, 1)
Next i

For i = 1 To r                          'Transpose of UM carried out for condensed format.
UT(i, 4) = UM(i, 1)
If i - 1 > 0 Then
UT(i, 3) = UM(i - 1, 2)
End If
If i - L > 0 Then
UT(i, 2) = UM(i - L, 3)
End If
If i - r / 2 > 0 Then
UT(i, 1) = UM(i - r / 2, 4)
End If
Next i

'SIMULTANEOUS SOLUTION OF SYSTEM OF EQUATIONS
'-----------------------------------------------------------------------------------------
'Modified Incomplete Cholesky Preconditioned CONJUGATE GRADIENT METHOD (MICCG)!!
'SEE "Preconditioned Conjugate-Gradient 2 (PCG2), A Computer Program for Solving Ground-Water flow Equations"
'by Mary C. Hill, USGS Water Resources Investigations Report 90-4048. 1990.
kitri = 0                    'Initialize iteration counter.
maxerrp = 1E+99              'Initialize prior iteration closure.
maxerr = 1                   'Initialize iteration closure.
GTCR = 0                     'Initialize GoTo Code for Residual Rounding.
trigger = 0                  'Initialize head adjustment trigger.
GTCRT = 0                    'Initialize trigger goto code.
Line10: '==========================       'Line label for top of solver.
ReDim Preserve SE(1 To r, 1 To 7), SEC(1 To r), hx(1 To r)
'Initialize Ax, rr, zz.
'-----------------quick homemade array multiplication
```

```
ReDim Ax(1 To r)
For i = 1 To r
Ax(i) = SE(i, 4) * hx(i)
If m > 1 Then
If L > 1 Then
 If i - L > 0 Then
  Ax(i) = Ax(i) + SE(i, 2) * hx(i - L)
 End If
 If i + L < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 6) * hx(i + L)
 End If
End If
End If
 If i - 1 > 0 Then
  Ax(i) = Ax(i) + SE(i, 3) * hx(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 5) * hx(i + 1)
 End If
 If i - r / 2 > 0 Then
  Ax(i) = Ax(i) + SE(i, 1) * hx(i - r / 2)
 End If
 If i + r / 2 < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 7) * hx(i + r / 2)
 End If
 Next i
'-----------------
ReDim Preserve Ax(1 To r)
ReDim rr(1 To r)
For i = 1 To r          'method step
rr(i) = SEC(i) - Ax(i)
Next i
          If GTCR = 2 Then
            GoTo Line14
           ElseIf GTCRT = 2 Then
            GoTo Line14
          End If
Do While maxerr > eps                       '#############Top of Method loop.
If kitri < 1000 Then                          'Iteration limit check.
Line11: '=========================
ReDim Preserve UT(1 To r, 1 To 4), rr(1 To r)
ReDim vv(1 To r)
For i = 1 To r          'forward sub
vv(i) = rr(i)
  If i - r / 2 > 0 Then
   vv(i) = vv(i) - UT(i, 1) * vv(i - r / 2)
  End If
  If L > 1 Then
   If m > 1 Then
    If i - L > 0 Then
      vv(i) = vv(i) - UT(i, 2) * vv(i - L)
    End If
   End If
  End If
  If i - 1 > 0 Then
   vv(i) = vv(i) - UT(i, 3) * vv(i - 1)
```

```
  End If
vv(i) = vv(i) / UT(i, 4)
Next i
ReDim Preserve DU(1 To r, 1 To 4), vv(1 To r)
ReDim zz(1 To r)
i = r
Do While i > 0           'back sub
zz(i) = vv(i)
  If i + r / 2 < r + 1 Then
   zz(i) = zz(i) - DU(i, 4) * zz(i + r / 2)
  End If
  If L > 1 Then
   If m > 1 Then
    If i + L < r + 1 Then
      zz(i) = zz(i) - DU(i, 3) * zz(i + L)
    End If
   End If
  End If
  If i + 1 < r + 1 Then
   zz(i) = zz(i) - DU(i, 2) * zz(i + 1)
  End If
'zz(i) = zz(i) / DU(i, 1)  DU(i,1)=1, so no need to divide.
i = i - 1
Loop
ReDim Preserve zz(1 To r)
          If GTCR = 1 Then
            GoTo Line12
          End If
If kitri = 0 Then
Line12: '==========================
 ReDim pp(1 To r)
 nip = 0
 For i = 1 To r
 pp(i) = zz(i)
 nip = nip + zz(i) * rr(i)
 Next i
           If GTCR = 1 Then
            ReDim Preserve pp(1 To r)
            GoTo Line13
           End If
ElseIf kitri > 0 Then
 dip = nip
 nip = 0
 For i = 1 To r
 nip = nip + zz(i) * rr(i)
 Next i
 beta = nip / dip
 For i = 1 To r
 pp(i) = zz(i) + beta * pp(i)
 Next i
End If
ReDim Preserve pp(1 To r)
kitri = kitri + 1         'Update iteration counter.
'-----------------array multiplication
Line13: '==========================
ReDim Preserve SE(1 To r, 1 To 7)
```

```
ReDim Ap(1 To r)
For i = 1 To r
Ap(i) = SE(i, 4) * pp(i)
If m > 1 Then
If L > 1 Then
 If i - L > 0 Then
  Ap(i) = Ap(i) + SE(i, 2) * pp(i - L)
 End If
 If i + L < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 6) * pp(i + L)
 End If
 End If
 End If
 If i - 1 > 0 Then
  Ap(i) = Ap(i) + SE(i, 3) * pp(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 5) * pp(i + 1)
 End If
 If i - r / 2 > 0 Then
  Ap(i) = Ap(i) + SE(i, 1) * pp(i - r / 2)
 End If
 If i + r / 2 < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 7) * pp(i + r / 2)
 End If
 Next i
 ReDim Preserve Ap(1 To r)
'-----------------
dip = 0                   'Remainder of method steps
For i = 1 To r            'For-Next gives internal products.
If Abs(nip) < 1E+100 Then      'Confirm stability.
dip = dip + pp(i) * Ap(i)
Else
 NegHead = 3
 NegHeadNM = nm + 38      'Intervene if not stable.
 GoTo Line50
End If
Next i
If nip <> 0 Then          'Zero residual check.
alpha = nip / dip
ReDim xp(1 To r)
For i = 1 To r            'Calculate next estimates.
xp(i) = hx(i)
hx(i) = hx(i) + alpha * pp(i)
rr(i) = rr(i) - alpha * Ap(i)
Next i
ReDim Preserve rr(1 To r), xp(1 To r), hx(1 To r)
           If GTCR = 1 Then
            GTCR = 2
            GoTo Line10
           ElseIf GTCR = 2 Then
            GoTo Line10
           End If
Line14: '==========================
Else              'Zero residual upshot.
 ReDim xp(1 To r)
```

```
 For i = 1 To r
  xp(i) = hx(i)
  hx(i) = hx(i)
  rr(i) = rr(i)
 Next i
End If            'Zero residual end if.

If trigger = 1 Then
 trigger = 0
 GTCRT = 0
 GoTo Line14a
End If
For i = 1 To r / 2
 If hx(i) < AqTi(i) - UMAi(i + r / 2) Then
  If -1 * dvtc(i) / dt > kvtb(i) Then
    hx(i) = (AqTi(i) - UMAi(i + r / 2)) + ((-1 * dvtc(i) / dt) - kvtb(i)) * dt / CAS(i)
   Else
    hx(i) = AqTi(i) - UMAi(i + r / 2)
  End If
  trigger = 1
 End If
Next i
If trigger = 1 Then
   GTCRT = 2
   GoTo Line10
End If
Line14a: '==========================

maxerr = 0
maxdelta = 0
For i = 1 To r                 'Calculate closure difference.
  If Abs(hx(i) - xp(i)) > maxerr Then
    maxerr = Abs(hx(i) - xp(i))
  End If
  If Abs(hx(i) - xp(i)) > maxdelta Then
    maxdelta = Abs(hx(i) - xp(i))
  End If
  If Abs(rr(i)) > maxerr Then
    maxerr = Abs(rr(i))
  End If
Next i
If maxdelta < 0.00000000001 Then
 If maxerr > eps Then
   If kitro + kitrtr < 100 Then
     maxerr = 0.5 * eps
     If kitri = 1 Then
      kitri = 2
     End If
    Else
     resord = -1 * WorksheetFunction.RoundDown(WorksheetFunction.Log10(maxerr), 0)
     eps = 1 * 10 ^ (-resord)
     msgres = "Residual Order = " & resord & ". "
   End If
 End If
End If
```

```
    If maxerrp <= maxerr Then                   'Watches for rounding error; if error evident, recalculates residual, instead of using erroneous
values.
    If GTCR = 0 Then
        GTCR = 1
        For i = 1 To r
            hx(i) = xp(i)
        Next i
        GoTo Line10
     ElseIf kitri > 50 Then
        subsflag = 76
    End If
    End If

Line15: '==========================

TransT = 0
Call Transition(i, r, Trans(), TransB(), ATC, Hni(), AqTi(), hx(), UMAi(), _
                CTi(), SIT, BEDi(), bppi(), UMSi(), Hniat(), UMATi(), _
                SB, CTP(), L, m, BedCP())
For i = 1 To r / 2
    If Trans(i) <> transp(i) Then
     TransT = 1
     ElseIf TransB(i) <> TransBp(i) Then
     TransT = 1
    End If
    If Trans(i + r / 2) <> transp(i + r / 2) Then
     TransT = 1
     ElseIf TransB(i + r / 2) <> TransBp(i + r / 2) Then
     TransT = 1
    End If
Next i
If subsflag = 1 Then
 subsflag = 0
 GoTo Line48
End If
If subsflag = 76 Then
  subsflag = 0
  GoTo Line16
End If
If subsflag = 77 Then
  subsflag = 0
  GoTo Line17
End If

If kitro + kitrtr < 200 Then
If kitri > 100 Then
Line15a: '==========================

If TransT = 1 Then
 GoTo Line16
End If

i = 1
For jm = 1 To m
For im = 1 To L
       If CT(jm, im) <> 4 Then
```

```
       If hx(i + r / 2) <= AqBi(i) Then
        GoTo Line16
       End If
       If ITS = "Deep Percolation" Then
        If hx(i) <= AqT(jm, im) - UMA(jm, im) Then
         GoTo Line16
        End If
       End If
       End If
i = i + 1
Next im
Next jm

Call KickOut(r, m, L, CT(), xpo(), hx(), rpc(), SB, SIT, _
            UMS(), BedMat(), UMA(), AqT(), Bed(), widpp(), bpp(), bppb(), widp(), _
            UMAT(), SHD(), KOC, AqBi(), ITS)
If KOC = 1 Then
   GoTo Line16
End If

End If                            'kitri.
Else
If kitri > 4 Then
GoTo Line15a
End If
End If

maxerrp = maxerr  'Set prior maxerr.

Else                                                 'Intervene to end iteration if not converging in a reasonable number of cycles.
 NegHead = 3
 NegHeadNM = nm + 38
 GoTo Line50
End If                                               'Iteration counter end if.

Loop                                        '##############Bottom of method loop.

Line16: '==========================

For i = 1 To r / 2
      If CTi(i) Mod 2 = 1 Then
        If hx(i + r / 2) <= AqBi(i) Then         'Check for dewatering.
          If kitro + kitrtr > 500 Then
           NegHead = 2
           NegHeadNM = nm + 38
           GoTo Line50
          Else
            If (Hni(i + r / 2) - AqBi(i)) > 0.2 * (AqTi(i) - AqBi(i)) Then
              hx(i + r / 2) = AqBi(i) + 0.2 * (AqTi(i) - AqBi(i))
             Else
              hx(i + r / 2) = Hni(i + r / 2)
            End If
            maxerr = 2 * eps
          End If
        End If
        If ITS = "Deep Percolation" Then
```

```
          If hx(i) <= AqTi(i) - UMAi(i + r / 2) Then
             If hx(i + r / 2) > AqTi(i) - UMAi(i + r / 2) Then
              If kitro + kitrtr < 500 Then
                hx(i) = hx(i + r / 2)
                maxerr = 2 * eps
               Else
                GoTo Line16a
              End If
              Else
Line16a: '==========================
               msgti = "Impulse Exceeds Aquitard Capacity. "
               flaw = 3
               NegHead = 46
               NegHeadNM = nm + 38
               GoTo Line50
             End If
            End If
          End If
        End If
Next i
If maxerr = 2 * eps Then
  subsflag = 77
  GoTo Line15
End If

Line17: '==========================
ReDim ho(1 To m, 1 To L), hoat(1 To m, 1 To L)
i = 1
For jm = 1 To m
 For im = 1 To L
  If CT(jm, im) <> 4 Then
ho(jm, im) = hofunc(hx(i + r / 2), AqT(jm, im), AqB(jm, im), UMA(jm, im), ATC, UMS(jm, im), Bed(jm, im), bpp(jm, im), WR(jm, im)) 'Characteristics
update.
hoat(jm, im) = hoatfunc(rpc(jm, im), hx(i), AqT(jm, im), AqB(jm, im), hx(i + r / 2), UMAT(jm, im), Hni(i + r / 2), UMA(jm, im), UMS(jm, im), Bed(jm,
im), bpp(jm, im), WRAT(jm, im))
  If AqT(jm, im) + hoat(jm, im) > DEC(jm, im) Then
   hoat(jm, im) = DEC(jm, im) - AqT(jm, im)
  End If
  End If
   i = i + 1
  Next im
Next jm
If subsflag = 1 Then
  subsflag = 0
  GoTo Line9
End If

Line17b: '==========================
ReDim Dewflag(1 To r / 2)
i = 1
For jm = 1 To m
For im = 1 To L
    If Trans(i) <> transp(i) Then
     TransC(i) = TransC(i) + 1
    End If
    If Trans(i + r / 2) <> transp(i + r / 2) Then
```

```
     TransC(i + r / 2) = TransC(i + r / 2) + 1
    ElseIf TransB(i + r / 2) <> TransBp(i + r / 2) Then
     If WRAT(jm, im) = 1 Then
      If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
       TransC(i + r / 2) = TransC(i + r / 2) + 1
      End If
     End If
    End If
      If TransC(i + r / 2) = 20 Then
       If Gflag(jm, im) = 0 Then
        Gflag(jm, im) = -1
       End If
      ElseIf TransC(i + r / 2) = 40 Then
       If Gflag(jm, im) < 1 Then
        Gflag(jm, im) = 1
        If xp(i) > AqTi(i) - UMAi(i + r / 2) Then
         GoTo Line17c
        ElseIf hx(i) > AqTi(i) - UMAi(i + r / 2) Then
Line17c: '=========================
        If msgtg = "" Then
         If kvq(jm, im) <> 999999.12321 Then
          msgtg = "Vertical Flux Distance Modulation. "
         End If
        End If
        End If
       End If
       End If
  xpo(i) = hx(i)
  xpo(i + r / 2) = hx(i + r / 2)
transp(i) = Trans(i)                                  'Set prior transition for next coefficient cycle.
transp(i + r / 2) = Trans(i + r / 2)
TransBp(i) = TransB(i)                                  'Set prior transition for next coefficient cycle.
TransBp(i + r / 2) = TransB(i + r / 2)
i = i + 1
Next im
Next jm

If TransT > 0 Then
  If kitrtr < 1000 Then
     kitrtr = kitrtr + 1
     GoTo Line9
    Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
 ElseIf maxerr > eps Then
  If kitro < 1000 Then
     kitro = kitro + 1
     GoTo Line9
   Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
End If
```

```
If GTC = 40 Then
GoTo Line40
End If
'---------------------------------------------------------------------------------------------
'END SIMULTANEOUS SOLUTION by MICCG!!
'
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Recursive portions of solution follow initial solution for time step.
  '##########Begin outside iteration (recursive solution loop) for aquifer thickness.
Line31: '=========================

Do While kitri > 1   '----------------Top of outside iteration loop, continues until convergence on first inner iteration after thickness update.
If kitro < 1000 Then                    'Outside iteration limit check if.
kitro = kitro + 1
  GTC = 40
  GoTo Line9    'Recompute coefficients from new head estimates, from there, repeat simultaneous solution by MICCG.
Line40: '=========================
 Else                                                       'Intervene to end outside iteration if not converging in a reasonable number of cycles.
  NegHead = 3
  NegHeadNM = nm + 38
  GoTo Line50
 End If                                                     'Outside iteration counter end if.
Loop                '----------------Bottom of outside iteration loop.
   '##########End outside iteration for aquifer thickness.
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Last of recursion.

Line45: '=========================
Call FortyFive(AqT(), CT(), DEC(), flaw, Hni(), Hniat(), Hnn(), Hnnat(), Hno(), Hnoat(), _
               ho(), hoat(), hoo(), hx(), ic, IMS, ITS, msgat, msgc, msgmt, msgt, msgtg, _
               msgti, msgumat, MSTA, nc, nm, o, Qc(), QCP(), Qm(), qs, Trans(), Transnm(), _
               Transpn(), Transpnm(), m, L, r, NegHead, NegHeadNM, subsflag, eps, TransB())
If subsflag = 50 Then
 subsflag = 0
 GoTo Line50
End If

'
'OUTPUT  (starts before end of time step loop)
'-------------------------------------------------------------------
RSP(n) = 0                          'Prepare for response flow calculation...............................................................
RSPA(n) = 0
ReDim Qnno(1 To m, 1 To L)
 Call Qnout(RSP(), RSPA(), Qnno(), i, jm, im, m, L, CT(), SB, Bed(), bpp(), AqT(), hx(), UMS(), kpp(), widpp(), lenp(), SHD(), msgbg, _
            SIT, WRAT(), Hnnat(), Hnn(), CC(), DD(), AA(), BB(), GG(), UMA(), AqTi(), CCt(), _
            DDt(), AAt(), BBt(), zone(), qrs, dt, UCF, n, qnoc(), r, Lons(), CTP(), BedCP())

subsflag = 1                           'Independent domain budget block.
GoTo Line15
Line48: '=========================
Qtt = qrs * Qtt                        'Volume into aquifer during step.
RSP(n) = (-1) * UCF * RSP(n)           'Volume out.
dS = 0
i = 1
For jm = 1 To m
  For im = 1 To L                      'Change in storage.
```

```
    dS = dS + FuncdV(Trans(i + r / 2), TransB(i + r / 2), WR(jm, im), ATC, (Hno(jm, im) + UMA(jm, im)), (Hnn(jm, im) + UMA(jm, im)), AqT(jm, im),
AqB(jm, im), Ss(jm, im), Sy(jm, im), SIT, Bed(jm, im), CT(jm, im), bpp(jm, im), SB, dx(im), dy(jm), UMA(jm, im), UMS(jm, im))
    dS = dS + FuncdVat(Trans(i), TransB(i), WRAT(jm, im), ATC, (Hnoat(jm, im) + UMAT(jm, im)), (Hnnat(jm, im) + UMAT(jm, im)), AqT(jm, im), Sigs(jm,
im), Sigma(jm, im), SIT, Bed(jm, im), CT(jm, im), bpp(jm, im), SB, dx(im), dy(jm), UMAT(jm, im), UMS(jm, im), UMAT(jm, im))
    i = i + 1
  Next im
Next jm
dS = qs * dS
BD = Abs(Qb - (dS + RSP(n)))          'Budget difference: volume in vs. change in storage plus volume out.
BLT = BLTF(Qb, dS, RSP(n), ic, Qm(nm), UCF, nc) 'Largest term.
If BLT > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then 'Minimum scale for evaluation.
If BD > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then
If 100 * BD / BLT > 0.1 Then 'Difference divided by largest term, converted to percentage, compared to acceptance level...1/10 of 1%.
 If msgc <> "Moist Surface! " Then
   NegHeadNM = nm + 38
   NegHead = 5
   If Abs(IMPSMX) + Abs(IMPSMN) > 0.00001 Then
     msgep = "Budget Gap. "
    Else
     msgep = "No Volume Budget. "
   End If
   GoTo Line50
  End If
 End If
 End If
 End If
RSP(n) = RSP(n) / ((-1) * UCF)
Qtt = Qtt / qrs

   If HTII > 0 Then                 'Stash head to plot.
    If o = nc Then
         HTS(nm) = -1 * Hnn(HTIJ, HTII)
         HTS2(nm) = -1 * Hnnat(HTIJ, HTII)
    End If
   End If
If nm = nh Then
  If o = nc Then
   For im = 1 To L
    For jm = 1 To m
      Hnnp(jm, im) = -1 * Hnn(jm, im)
      Hnnpat(jm, im) = -1 * Hnnat(jm, im)
      If Hnnpat(jm, im) = -(AqT(jm, im) - UMA(jm, im)) Then
       ATDF = 1
      End If
    Next jm
  Next im
 End If
End If

n = n + 1                         'TIME STEP PROGRESSION.
o = o + 1
If o = nc + 1 Then
 o = 1
 nm = nm + 1
End If
```

```
For im = 1 To L
 For jm = 1 To m
   Hn(jm, im) = Hnn(jm, im)      'UPDATE HEAD.
   Hno(jm, im) = Hn(jm, im)
   Hnat(jm, im) = Hnnat(jm, im)
   Hnoat(jm, im) = Hnat(jm, im)
 Next jm
Next im

Loop                             'Time step loop!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                 'END OF TIME STEP COMPUTATION LOOP (continue output section)
Line50: '==========================

 Sheet10.Unprotect Password:=CPC
 Sheet10.Cells(133, 2).Value = "Head Time Series at Node"
 If HTII > 0 Then
   Sheet10.Cells(134, 7).Value = HTII
   Sheet10.Cells(135, 7).Value = HTIJ
    If NegHead > 1 Then
       hrb = NegHeadNM - 38 + 140 - 1
      Else
       hrb = no + 140
    End If
       Sheet10.Range("c140:" & "c" & hrb).Value = WorksheetFunction.Transpose(HTS())        'Output head time series.
       Sheet10.Range("d140:" & "d" & hrb).Value = WorksheetFunction.Transpose(HTS2())
   Else
   Sheet10.Cells(135, 3).Value = "To plot an output head time series, please specify, on the Time tab, non-zero node indeces for the desired
location.  Thank you!"
 End If
     Sheet10.Cells(72, 2).Value = "Aquifer Head Surface Output"
     Sheet10.Cells(72, 55).Value = "Aquitard Head Surface Output"
If nh > 0 Then
  Sheet10.Cells(73, 7).Value = nh
If nh < no + 1 Then
hrb = 76 + m
hrr = crf(L)
hrr2 = crf2(L)
If msgsep = "2" Then
  Sheet10.Range("bc74").Value = 1
  Else
  Sheet10.Range("bc74").Value = 0
End If
Sheet10.Range("bc73").Value = ATDF
Sheet10.Range("c77:" & hrr & hrb).Value = Hnnp()        'Output head to plot.
Sheet10.Range("bd77:" & hrr2 & hrb).Value = Hnnpat()
Else
GoTo Line60
End If
Else
Line60: '==========================
Sheet10.Cells(75, 3).Value = "To plot head, please specify, on the Time tab, a period number for plotting that is greater than zero and less than or
equal to the number of simulation periods.  Thank you!"
End If
  If NegHead > 1 Then
  If NegHeadNM - 38 - 1 < nh Then
  hrb = 76 + m
```

```
  hrr = crf(L)
  hrr2 = crf2(L)
  Sheet10.Range("c77:" & hrr & hrb).ClearContents
  Sheet10.Range("bd77:" & hrr2 & hrb).ClearContents
  End If
  End If
 Sheet10.Protect Password:=CPC

'Compute Impulse and Response Outputs and Totals.
Call IROT(RSPT, RSPTA, IMPST, n, nm, nn, o, RSPO(), RSPOA(), nc, RSP(), RSPA(), Qm(), IMPSMX, IMPSMN)

Sheet11.Unprotect Password:=CPC

Sheet11.Cells(20, 20).Value = CTPflg

hrb = 38 + no
Sheet11.Cells(6, 22).Value = flaw               'Flag indicates whether oscillation flaw or no.
Sheet11.Range("e29").ClearContents              'Message prep.
   If msgh222(msgim2, msgti, msgep, NegHead) > 1 Then
   Sheet11.Cells(6, 22).Value = msgh222(msgim2, msgti, msgep, NegHead)
   End If
If hrb222(msgim2, msgti, msgep, NegHead, NegHeadNM) > 0 Then
hrb = hrb222(msgim2, msgti, msgep, NegHead, NegHeadNM)
End If
   msgh = msgpb & msgh
   msgtt = msgtt & FuncNegTT(NegHead)
If msgep = "Budget Gap. " Then
Sheet11.Cells(10, 10).Value = epse
Else
Sheet11.Cells(10, 10).ClearContents
End If
Sheet11.Cells(10, 11).Value = fun1011(NegHead, msgres, epse)

If hrb > 38 Then
Sheet11.Range("e39:" & "e" & hrb).Value = WorksheetFunction.Transpose(RSPOA())     'Output response time series, Zone 1, then total.
Sheet11.Range("f39:" & "f" & hrb).Value = WorksheetFunction.Transpose(RSPO())
 If IMPSMX <> 999999999.123457 Then
  Else
  msgh = "Impulse Alternation. "
 End If
End If

Call SOB(St, PD, no, NegHead, hrb, RSPOFM, CPC, RSPO(), _
Qm(), IMPST, IMPSMX, IMPSMN, RSPT)

Else                                    'Thickness input else.
Sheet11.Unprotect Password:=CPC
   msgh = msghfun(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1))
   If msgh22(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1)) = 2 Then
   Sheet11.Cells(6, 22).Value = 2
   End If
End If                                  'End thickness input if.
   If TipW(3) = 1 Then
   msgtp = "Principal Aquifer Top at or above Initial Piezometric Surface. "
   End If
```

```
Sheet11.Protect Password:=CPC

Call Messenger(msgc, msgtp, msgim, msgim2, msgt, msgti, msgbg, msgat, msgtt, msgh, msgres, msg, msgmt, msgtg, msgep, CPC, msgu, msgumat, msgclt, _
               msgwp, msgsep, msgctp, msgrza)

End Sub
'*************************************************************************************************************************************************
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
'*************************************************************************************************************************************************
Sub IMTC()                          'Solution of implicit representation for Complete incision adjustment.

Dim dt As Double                      'Dimension calculation time step.
Dim dx() As Double, dy() As Double    'Dimension increments.
Dim k() As Double                     'Dimension hydraulic conductivity.
Dim Sy() As Double, Ss() As Double    'Dimension storage properties.
Dim hoo() As Double, ho() As Double   'Dimension initial saturated thickness, transmitting thickness.
Dim AqT() As Double, AqB() As Double  'Dimension aquifer top, bottom.
Dim Hn() As Double, Hno() As Double   'Dimension initial head, original for step, single index vector.
Dim Hni() As Double, Hnp() As Double  'Dimension single index initial head, permanent initial head.
Dim Qd() As Double, Qm() As Double    'Dimension net flow impulse distribution, impulse time series.
Dim Qdmx As Double, Qdmn As Double    'Dimension impulse distribution max and min.
Dim Qdmna  As Double                  'Dimension distribution absolute value min.
Dim Qc() As Double, QCP() As Double   'Dimension impulse for calculation.
Dim AA() As Double, BB() As Double    'Dimension flow coefficients: A, B.
Dim CC() As Double, DD() As Double    'Dimension flow coefficients.
Dim EE() As Double, b() As Double     'Dimension volume coefficients.
Dim SE() As Double, SEC() As Double   'Dimension matrix to hold system of equations, redimensioned below with element limits.
Dim Hnn() As Double, Hnnp() As Double 'Dimension next time step differential head, head to plot.
Dim Qnno() As Double                  'Dimension next time step flow output, to be determined for response cells, converting calculated head by
Storage Coefficient, etc.
Dim RSP() As Double, RSPA() As Double 'Dimension Response, Zone Response.
Dim RSPO() As Double, RSPT As Double  'Dimension Response Output, Response Total.
Dim RSPOA() As Double, RSPTA As Double 'Dimension Response Output, Response Total, both for Zone.
Dim i As Integer, z As Integer         'Dimension matrix row index, confinement threshold index.
Dim im As Integer, jm As Integer      'Dimension model row, column indices.
Dim L As Integer, m As Integer, r As Integer 'Model: cells on one side, cells on the other side; system rows.
Dim ITS As String, IMS As String      'Dimension Impulse Type Specification, Impulse Magnitude Specification.
Dim CT() As Integer, CTi() As Integer 'Dimension Cell Type.
Dim IA As Double, IAP As Double       'Dimension Input Area effects.
Dim qs As Double, qrs As Double       'Dimension Flow Sign, Flow Response Sign.
Dim PD As String, ic As String        'Dimension Period Denomination, Impulse Character.
Dim u As Long, n As Long, nc As Long  'Dimension units per period, number of periods, number of sub-periods.
Dim nn As Long, nh As Long            'Dimension number of proxy periods, number of period to plot heads.
Dim no As Long, nm As Long            'Dimension original number of periods, number of of period in undivided increments.
Dim o As Long, deno As Long           'Dimension period counter, characteristic impulse denominator.
Dim ATC As String                     'Dimension Aquifer Thickness Character.
Dim UCF As Double                     'Dimension Unit Conversion Factor.
Dim zone() As String                  'Dimension response Zone.
Dim msg As String, msgc As String     'Dimension warning messages.
Dim msgt As String, msgti As String   'Dimension warning messages.
Dim msgbg As String                   'Dimension warning messages.
Dim msgtt As String, msgh As String   'Dimension warning messages.
Dim msgtp As String, msgmt As String  'Dimension warning messages.
Dim msgim As String, msgim2 As String 'Dimension warning messages.
```

```
Dim msgpb As String, msgep As String  'Dimension warning messages.
Dim msgtg As String                   'Dimension warning messages.
Dim MSTA As Double                    'Dimension minimum saturated thickness allowed.
Dim DEC() As Double                   'Dimension depth of earth cover.
Dim IMPST As Double, FCR As Double    'Dimension impulse total, final cumulative ratio.
Dim SO As String                      'Dimension Secondary Output type.
Dim SOV() As Variant                  'Dimension Secondary Output variable.
Dim CUMI As Double, CUMR As Double    'Dimension cumulative impulse and response, respectively, through any time step.
Dim CUMRn() As Double                 'Dimension cumulative response at each time step.
Dim tt As Long                        'Dimension secondary output period index for impact factor calculation.
Dim CPC As Variant                    'Dimension some dummy text here.
CPC = "*******"                       'Dimension nothing to see here again.
Dim RSPOFM As Integer                 'Dimension response output flag to catch initial head imbalance.
Dim RSPOFM2 As Integer                'Dimension stream head imbalance flag.
Dim count2 As Double, count13 As Double 'Dimension storage and area weighted counters for cell types.
Dim headsum2 As Double                'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headave2 As Double                'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headsum2s As Double               'Dimension stream head sum.
Dim headsum13 As Double               'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headave13 As Double               'Dimension initial storage and area weighted head sums and averages by cell type.
Dim SB As String, SIT As String       'Dimesion relative permeability class of streambed, streambed incision type.
Dim kpp() As Double, bpp() As Double  'Dimension discrete streambed conductivity, thickness.
Dim kppb() As Double, bppb() As Double 'Dimension stream bank parameters.
Dim bppi() As Double, widp() As Double 'Dimension bed thickness single index, dummy width.
Dim widpp() As Double, lenp() As Double 'Dimension streambed width, length.
Dim hrb As Variant, hrr As Variant    'Dimension output cell range extents.
Dim epse As Integer                   'Dimension iteration closure threshold exponent.
Dim ThickCon As String                'Dimension thickness contribution option...update timing.
Dim IMPSMX As Double, IMPSMN As Double 'Dimension impulse max and min.
Dim NegHead As Integer                'Dimension negative head flag.
Dim NegHeadNM As Integer              'Dimension neghead error period.
Dim AqBi() As Double, AqTi() As Double 'Dimension single-index variables for bottom and top.
Dim AvTp As String                    'Dimension transmission characteristic average type.
Dim Bed() As Double, BEDi() As Double 'Dimension discrete bed elevation.
Dim SHD() As Double                   'Dimension stream head.
Dim TipW() As Integer                 'Dimension thickness input parameter watch variable.
Dim hx() As Double, eps As Double     'Dimension successive approximation head vector, convergence threshold for MICCG.
Dim Ax() As Double, rr() As Double    'Dimension MICCG solver variables.
Dim vv() As Double, zz() As Double    'Dimension MICCG variables.
Dim UT() As Double, DU() As Double    'Dimension MICCG solver preconditioner matrices.
Dim UM() As Double                    'Dimension MICCG upper preconditioner matrix.
Dim pp() As Double, Ap() As Double    'Dimension MICCG solver variables.
Dim nip As Double, dip As Double      'Dimension MICCG solver variables.
Dim alpha As Double, beta As Double   'Dimension MICCG variables.
Dim xp() As Double, maxerr As Double  'Dimension MICCG solver variables.
Dim maxerrp As Double                 'Dimension prior max error.
Dim maxdelta As Double                'Dimension maximum head estimate change.
Dim resord As Integer                 'Dimension residual order.
Dim msgres As String                  'Dimension residual order message.
Dim kitr As Integer, kitri As Integer 'Dimension iteration trackers.
Dim kitrtr As Integer, kitro As Integer 'Dimension iteration trackers.
Dim GTC As Integer, GTCR As Integer   'Dimension GoTo codes.
Dim St As String                      'Dimension schedule type.
Dim Trans() As Integer                'Dimension pressurization transition flag.
Dim transp() As Integer               'Dimension prior pressurization transition flag.
Dim TransT As Integer                 'Dimension pressurization transition aggregate flag.
```

```
Dim Transnm() As Integer              'Dimension period transition.
Dim Transpnm() As Integer             'Dimension prior period transition.
Dim IFT As Double                     'Dimension impact factor total.
Dim IGTC As Integer                   'Dimension impulse distribution GoTo code.
Dim dH As Double, Qt As Double        'Dimension Newton solver variables for impulse distribution.
Dim Qtt As Double, Qf As Double       'Dimension impulse distribution variables.
Dim ObF As Double, ObFF As Double     'Dimension Newton solver variables for impulse distribution.
Dim dFdu As Double                    'Dimension Newton solver variable, prior Qttn.
Dim dS As Double                      'Dimension change in storage for global budget.
Dim BD As Double, BLT As Double       'Dimension budget difference, budget largest term.
Dim PermFact() As Double              'Dimension permissive bed coefficient factor.
Dim HTS() As Double                   'Dimension head time series output.
Dim HTII As Integer, HTIJ As Integer  'Dimension head time series location im index, jm index.
Dim UMS() As Double, UMA() As Double  'Dimension Ultimate Matric Suction below streambed, aquifer top.
Dim UMAi() As Double, UMSi() As Double 'Dimension single index variables.
Dim Qttp As Double                    'Dimension previous impulse.
Dim Transpn() As Integer              'Dimension previous transition.
Dim flaw As Integer                   'Dimension flaw flag.
Dim hxa As Double, hxb As Double      'Dimension head place holders for channel side computation.
Dim hxc As Double, hxd As Double      'Dimension head place holders.
Dim side11() As Double, side12() As Double 'Dimension wet channel side for complete incision adjustment.
Dim side21() As Double, side22() As Double 'Dimension wet channel side for complete incision adjustment.
Dim side31() As Double, side32() As Double 'Dimension other channel side.
Dim side41() As Double, side42() As Double 'Dimension other channel side.
Dim Lons() As Integer, subsflag As Integer 'Dimension long side tracker, routine substitution flag.
Dim Lon1 As Integer, Lon2 As Integer  'Dimension long side roation trackers.
Dim msgu As String, msgumat As String 'Dimension matric suction messages.
Dim csea() As Double, csba() As Double 'Dimension channel side e and b coefficient, aquifer.
Dim SidMat() As Integer               'Dimension channel side head condition flag.
Dim BedMat() As Integer               'Dimension streambed head condition flag.
Dim msgclt As String, msgwp As String 'Dimension Complete override, stream width messages.
Dim ovp() As Double, ovw() As Double  'Dimension bank overlap variables.
Dim beds() As Double                  'Dimension surrounding bed elevation variable.
Dim THRS() As Double                  'Dimension Storage Parameter Threshold level stack.
Dim THRCNT() As Integer               'Dimension Storage Parameter Threshold level count.
Dim S As Double                       'Dimension single value Storage Coefficient for use in loops.
Dim hob() As Double                   'Dimension complete incision adjustment height from node to edges.
Dim NBRS() As Integer                 'Dimension nearby response cell counter per side.
Dim ph As Integer                     'Dimension stream orientation placeholder.
Dim delta As Double                   'Dimension MICCG upper preconditioner matrix diagonal scale factor to prevent negatives.
Dim PFSflag As Integer                'Dimension PermFact scale flag.
Dim MaxTerm As Double                 'Dimension maximum augmented matrix entry value.
Dim SEB As String                     'Dimension stream end bank option.
Dim Qb As Double                      'Dimension domain step impulse for budget block.

L = Sheet3.Cells(12, 8).Value         'Load number of cells on one side of model.
m = Sheet3.Cells(15, 8).Value         'Load number of cells on other side of model.
r = L * m                             'Number of rows in equation matrix.

no = Sheet11.Cells(13, 5).Value       'Load number of periods.
nc = Sheet11.Cells(16, 5).Value       'Load number of sub-period steps.
nn = no * nc                          'Set number of total steps.

ReDim dx(1 To L), dy(1 To m)          'Set custom array dimensions.
ReDim k(1 To m, 1 To L)
ReDim hoo(1 To m, 1 To L), ho(1 To m, 1 To L)
```

```
ReDim Hno(1 To m, 1 To L), Hn(1 To m, 1 To L)
ReDim Hnp(1 To m, 1 To L)
ReDim Qd(1 To m, 1 To L), Qm(1 To no)
ReDim Hnn(1 To m, 1 To L), Hnnp(1 To m, 1 To L)
ReDim Qnno(1 To m, 1 To L)
ReDim RSP(1 To nn), RSPA(1 To nn)
ReDim RSPO(1 To no)
ReDim RSPOA(1 To no)
ReDim CT(1 To m, 1 To L)
ReDim zone(1 To m, 1 To L)
ReDim DEC(1 To m, 1 To L)
ReDim Sy(1 To m, 1 To L), Ss(1 To m, 1 To L)
ReDim SOV(1 To no)
ReDim CUMRn(0 To no)
ReDim kpp(1 To m, 1 To L), bpp(1 To m, 1 To L), bppi(1 To r)
ReDim kppb(1 To m, 1 To L), bppb(1 To m, 1 To L)
ReDim widp(1 To m, 1 To L), widpp(1 To m, 1 To L)
ReDim lenp(1 To m, 1 To L)
ReDim AqBi(1 To r), AqTi(1 To r)
ReDim Bed(1 To m, 1 To L), BEDi(1 To r), SHD(1 To m, 1 To L)
ReDim AqT(1 To m, 1 To L), AqB(1 To m, 1 To L)
ReDim TipW(1 To 5)
ReDim Transpnm(1 To m, 1 To L)
ReDim Hni(1 To r), CTi(1 To r)
ReDim Qc(1 To m, 1 To L), QCP(1 To m, 1 To L)
ReDim HTS(0 To no)
ReDim UMS(1 To m, 1 To L), UMA(1 To m, 1 To L)
ReDim UMAi(1 To r), UMSi(1 To r)
ReDim PermFact(1 To m, 1 To L), Lons(1 To m, 1 To L)
ReDim Transpn(1 To r)
ReDim ovp(1 To m, 1 To L, 1 To 4), ovw(1 To m, 1 To L, 1 To 4)
ReDim beds(1 To m, 1 To L, 1 To 4)
ReDim THRS(1 To m, 1 To L, 1 To 5)
ReDim THRCNT(1 To m, 1 To L)
ReDim hob(1 To m, 1 To L, 1 To 4)
ReDim NBRS(1 To m, 1 To L, 1 To 4)

ReDim hx(1 To r)

'
'INPUT
'---------------------------------------------------------------------
RSPOFM = 1                          'Initialize flags all clear.
RSPOFM2 = 1
subsflag = 0

St = WorksheetFunction.Proper(Sheet11.Cells(12, 5).Value) 'Load schedule type.
SB = WorksheetFunction.Proper(Sheet12.Cells(7, 6).Value)  'Load relative permeability class of streambed.
SIT = WorksheetFunction.Proper(Sheet12.Cells(8, 6).Value) 'Load whether streambed incision represented.
SO = WorksheetFunction.Proper(Sheet11.Cells(19, 5).Value)      'Load secondary output type.
ITS = WorksheetFunction.Proper(Sheet3.Cells(7, 8).Value)       'Load Impulse Type Specification.
IMS = WorksheetFunction.Proper(Sheet8.Cells(7, 6).Value)       'Load Impulse Magnitude Specification.
ic = WorksheetFunction.Proper(Sheet11.Cells(17, 5).Value)      'Load Impulse Character.
ATC = WorksheetFunction.Proper(Sheet3.Cells(9, 8).Value)       'Load Aquifer Thickness Character.
AvTp = WorksheetFunction.Proper(Sheet3.Cells(10, 8).Value)     'Load characteristic average type.
ThickCon = WorksheetFunction.Proper(Sheet3.Cells(11, 8).Value)  'Load Thickness Contribution...Update Type.
```

```
HTII = Sheet11.Cells(24, 5).Value                               'Load head time series output location i index.
HTIJ = Sheet11.Cells(25, 5).Value                               'Load head time series output location j index.
flaw = 1                                                        'Initialize flaw flag.
SEB = WorksheetFunction.Proper(Sheet12.Cells(12, 14).Value)     'Load Stream End Bank Option.

epse = Sheet11.Cells(10, 5).Value    'Load closure threshold exponent.
eps = 1 * 10 ^ (-epse)               'Set convergence threshold value.

msg = Sheet11.Cells(30, 5).Value      'Load any warning message.

If ITS = "Well Pumping" Then          'Set flow signs for practical conventions.
 qs = 1
 qrs = -1
 Else
 qs = -1
 qrs = 1
End If
If WorksheetFunction.Proper(Sheet3.Cells(8, 8).Value) = "Acre-Feet" Then   'Set Unit Conversion Factor.
 UCF = 43560
 Else
 UCF = 1
End If
If WorksheetFunction.Proper(Sheet4.Cells(4, 4).Value) = "A" Then  'Load Min Sat Thick Allow.
 MSTA = Sheet4.Cells(15, 5).Value
 Else:
 MSTA = 0
End If

PD = WorksheetFunction.Proper(Sheet11.Cells(14, 5).Value) 'Load time unit denomination.
u = Sheet11.Cells(15, 5).Value       'Load units per period.
nh = Sheet11.Cells(23, 5).Value      'Load period to plot heads.
If PD = "Days" Then                  'Set time step.
 dt = u / nc
 ElseIf PD = "Months" Then
 dt = u * 365.25 / 12 / nc
 ElseIf PD = "Hours" Then
 dt = u / nc / 24
End If
If ic = "Steady" Then
 deno = nc
 Else
 deno = 1
End If

If WorksheetFunction.Proper(Sheet3.Cells(13, 8).Value) = "Uniform" Then  'Load x-axis increment sizes for uniform case.
im = 1
Do While im < L + 1
 dx(im) = Sheet3.Cells(14, 8).Value
 im = im + 1
Loop
Else
im = 1
Do While im < L + 1
 dx(im) = Sheet3.Cells(25, 3 + im).Value   'Load x-axis increment sizes for discrete case.
  im = im + 1
Loop
```

```
End If

If WorksheetFunction.Proper(Sheet3.Cells(16, 8).Value) = "Uniform" Then  'Load y-axis increment sizes for uniform case.
jm = 1
Do While jm < m + 1
 dy(jm) = Sheet3.Cells(17, 8).Value
 jm = jm + 1
Loop
Else
jm = 1
Do While jm < m + 1
 dy(jm) = Sheet3.Cells(26 + jm, 2).Value   'Load y-axis increment sizes for discrete case.
 jm = jm + 1
Loop
End If

n = 1
Do While n < no + 1
 Qm(n) = Sheet11.Cells(38 + n, 4).Value  'Load impulse magnitude time series.
 If n = 1 Then
 IMPSMX = Qm(1)
 IMPSMN = Qm(1)
 End If
 If IMPSMX < Qm(n) Then
  IMPSMX = Qm(n)
 End If
 If IMPSMN > Qm(n) Then
  IMPSMN = Qm(n)
 End If
 n = n + 1
Loop

For im = 1 To L
For jm = 1 To m
   CT(jm, im) = Sheet3.Cells(26 + jm, 3 + im).Value
Next jm
Next im

 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1                          'Parameterization block.
    PermFact(jm, im) = 1
    If CT(jm, im) = 2 Then
     If dy(jm) > dx(im) Then
      lenp(jm, im) = dy(jm)                     'Load stream lengths (longer of cell dimensions).
      widp(jm, im) = dx(im)
      Lons(jm, im) = 1
      Else
      lenp(jm, im) = dx(im)
      widp(jm, im) = dy(jm)
      Lons(jm, im) = 2
     End If
    Call Lon12(Lon1, Lon2, ph, jm, im, CT(), L, m)
    If Lon1 <> Lon2 Then
```

```
    If Lon1 > Lon2 Then
      Lons(jm, im) = 1
       lenp(jm, im) = dy(jm)
       widp(jm, im) = dx(im)
     Else
      Lons(jm, im) = 2
       lenp(jm, im) = dx(im)
       widp(jm, im) = dy(jm)
    End If
   End If
  End If
If ATC = "Confined" Then
 If WorksheetFunction.Proper(Sheet4.Cells(8, 5).Value) = "Uniform" Then
  AqT(jm, im) = -1 * Sheet4.Cells(9, 5).Value
  Else
  AqT(jm, im) = -1 * Sheet4.Cells(23 + jm, 2 + im).Value
 End If
End If
 If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
  AqB(jm, im) = -1 * Sheet4.Cells(12, 5).Value
  Else
  AqB(jm, im) = -1 * Sheet4.Cells(82 + jm, 2 + im).Value
 End If
   If WorksheetFunction.Proper(Sheet5.Cells(8, 5).Value) = "Uniform" Then
     DEC(jm, im) = -1 * Sheet5.Cells(9, 5).Value
     Else
      If CT(jm, im) = 2 Then
       If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
         DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
        Else
         DEC(jm, im) = 0
       End If
       Else
         DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
      End If
   End If
 If WorksheetFunction.Proper(Sheet6.Cells(8, 6).Value) = "Uniform" Then
  k(jm, im) = Sheet6.Cells(9, 6).Value
  Else
  k(jm, im) = Sheet6.Cells(16 + jm, 2 + im).Value
 End If
 If Sheet9.Cells(7, 6).Value = 1 Then
  zone(jm, im) = 1
  Else
  zone(jm, im) = Sheet9.Cells(15 + jm, 2 + im).Value
 End If
 If WorksheetFunction.Proper(Sheet7.Cells(8, 7).Value) = "Uniform" Then
  Sy(jm, im) = Sheet7.Cells(9, 7).Value
  Else
  Sy(jm, im) = Sheet7.Cells(18 + jm, 2 + im).Value
 End If
 If WorksheetFunction.Proper(Sheet7.Cells(10, 7).Value) = "Uniform" Then
  Ss(jm, im) = Sheet7.Cells(11, 7).Value
  Else
  Ss(jm, im) = Sheet7.Cells(76 + jm, 2 + im).Value
 End If
```

```
  If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) = "Uniform" Then
   Hn(jm, im) = -1 * Sheet10.Cells(9, 5).Value
   Else
   Hn(jm, im) = -1 * Sheet10.Cells(16 + jm, 2 + im).Value
  End If
  Hno(jm, im) = Hn(jm, im)
  Hnp(jm, im) = Hn(jm, im)
If WorksheetFunction.Proper(Sheet4.Cells(11, 14).Value) <> "None" Then
 If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
  UMA(jm, im) = Sheet4.Cells(12, 14).Value                      'Load Ultimate Matric Suction in Aquifer.
  Else
  UMA(jm, im) = Sheet4.Cells(200 + jm, 2 + im).Value
 End If
End If
  If CT(jm, im) = 2 Then
     If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
      kpp(jm, im) = Sheet12.Cells(13, 6).Value
      bpp(jm, im) = Sheet12.Cells(14, 6).Value
      Else
      kpp(jm, im) = Sheet12.Cells(27 + jm, 2 + im).Value
      bpp(jm, im) = Sheet12.Cells(87 + jm, 2 + im).Value
     End If
     If WorksheetFunction.Proper(Sheet12.Cells(13, 14).Value) = "Equivalent" Then
      kppb(jm, im) = kpp(jm, im)
      bppb(jm, im) = bpp(jm, im)
     Else
     If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
      kppb(jm, im) = Sheet12.Cells(14, 14).Value
      bppb(jm, im) = Sheet12.Cells(15, 14).Value
      Else
      kppb(jm, im) = Sheet12.Cells(387 + jm, 2 + im).Value
      bppb(jm, im) = Sheet12.Cells(447 + jm, 2 + im).Value
     End If
     End If  'Equivalent.
    If SB <> "Permissive" Then
    If SIT <> "None" Then
    If WorksheetFunction.Proper(Sheet12.Cells(9, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       SHD(jm, im) = -1 * Sheet12.Cells(15, 6).Value
       Else
       SHD(jm, im) = -1 * Sheet12.Cells(147 + jm, 2 + im).Value
      End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
    If SIT = "None" Then
       Bed(jm, im) = SHD(jm, im)
       ElseIf SIT <> "None" Then
        If WorksheetFunction.Proper(Sheet12.Cells(10, 6).Value) = "Specified" Then
         If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
```

```
          Bed(jm, im) = -1 * Sheet12.Cells(16, 6).Value
          Else
          Bed(jm, im) = -1 * Sheet12.Cells(207 + jm, 2 + im).Value
         End If
         Else
         Bed(jm, im) = SHD(jm, im)
        End If
       End If
           If Bed(jm, im) > SHD(jm, im) Then
            Bed(jm, im) = SHD(jm, im)
            RSPOFM2 = 2
           End If
     If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       widpp(jm, im) = Sheet12.Cells(17, 6).Value
       Else
       widpp(jm, im) = Sheet12.Cells(267 + jm, 2 + im).Value
      End If
      Else
      widpp(jm, im) = widp(jm, im)
     End If
  If AqB(jm, im) >= Bed(jm, im) - bpp(jm, im) Then
  TipW(1) = 3
  End If
If SIT <> "None" Then
If WorksheetFunction.Proper(Sheet12.Cells(7, 14).Value) = "Simple" Then
If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
  UMS(jm, im) = Sheet12.Cells(8, 14).Value                     'Load Ultimate Matric Suction Below Bed.
  Else
  UMS(jm, im) = Sheet12.Cells(327 + jm, 2 + im).Value
End If
End If
 If UMA(jm, im) > UMS(jm, im) Then
   UMS(jm, im) = UMA(jm, im)
   msgu = "Matric Suction Value(s) Altered! "
 End If
End If
   If Bed(jm, im) - bpp(jm, im) - AqB(jm, im) <= UMS(jm, im) Then
      UMS(jm, im) = Bed(jm, im) - bpp(jm, im) - AqB(jm, im)
   End If
  If widpp(jm, im) = widp(jm, im) Then
     UMA(jm, im) = UMS(jm, im)
  End If
     If widpp(jm, im) > widp(jm, im) Then
      widpp(jm, im) = widp(jm, im)
      msgwp = "Stream Width Limited to Cell Width. "
     End If
  ElseIf CT(jm, im) <> 2 Then
 If CT(jm, im) = 1 Then
 If Right(IMS, 4) = "Head" Then
  If IMS = "Discrete, Head" Then
    Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
    ElseIf IMS = "Uniform, Head" Then
    Qd(jm, im) = 1
  End If
  ElseIf Right(IMS, 4) = "lume" Then
```

```
   If IMS = "Discrete, Volume" Then
    Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
    ElseIf IMS = "Uniform, Volume" Then
    Qd(jm, im) = 1
  End If
  End If
  If Qdmx = 0 Then
     Qdmx = Qd(jm, im)
     End If
     If Qdmn = 0 Then
     Qdmn = Qd(jm, im)
     End If
     If Qdmna = 0 Then
     Qdmna = Abs(Qd(jm, im))
     End If
     If Qd(jm, im) > Qdmx Then
     Qdmx = Qd(jm, im)
     End If
     If Qd(jm, im) < Qdmn Then
     Qdmn = Qd(jm, im)
     End If
     If Qd(jm, im) <> 0 Then
     If Abs(Qd(jm, im)) < Qdmna Then
     Qdmna = Abs(Qd(jm, im))
     End If
  End If
 End If
End If 'CT2.
  If CT(jm, im) <> 4 Then
  If DEC(jm, im) > 0 Then
  TipW(1) = 1
  End If
  If Hn(jm, im) > DEC(jm, im) Then
  TipW(2) = 1
  End If
  If ATC = "Confined" Then
   If AqT(jm, im) >= Hn(jm, im) Then
   TipW(3) = 1
   End If
   If AqB(jm, im) >= AqT(jm, im) Then
   TipW(4) = 1
   End If
  Else
   If AqB(jm, im) >= Hn(jm, im) Then
   TipW(5) = 1
   End If
  End If
  End If
   AqBi(i) = AqB(jm, im)              'Fill single index variables.
   AqTi(i) = AqT(jm, im)
   BEDi(i) = Bed(jm, im)
   CTi(i) = CT(jm, im)
   bppi(i) = bpp(jm, im)
   UMAi(i) = UMA(jm, im)
   UMSi(i) = UMS(jm, im)
   hx(i) = Hn(jm, im)
```

```
   i = i + 1
   im = im + 1
  Loop
  jm = jm + 1
 Loop

 If HTII > 0 Then
    HTS(0) = -1 * Hn(HTIJ, HTII)
 End If

 If Qdmx > 0 Then
  If Qdmn < 0 Then
   msgim = "Mixed Impulse. "
  End If
 End If

Call Neigh(CT(), L, m, NBRS(), widp(), widpp(), bppb())

Call OVPERer(m, L, jm, im, ovp(), ovw(), Bed(), beds(), widpp(), Lons(), widp(), bppb(), dx(), dy(), SIT, msgclt, NBRS(), SEB)
If Left(msgclt, 2) = "Ex" Then
 NegHead = 7
 NegHeadNM = 38
 flaw = 3
 GoTo Line50
End If

Call HoHo(hx(), m, L, jm, im, ovp(), ovw(), beds(), Bed(), hob(), ho(), widpp(), Lons(), widp(), bppb(), bpp(), dx(), dy(), _
          SIT, AqT(), AqB(), UMA(), UMS(), ATC, CT(), lenp(), SB, DEC())
Call Hoot(m, L, jm, im, ho(), hoo())

For jm = 1 To m
 For im = 1 To L
    If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
     S = SfuncC(CT(jm, im), SIT, ATC, (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), widp(jm, im), _
               lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
               SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
               ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
               ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
     If CT(jm, im) = 2 Then
     headsum2 = headsum2 + Hn(jm, im) * dx(im) * dy(jm) * S
     count2 = count2 + dx(im) * dy(jm) * S
     headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
     End If
     If CT(jm, im) Mod 2 = 1 Then
     headsum13 = headsum13 + Hn(jm, im) * dx(im) * dy(jm) * S
     count13 = count13 + dx(im) * dy(jm) * S
     End If
    Else
     If CT(jm, im) = 2 Then
     headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
     End If
   End If
 Next im
Next jm

If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
```

```
headave2 = headsum2 / count2
headave13 = headsum13 / count13
 If Abs(headave2 - headave13) > 0.000001 Then             'Check for initial head balance.
    RSPOFM = 2
 End If
End If
If Abs(headsum2s) > 0.000001 Then
   RSPOFM = 4
End If

'Establish and sort pressurization thresholds.
Call THstack(ovp(), ovw(), beds(), dx(), dy(), AqB(), _
             AqT(), Bed(), bpp(), UMS(), UMA(), ATC, _
             THRS(), THRCNT(), m, L, CT(), SIT, widp(), bppb(), widpp())

For jm = 1 To m        'Check that aquifer bottom beneath bed and banks.
For im = 1 To L
 If THRCNT(jm, im) > 0 Then
  If CT(jm, im) = 2 Then
     If AqB(jm, im) >= THRS(jm, im, THRCNT(jm, im)) Then
       TipW(1) = 3
     End If
   ElseIf CT(jm, im) <> 4 Then
     If AqB(jm, im) > THRS(jm, im, THRCNT(jm, im)) Then
       TipW(1) = 5
     End If
  End If
 End If
Next im
Next jm

If TipW(1) + TipW(2) + TipW(4) + TipW(5) = 0 Then                                  'Condition routine on valid thickness input, end if near end of
subroutine.

'
'TIME STEP COMPUTATION LOOP
'-----------------------------------------------------------------------
NegHead = 1

n = 1                              'Initialize time step counter.
o = 1
nm = 1
Do While n < nn + 1                'Loop for time steps!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

GTC = 0                            'Initialize codes for each time step.
kitro = 0
kitrtr = 0
TransT = 0
delta = 0
PFSflag = 0
MaxTerm = 0

 If n > 1 Then
 Qttp = Qtt
   For i = 1 To r
    Transpn(i) = Trans(i)
```

```
   Next i
 End If
ReDim Trans(1 To r)
ReDim transp(1 To r)
If o = 1 Then
 ReDim Transnm(1 To m, 1 To L)
End If

If o > 1 Then                                   'IMPULSE DISTRIBUTION BLOCK.+++++++++++++++
  If ic = "Instantaneous" Then
    ReDim Qc(1 To m, 1 To L)
    dH = 0
    Qtt = 0
    kitr = 0
    Qb = 0
  End If
    GoTo Line8
End If
If ic = "Steady" Then
   Qb = -1 * UCF * Qm(nm) / nc
   Qm(nm) = Qm(nm) / nc
  Else
   Qb = -1 * UCF * Qm(nm)
End If
'||||
dH = 0
Qf = 0
  IA = 0
  IAP = 0
    jm = 1
    Do While jm < m + 1                                 'TALLEY RELATIVE IMPULSE MAGNITUDE WEIGHTING.
      im = 1
      Do While im < L + 1
        If CT(jm, im) = 1 Then                            'CT if - impulse cells.
         If Right(IMS, 4) = "lume" Then                       'IMS if - Volumetric impulse distribution case.
          IA = IA + Qd(jm, im)
          If Qd(jm, im) > 0 Then                          'Alternate denominator for zero net impulse case, by volume.
            IAP = IAP + Qd(jm, im)
          End If                                          'Denominator end if.
          ElseIf Right(IMS, 4) = "Head" Then                  'IMS ElseIf.
          S = SfuncC(CT(jm, im), SIT, ATC, (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm,
im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
                    SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                    ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                    ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
          IA = IA + (Qd(jm, im) * dx(im) * dy(jm) * S)
          If Qd(jm, im) > 0 Then                          'Alternate denominator for zero net impulse case.
            IAP = IAP + (Qd(jm, im) * dx(im) * dy(jm) * S)
          End If                                          'Denominator end if.
        End If                                                'IMS end if.
        End If                                            'CT end if
        im = im + 1                                       'Increment to loop.
      Loop
      jm = jm + 1
    Loop
If IMS = "Uniform, Head" Then                   'IMS if.++++++++++++++++++++++++++++++++
```

```
    Qf = UCF * qs * Qm(nm)
    dH = -1 * Qf * Qdmna / IA
     GTC = 1
     GoTo Line4
Line1:   '=========================
ElseIf IMS = "Uniform, Volume" Then           'IMS elseif.+++++++++++++++++++++++++++++++
   jm = 1                                            'Initialize loops, separate to allow prior loops to total denominator, AI.
   Do While jm < m + 1
   im = 1                                          'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   Do While im < L + 1
   If CT(jm, im) = 1 Then                          'CT if - only impulse cells.
    Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IA 'Fill impulse volume.
   End If                                          'CT end if
   im = im + 1                                     'Increment to loop.
   Loop
   jm = jm + 1
  Loop
ElseIf IMS = "Discrete, Head" Then            'IMS elseif.+++++++++++++++++++++++++++++++
  If msgim = "Mixed Impulse. " Then
     msgti = "Discrete Impulse Distribution by Head Requires Like Signs Throughout Grid. Distribute by Volume Instead. "
     GoTo Line50
  End If
    If IA = 0 Then
     msgti = "Discrete Impulse Distribution by Head Requires Non-zero Weighting. "
     GoTo Line50
    End If
    If IA > 0 Then                                    'IA If.
         Qf = UCF * qs * Qm(nm)
         dH = -1 * Qf * Qdmna / IA
       ElseIf IA < 0 Then                             'IA ElseIf.
         Qf = UCF * qs * Qm(nm) * (-1)
         dH = -1 * Qf * Qdmna / IA
    End If                                            'IA end if.
     GTC = 2
     GoTo Line4
Line2:   '=========================
ElseIf IMS = "Discrete, Volume" Then          'IMS elseif.+++++++++++++++++++++++++++++++
   jm = 1                                             'Initialize loops, separate to allow prior loops to total denominator, AI.
   Do While jm < m + 1
   im = 1                                          'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   Do While im < L + 1
   If CT(jm, im) = 1 Then                          'CT if - only impulse cells.
    If IA <> 0 Then                                'IA If.
    Qc(jm, im) = UCF * qs * Qm(nm) * Abs(Qd(jm, im)) / IA 'Fill impulse volume.
    Else
      If IAP > 0 Then                                'IAP if.
      msgim2 = "10"               'Zero NET impulse case.
      Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IAP
      Else
      msgim2 = "11"                     'Zero impulse case.
      GoTo Line50
      End If                                          'End IAP end if.
    End If                                            'IA end if.
   End If                                          'CT end if
   im = im + 1                                     'Increment to loop.
   Loop
```

```
   jm = jm + 1
  Loop
End If                                          'IMS end if.
'||||
  If ic = "Instantaneous" Then                  'ic if.++++++++++++++++++++++++++++++++++
  Qtt = 0
  i = 1
  jm = 1                                          'Initialize loops, separate to allow prior loops to total denominator, AI.
  Do While jm < m + 1
   im = 1                                        'CONVERT INSTANTANEOUS IMPULSE TO HEAD.
   Do While im < L + 1
   If CT(jm, im) = 1 Then                        'CT if - only impulse cells.
         Qtt = Qtt + Qc(jm, im)                  'For Domain Volumetric Budget Later.
         S = SfuncC(CT(jm, im), SIT, ATC, (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm,
im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
                  SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                  ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                  ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
         If THRCNT(jm, im) > 0 Then 'Confinement threshold present.
          If Qc(jm, im) >= 0 Then 'H falling.
            For z = 1 To THRCNT(jm, im)
             If (Hn(jm, im) + UMA(jm, im)) >= THRS(jm, im, z) Then
              If (Hn(jm, im) + UMA(jm, im)) - Qc(jm, im) / (dx(im) * dy(jm) * S) < THRS(jm, im, z) Then  'Depressurization.
                 Hn(jm, im) = HnDepres(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), _
                                 Qc(jm, im), Hn(jm, im), dx(im), dy(jm), Ss(jm, im), Sy(jm, im), AqT(jm, im), AqB(jm, im), _
                                 CT(jm, im), SIT, ATC, ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                 beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                 ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, "A", widp(jm, im), lenp(jm, im), widpp(jm, im), _
                                 Bed(jm, im), bpp(jm, im), bppb(jm, im), SB, UMA(jm, im), UMS(jm, im))
                     If msgti = "" Then
                     msgti = "Impulse Distribution Through Confinement Transition. "
                     End If
                  Exit For
                Else  'Hn...
                  Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * S) 'Pressurized.
                  Exit For
               End If  'Hn...
             End If
             If z = THRCNT(jm, im) Then
              Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * S) 'Depressurized below lowest threshold.
             End If
            Next z
           ElseIf Qc(jm, im) < 0 Then 'H rising.
            For z = THRCNT(jm, im) To 1 Step -1
             If (Hn(jm, im) + UMA(jm, im)) < THRS(jm, im, z) Then
              If (Hn(jm, im) + UMA(jm, im)) - Qc(jm, im) / (dx(im) * dy(jm) * S) >= THRS(jm, im, z) Then 'Pressurization.
                 Hn(jm, im) = HnPres(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), _
                                 Qc(jm, im), Hn(jm, im), dx(im), dy(jm), Ss(jm, im), Sy(jm, im), AqT(jm, im), AqB(jm, im), _
                                 CT(jm, im), SIT, ATC, ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                 beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                 ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, "A", widp(jm, im), lenp(jm, im), widpp(jm, im), _
                                 Bed(jm, im), bpp(jm, im), bppb(jm, im), SB, UMA(jm, im), UMS(jm, im))
                     If msgti = "" Then
                     msgti = "Impulse Distribution Through Confinement Transition. "
                     End If
                 Exit For
```

```
                Else  'Hn-Qc...
                  Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * S) 'Depressurized.
                  Exit For
                End If  'Hn-Qc...
              End If
              If z = 1 Then
               Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * S) 'Pressurized Above highest threshold.
              End If
            Next z
          End If 'Qc
         ElseIf THRCNT(jm, im) = 0 Then 'No confinement threshold present.
          Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * S)
        End If                          'Confinement threshold end if.
        If Hn(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
         If ATC = "Unconfined" Then
          GoTo Line3
         Else
          If ITS = "Deep Percolation" Then
Line3:   '=========================
             msgc = "Inundated Ground! "
             msgti = "Impulse Exceeds Aquifer Capacity. "
             flaw = 3
             NegHead = 6
             NegHeadNM = nm + 38
             GoTo Line50
          End If
         End If
        End If
        Qc(jm, im) = 0                              'Annul converted flow (instantaneous).
      End If                                        'CT end if.
      i = i + 1
      im = im + 1                                       'Increment to loop.
      Loop
    jm = jm + 1
    Loop
    ElseIf ic = "Steady" Then                   'ic elseif.++++++++++++++++++++++++++++++
    Qtt = 0
    For jm = 1 To m                                  'CONVERT STEADY IMPULSE TO FLOW.
    For im = 1 To L
      If CT(jm, im) = 1 Then
        Qtt = Qtt + Qc(jm, im)                      'For Domain Volumetric Budget Later.
        Qc(jm, im) = Qc(jm, im) / (dt)              'Convert to volume per time.
      End If
   Next im
   Next jm
  End If                                        'ic end if.
'||||
If ic = "Steady" Then                           'Restore Qm for later use.
    Qm(nm) = Qm(nm) * nc
End If
GoTo Line8                                      'Direct past solver.++++++++++++++++++++++
'||||>
Line4:          '=========================
'START NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION OVER GRID BY HEAD]]]]]]]]]]
   kitr = 0                                      'Initialize iteration counter.
   IGTC = 0                                      'Initialize impulse go to code.
```

```
   ObF = 1                                      'Initialize objective function.
   Do While Abs(ObF) > 0.0000001        'Solver loop.
   If kitr < 500 Then                           'Iteration limit check
     kitr = kitr + 1                            'Increment iteration counter.
Line5:          '==========================
   Qt = 0                                       'Initialize virtual volume tally.
   '((((>
   i = 1
   jm = 1                                       'Prepare for calculation loop.
   Do While jm < m + 1
   im = 1
   Do While im < L + 1                          'Calculation loop.
   If CT(jm, im) = 1 Then                       'CT if - cell type check.
    S = SfuncC(CT(jm, im), SIT, ATC, (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im),
bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
              SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
              ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
              ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
    If THRCNT(jm, im) > 0 Then 'Confinement threshold present.
           If ic = "Steady" Then                'ic if.
              Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * S * dx(im) * dy(jm)
           Else                                  'ic else - instantaneous.
           If (Qd(jm, im) * dH / Qdmna) <= 0 Then 'H falling.
            For z = 1 To THRCNT(jm, im)
             If (Hn(jm, im) + UMA(jm, im)) >= THRS(jm, im, z) Then
              If (Hn(jm, im) + UMA(jm, im)) + (Qd(jm, im) * dH / Qdmna) < THRS(jm, im, z) Then 'Depressurization.
                 Qc(jm, im) = QcDepres(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), _
                                  Hn(jm, im), dx(im), dy(jm), Ss(jm, im), Sy(jm, im), AqT(jm, im), AqB(jm, im), _
                                  CT(jm, im), SIT, ATC, ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                  beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                  ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, Qd(jm, im), dH, Qdmna, "A", widp(jm, im), lenp(jm, im),
widpp(jm, im), _
                                  Bed(jm, im), bpp(jm, im), bppb(jm, im), SB, UMA(jm, im), UMS(jm, im))
                Exit For
               Else  'Hn-Qc...
                 Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * S * dx(im) * dy(jm) 'Pressurized.
                 Exit For
              End If  'Hn-Qc...
             End If
             If z = THRCNT(jm, im) Then
              Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * S * dx(im) * dy(jm) 'Depressurized below lowest threshold.
             End If
            Next z
           ElseIf (Qd(jm, im) * dH / Qdmna) > 0 Then 'H rising.
            For z = THRCNT(jm, im) To 1 Step -1
             If (Hn(jm, im) + UMA(jm, im)) < THRS(jm, im, z) Then
              If (Hn(jm, im) + UMA(jm, im)) + (Qd(jm, im) * dH / Qdmna) >= THRS(jm, im, z) Then 'Pressurization.
                 Qc(jm, im) = QcPres(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), _
                                  Hn(jm, im), dx(im), dy(jm), Ss(jm, im), Sy(jm, im), AqT(jm, im), AqB(jm, im), _
                                  CT(jm, im), SIT, ATC, ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                  beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                  ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, Qd(jm, im), dH, Qdmna, "A", widp(jm, im), lenp(jm, im),
widpp(jm, im), _
                                  Bed(jm, im), bpp(jm, im), bppb(jm, im), SB, UMA(jm, im), UMS(jm, im))
                Exit For
               Else  'Hn-Qc...
```

```
                Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * S * dx(im) * dy(jm) 'Depressurized.
                Exit For
              End If  'Hn-Qc...
             End If
             If z = 1 Then
              Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * S * dx(im) * dy(jm) 'Pressurized Above highest threshold.
             End If
            Next z
          End If 'H rise or fall.
          End If                                'ic end if.
        ElseIf THRCNT(jm, im) = 0 Then 'No confinement threshold present.
         Qc(jm, im) = -(Qd(jm, im) * dH / Qdmna) * S * dx(im) * dy(jm)
    End If                                       'Confinement threshold end if.
    If IA <> 0 Then
    Qt = Qt + Qc(jm, im)                        'For distribution by Newton Solver.
    Else
    If Qd(jm, im) > 0 Then
    Qt = Qt + Qc(jm, im)
    End If
    End If
  End If                                       'CT end if.
   i = i + 1
   im = im + 1
   Loop
   jm = jm + 1
   Loop                                         'Close calculation loop.
   '((((>
   If IGTC = 6 Then
   IGTC = 0
   GoTo Line6
   End If
   '((((>
   ObF = Qf - Qt                                'Compute objective function [F(n)].
  If Abs(ObF) <= 0.0000001 Then
  GoTo Line7
  End If
  dH = (1 + 0.00000001) * dH                    'Increment head increment.
   IGTC = 6
   GoTo Line5
Line6:   '==========================
  ObFF = Qf - Qt                               'Recompute objective function [F(n+1)].
  dFdu = (ObFF - ObF) / (dH - dH / (1 + 0.00000001)) 'Calculate derivative.
  dH = dH / (1 + 0.00000001)                   'Restore head increment.
  dH = dH - ObF / dFdu                         'Compute next estimate for head increment.
Line7:   '==========================
  Else                                                 'Intervene to end iteration if not converging in a reasonable number of cycles.
   NegHead = 4
   NegHeadNM = nm + 38
   GoTo Line50
  End If                                               'Iteration counter end if.
  Loop                                         'Close solver loop.
'END NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION]]]]]]]]]]
If GTC = 1 Then
 GoTo Line1
ElseIf GTC = 2 Then
 GoTo Line2
```

```
End If                                  'END IMPULSE DISTRIBUTION BLOCK.++++++++++

Line8: '==========================

i = 1                                                 'Re-initialize head vectors.
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1
If CT(jm, im) <> 4 Then
  If Hn(jm, im) <= AqB(jm, im) Then                                'Check that head not beneath aquifer.
    NegHead = 2
    NegHeadNM = nm + 38
    GoTo Line50
   Else
  End If
End If
Hni(i) = Hn(jm, im)                                               'Fill vectors.
hx(i) = Hn(jm, im)
    im = im + 1                                                   'Increment to loop.
    i = i + 1
  Loop
  jm = jm + 1
 Loop                                                             'Thickness update.
 Call HoHo(hx(), m, L, jm, im, ovp(), ovw(), beds(), Bed(), hob(), ho(), widpp(), Lons(), widp(), bppb(), bpp(), dx(), dy(), _
          SIT, AqT(), AqB(), UMA(), UMS(), ATC, CT(), lenp(), SB, DEC())

Line9: '==========================       'Line label for point of return, if pressurization transition in first set of iterations, and for outside
iteration.
'
'FILL PRIMARY COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------------------------------
ReDim AA(1 To m, 1 To L), BB(1 To m, 1 To L), CC(1 To m, 1 To L)  'Redimension A, B, C to reset all to empty.
ReDim DD(1 To m, 1 To L), EE(1 To m, 1 To L), b(1 To m, 1 To L)   'Redimension D, E, b to reset all to empty.
ReDim side11(1 To m, 1 To L), side21(1 To m, 1 To L), side31(1 To m, 1 To L), side41(1 To m, 1 To L)
ReDim side12(1 To m, 1 To L), side22(1 To m, 1 To L), side32(1 To m, 1 To L), side42(1 To m, 1 To L)
ReDim csea(1 To m, 1 To L, 1 To 4)
ReDim csba(1 To m, 1 To L, 1 To 4)
ReDim SidMat(1 To m, 1 To L, 1 To 4)
ReDim BedMat(1 To m, 1 To L)
 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1                        'Calculate coefficient values.
   If CT(jm, im) <> 4 Then
    hxa = 9999997
    hxb = 9999997
    hxc = 9999997
    hxd = 9999997
    If jm - 1 > 0 Then
    If CT(jm - 1, im) < 4 Then
      AA(jm, im) = khbarC(AvTp, k(jm - 1, im), hob(jm - 1, im, 3), dy(jm - 1), _
                              k(jm, im), hob(jm, im, 1), dy(jm), dx(im))
      If CT(jm - 1, im) <> 2 Then
       hxa = hx(i - L)
```

```
       Else
       hxa = 9999992
      End If
     End If
    End If
    If im - 1 > 0 Then
     If CT(jm, im - 1) < 4 Then
       BB(jm, im) = khbarC(AvTp, k(jm, im - 1), hob(jm, im - 1, 4), dx(im - 1), _
                                 k(jm, im), hob(jm, im, 2), dx(im), dy(jm))
       If CT(jm, im - 1) <> 2 Then
        hxb = hx(i - 1)
        Else
        hxb = 9999992
       End If
     End If
    End If
    If jm + 1 < m + 1 Then
     If CT(jm + 1, im) < 4 Then
       CC(jm, im) = khbarC(AvTp, k(jm + 1, im), hob(jm + 1, im, 1), dy(jm + 1), _
                                 k(jm, im), hob(jm, im, 3), dy(jm), dx(im))
       If CT(jm + 1, im) <> 2 Then
        hxc = hx(i + L)
        Else
        hxc = 9999992
       End If
     End If
    End If
    If im + 1 < L + 1 Then
     If CT(jm, im + 1) < 4 Then
       DD(jm, im) = khbarC(AvTp, k(jm, im + 1), hob(jm, im + 1, 2), dx(im + 1), _
                                 k(jm, im), hob(jm, im, 4), dx(im), dy(jm))
       If CT(jm, im + 1) <> 2 Then
        hxd = hx(i + 1)
        Else
        hxd = 9999992
       End If
      End If
     End If
   End If
   EE(jm, im) = EE(jm, im) + FuncEC(AA(jm, im), BB(jm, im), CC(jm, im), DD(jm, im), Trans(i), dx(im), dy(jm), dt, Hn(jm, im), hx(i), CT(jm, im), SIT,
_
                 ATC, AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm,
im), SB, UMA(jm, im), UMS(jm, im), _
                 ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), _
                 beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4),
_
                 THRS(jm, im, 5), THRCNT(jm, im))
   b(jm, im) = b(jm, im) + FuncBC(Qc(jm, im), Trans(i), dx(im), dy(jm), dt, Hn(jm, im), hx(i), CT(jm, im), SIT, _
                 ATC, AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm,
im), SB, UMA(jm, im), UMS(jm, im), _
                 ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), _
                 beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4),
_
                 THRS(jm, im, 5), THRCNT(jm, im))
    If CT(jm, im) = 2 Then
     If hx(i) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
```

```
      EE(jm, im) = EE(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) / bpp(jm, im)                          'Includes full bed flux term.
      b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * SHD(jm, im) / bpp(jm, im)              'Includes full bed flux term.
     ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) / bpp(jm, im)
'Bed flux limited (drawn down below bottom).
    End If
      side11(jm, im) = sidefunc1c(NBRS(jm, im, 1), Bed(jm, im), SHD(jm, im), hx(i), hxa, dx(im), Lons(jm, im), 2, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB) '2 is code for dx side.
      side12(jm, im) = sidefunc2c(NBRS(jm, im, 1), Bed(jm, im), SHD(jm, im), hx(i), hxa, dx(im), Lons(jm, im), 2, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB)
      side21(jm, im) = sidefunc1c(NBRS(jm, im, 2), Bed(jm, im), SHD(jm, im), hx(i), hxb, dy(jm), Lons(jm, im), 1, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB)
      side22(jm, im) = sidefunc2c(NBRS(jm, im, 2), Bed(jm, im), SHD(jm, im), hx(i), hxb, dy(jm), Lons(jm, im), 1, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB)
      side31(jm, im) = sidefunc1c(NBRS(jm, im, 3), Bed(jm, im), SHD(jm, im), hx(i), hxc, dx(im), Lons(jm, im), 2, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB)
      side32(jm, im) = sidefunc2c(NBRS(jm, im, 3), Bed(jm, im), SHD(jm, im), hx(i), hxc, dx(im), Lons(jm, im), 2, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB)
      side41(jm, im) = sidefunc1c(NBRS(jm, im, 4), Bed(jm, im), SHD(jm, im), hx(i), hxd, dy(jm), Lons(jm, im), 1, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB)
      side42(jm, im) = sidefunc2c(NBRS(jm, im, 4), Bed(jm, im), SHD(jm, im), hx(i), hxd, dy(jm), Lons(jm, im), 1, widpp(jm, im), lenp(jm, im),
widp(jm, im), bppb(jm, im), SEB)
     If side11(jm, im) + side12(jm, im) > 0 Then
     If Lons(jm, im) = 2 Then
      If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
        SidMat(jm, im, 1) = sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i))
          csea(jm, im, 1) = sideE(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side11(jm, im), side12(jm, im))
          csba(jm, im, 1) = sideB(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side11(jm, im), side12(jm, im))
        EE(jm, im) = EE(jm, im) + csea(jm, im, 1)
        b(jm, im) = b(jm, im) - csba(jm, im, 1)
       ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
        GoTo Line9a1
      End If
      ElseIf Lons(jm, im) = 1 Then
Line9a1:     '==========================
        SidMat(jm, im, 1) = sThresh(UMA(jm - 1, im), Bed(jm, im), SHD(jm, im), hxa)
          csea(jm, im, 1) = sideE(Bed(jm, im), UMA(jm - 1, im), SHD(jm, im), hxa, kppb(jm, im), bppb(jm, im), side11(jm, im), side12(jm, im))
          csba(jm, im, 1) = sideB(Bed(jm, im), UMA(jm - 1, im), SHD(jm, im), hxa, kppb(jm, im), bppb(jm, im), side11(jm, im), side12(jm, im))
        EE(jm - 1, im) = EE(jm - 1, im) + csea(jm, im, 1)
        b(jm - 1, im) = b(jm - 1, im) - csba(jm, im, 1)
      End If 'Lons
      End If
     If side21(jm, im) + side22(jm, im) > 0 Then
     If Lons(jm, im) = 1 Then
      If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
        SidMat(jm, im, 2) = sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i))
          csea(jm, im, 2) = sideE(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side21(jm, im), side22(jm, im))
          csba(jm, im, 2) = sideB(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side21(jm, im), side22(jm, im))
        EE(jm, im) = EE(jm, im) + csea(jm, im, 2)
        b(jm, im) = b(jm, im) - csba(jm, im, 2)
        ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
        GoTo Line9a2
      End If
      ElseIf Lons(jm, im) = 2 Then
Line9a2:     '==========================
        SidMat(jm, im, 2) = sThresh(UMA(jm, im - 1), Bed(jm, im), SHD(jm, im), hxb)
```

```
            csea(jm, im, 2) = sideE(Bed(jm, im), UMA(jm, im - 1), SHD(jm, im), hxb, kppb(jm, im), bppb(jm, im), side21(jm, im), side22(jm, im))
            csba(jm, im, 2) = sideB(Bed(jm, im), UMA(jm, im - 1), SHD(jm, im), hxb, kppb(jm, im), bppb(jm, im), side21(jm, im), side22(jm, im))
          EE(jm, im - 1) = EE(jm, im - 1) + csea(jm, im, 2)
          b(jm, im - 1) = b(jm, im - 1) - csba(jm, im, 2)
         End If 'Lons
         End If
       If side31(jm, im) + side32(jm, im) > 0 Then
       If Lons(jm, im) = 2 Then
        If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
          SidMat(jm, im, 3) = sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i))
            csea(jm, im, 3) = sideE(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side31(jm, im), side32(jm, im))
            csba(jm, im, 3) = sideB(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side31(jm, im), side32(jm, im))
          EE(jm, im) = EE(jm, im) + csea(jm, im, 3)
          b(jm, im) = b(jm, im) - csba(jm, im, 3)
         ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
          GoTo Line9a3
        End If
        ElseIf Lons(jm, im) = 1 Then
Line9a3:     '=========================
          SidMat(jm, im, 3) = sThresh(UMA(jm + 1, im), Bed(jm, im), SHD(jm, im), hxc)
            csea(jm, im, 3) = sideE(Bed(jm, im), UMA(jm + 1, im), SHD(jm, im), hxc, kppb(jm, im), bppb(jm, im), side31(jm, im), side32(jm, im))
            csba(jm, im, 3) = sideB(Bed(jm, im), UMA(jm + 1, im), SHD(jm, im), hxc, kppb(jm, im), bppb(jm, im), side31(jm, im), side32(jm, im))
          EE(jm + 1, im) = EE(jm + 1, im) + csea(jm, im, 3)
          b(jm + 1, im) = b(jm + 1, im) - csba(jm, im, 3)
          End If 'Lons
         End If
       If side41(jm, im) + side42(jm, im) > 0 Then
       If Lons(jm, im) = 1 Then
        If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
           SidMat(jm, im, 4) = sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i))
            csea(jm, im, 4) = sideE(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side41(jm, im), side42(jm, im))
            csba(jm, im, 4) = sideB(Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side41(jm, im), side42(jm, im))
          EE(jm, im) = EE(jm, im) + csea(jm, im, 4)
          b(jm, im) = b(jm, im) - csba(jm, im, 4)
         ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
          GoTo Line9a4
        End If
        ElseIf Lons(jm, im) = 2 Then
Line9a4:     '=========================
          SidMat(jm, im, 4) = sThresh(UMA(jm, im + 1), Bed(jm, im), SHD(jm, im), hxd)
            csea(jm, im, 4) = sideE(Bed(jm, im), UMA(jm, im + 1), SHD(jm, im), hxd, kppb(jm, im), bppb(jm, im), side41(jm, im), side42(jm, im))
            csba(jm, im, 4) = sideB(Bed(jm, im), UMA(jm, im + 1), SHD(jm, im), hxd, kppb(jm, im), bppb(jm, im), side41(jm, im), side42(jm, im))
          EE(jm, im + 1) = EE(jm, im + 1) + csea(jm, im, 4)
          b(jm, im + 1) = b(jm, im + 1) - csba(jm, im, 4)
         End If 'Lons
        End If
     End If
    i = i + 1
    im = im + 1
   Loop
   jm = jm + 1
 Loop

 i = 1
 jm = 1
 Do While jm < m + 1
```

```
   im = 1
   Do While im < L + 1
   If CT(jm, im) = 4 Then
    GoTo line9e
   ElseIf CT(jm, im) = 2 Then
   If SB = "Permissive" Then
line9e:      '=========================
    AA(jm, im) = 0
    BB(jm, im) = 0
    CC(jm, im) = 0
    DD(jm, im) = 0
    EE(jm, im) = PermFact(jm, im) * dx(im) * dy(jm) / dt                    'Factor scales coefficients for procedure stability without affecting
result.
    b(jm, im) = -EE(jm, im) * Hnp(jm, im)
   End If
   End If
    If PFSflag < 1 Then
     If MaxTerm < Abs(EE(jm, im)) Then
      MaxTerm = Abs(EE(jm, im))
     End If
    End If
    i = i + 1
    im = im + 1
   Loop
   jm = jm + 1
 Loop

If PFSflag < 1 Then
   PFSflag = 1
End If

'
'WRITE AUGMENTED MATRIX BY POPULATING COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------------------
ReDim SE(1 To r, 1 To 5), SEC(1 To r), UM(1 To r, 1 To 3), UT(1 To r, 1 To 3), DU(1 To r, 1 To 3)       'Redimension matrices to obviate memory
errors(row limit, column limit).
i = 1                              'Initialize system of equations row index, also index for model cell identification serial.
For jm = 1 To m                    'Initialize model space row index.  Note: convention for matrix indeces is different than model.  Model columns
are im, Matrix columns are j.
  For im = 1 To L                          'Initialize and reset model space column index.

   SE(i, 3) = -EE(jm, im)         'Coefficient for cell's own next-step head.
   SEC(i) = b(jm, im)             'Constant or intercept value for row.
   If m > 1 Then
   If L > 1 Then
   If jm + 1 < m + 1 Then
    SE(i, 5) = CC(jm, im)           'Coefficient for next-step head in model cell beneath, matrix place L to the right.
   End If
   If jm - 1 > 0 Then
    SE(i, 1) = AA(jm, im)           'Coefficient for next-step head in model cell above, matrix place L to the left.
   End If
   End If
   End If
   If L > 1 Then
    If im + 1 < L + 1 Then
     SE(i, 4) = DD(jm, im)           'Coefficient for next-step head in model cell to right, matrix place one to the right.
```

```
    End If
    If im - 1 > 0 Then
     SE(i, 2) = BB(jm, im)           'Coefficient for next-step head in model cell to the left, matrix place one to the left.
    End If
   Else
    If jm + 1 < m + 1 Then
     SE(i, 4) = CC(jm, im)           'Coefficient for next-step head in model cell beneath, matrix place one to the right.
    End If
    If jm - 1 > 0 Then
     SE(i, 2) = AA(jm, im)           'Coefficient for next-step head in model cell above, matrix place one to the left.
    End If
   End If

   SE(i, 1) = -1 * SE(i, 1)
   SE(i, 2) = -1 * SE(i, 2)
   SE(i, 3) = -1 * SE(i, 3)
   SE(i, 4) = -1 * SE(i, 4)
   SE(i, 5) = -1 * SE(i, 5)
   SEC(i) = -1 * SEC(i)

     If CT(jm, im) = 4 Then
      GoTo Line9f
     ElseIf CT(jm, im) = 2 Then
     If SB = "Permissive" Then
Line9f:      '=========================
      If PFSflag < 2 Then
       If Abs(SE(i, 3)) < 0.75 * MaxTerm Then
        PermFact(jm, im) = PermFact(jm, im) * WorksheetFunction.RoundUp(0.75 * MaxTerm / SE(i, 3), 0)  'Increment factor to recompute coefficient for
mathematical stability.
        subsflag = 1
       End If
      End If
     End If
     End If

   i = i + 1
  Next im
Next jm
    If subsflag = 1 Then
     PFSflag = 2
     subsflag = 0
     GoTo Line9
    End If

Line9g:      '=========================
 For i = 1 To r
  UM(i, 1) = (1 + delta) * SE(i, 3)            'Modified incomplete Cholesky factorization upper conditioner matrix.
 Next i                                             'Initial loop sets all uii values to aii, unless delta non-zero.
 For i = 1 To r
  If i - 1 > 0 Then
   UM(i, 1) = UM(i, 1) - (SE(i - 1, 4) ^ 2) / UM(i - 1, 1)                                  'Xi, Hill pg. 10.
   UM(i, 1) = UM(i, 1) - SE(i - 1, 4) * SE(i - 1, 5) / UM(i - 1, 1)                         'Phi.
     If i + L - 1 <= r Then
       UM(i + L - 1, 1) = UM(i + L - 1, 1) - SE(i - 1, 4) * SE(i - 1, 5) / UM(i - 1, 1)
     End If
   If i - L > 0 Then
```

```
    UM(i, 1) = UM(i, 1) - (SE(i - L, 5) ^ 2) / UM(i - L, 1)                                  'Tau.
   End If
  End If
  If UM(i, 1) < 0.001 Then
   If delta < 1000000 Then
     delta = 1.5 * delta + 0.001
     GoTo Line9g
   End If
  End If
  If i < r Then
   UM(i, 2) = SE(i, 4)
  End If
  If i <= r - L Then
   UM(i, 3) = SE(i, 5)
  End If
  DU(i, 1) = 1 'UM(i, 1) * (1 / UM(i, 1)) 'Diagonal matrix of 1/UM(i,1) times UM.
  DU(i, 2) = UM(i, 2) / UM(i, 1)
  DU(i, 3) = UM(i, 3) / UM(i, 1)
 Next i

For i = 1 To r                          'Transpose of UM carried out for condensed format.
UT(i, 3) = UM(i, 1)
If i - 1 > 0 Then
UT(i, 2) = UM(i - 1, 2)
End If
If i - L > 0 Then
UT(i, 1) = UM(i - L, 3)
End If
Next i

'SIMULTANEOUS SOLUTION OF SYSTEM OF EQUATIONS
'-----------------------------------------------------------------------------------------
'Modified Incomplete Cholesky Preconditioned CONJUGATE GRADIENT METHOD (MICCG)!!
'SEE "Preconditioned Conjugate-Gradient 2 (PCG2), A Computer Program for Solving Ground-Water flow Equations"
'by Mary C. Hill, USGS Water Resources Investigations Report 90-4048. 1990.
Line10: '==========================       'Line label for top of solver.
ReDim Preserve SE(1 To r, 1 To 5), SEC(1 To r), hx(1 To r)
kitri = 0                      'Initialize iteration counter.
maxerrp = 1E+99                'Initialize prior iteration closure.
maxerr = 1                     'Initialize iteration closure.
GTCR = 0                       'Initialize GoTo Code for Residual Rounding.
Line11: '==========================
'Initialize Ax, rr, zz.
'-----------------quick homemade array multiplication
ReDim Ax(1 To r)
For i = 1 To r
Ax(i) = SE(i, 3) * hx(i)
If m > 1 Then
If L > 1 Then
  If i - L > 0 Then
   Ax(i) = Ax(i) + SE(i, 1) * hx(i - L)
  End If
  If i + L < r + 1 Then
   Ax(i) = Ax(i) + SE(i, 5) * hx(i + L)
  End If
End If
```

```
End If
 If i - 1 > 0 Then
  Ax(i) = Ax(i) + SE(i, 2) * hx(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 4) * hx(i + 1)
 End If
Next i
'-----------------
ReDim Preserve Ax(1 To r)
ReDim rr(1 To r)
For i = 1 To r          'method step
rr(i) = SEC(i) - Ax(i)
Next i
          If GTCR = 2 Then
            GoTo Line14
          End If
Do While maxerr > eps                       '##############Top of Method loop.
If kitri < 1000 Then                          'Iteration limit check.
Line11b: '==========================
ReDim Preserve UT(1 To r, 1 To 3), rr(1 To r)
ReDim vv(1 To r)
For i = 1 To r          'forward sub
vv(i) = rr(i)
If L > 1 Then
 If m > 1 Then
  If i - L > 0 Then
   vv(i) = vv(i) - UT(i, 1) * vv(i - L)
  End If
 End If
End If
If i - 1 > 0 Then
vv(i) = vv(i) - UT(i, 2) * vv(i - 1)
End If
vv(i) = vv(i) / UT(i, 3)
Next i
ReDim Preserve DU(1 To r, 1 To 3), vv(1 To r)
ReDim zz(1 To r)
i = r
Do While i > 0           'back sub
zz(i) = vv(i)
If L > 1 Then
 If m > 1 Then
  If i + L < r + 1 Then
    zz(i) = zz(i) - DU(i, 3) * zz(i + L)
  End If
 End If
End If
If i + 1 < r + 1 Then
zz(i) = zz(i) - DU(i, 2) * zz(i + 1)
End If
'zz(i) = zz(i) / DU(i, 1)  DU(i,1)=1, so no need to divide.
i = i - 1
Loop
ReDim Preserve zz(1 To r)
          If GTCR = 1 Then
```

```
           GoTo Line12
         End If
If kitri = 0 Then
Line12: '=========================
 ReDim pp(1 To r)
 nip = 0
 For i = 1 To r
 pp(i) = zz(i)
 nip = nip + zz(i) * rr(i)
 Next i
           If GTCR = 1 Then
            ReDim Preserve pp(1 To r)
            GoTo Line13
           End If
ElseIf kitri > 0 Then
 dip = nip
 nip = 0
 For i = 1 To r
 nip = nip + zz(i) * rr(i)
 Next i
 beta = nip / dip
 For i = 1 To r
 pp(i) = zz(i) + beta * pp(i)
 Next i
End If
ReDim Preserve pp(1 To r)
kitri = kitri + 1         'Update iteration counter.
'-----------------array multiplication
Line13: '=========================
ReDim Preserve SE(1 To r, 1 To 5)
ReDim Ap(1 To r)
For i = 1 To r
Ap(i) = SE(i, 3) * pp(i)
If m > 1 Then
If L > 1 Then
  If i - L > 0 Then
   Ap(i) = Ap(i) + SE(i, 1) * pp(i - L)
  End If
  If i + L < r + 1 Then
   Ap(i) = Ap(i) + SE(i, 5) * pp(i + L)
  End If
End If
End If
 If i - 1 > 0 Then
  Ap(i) = Ap(i) + SE(i, 2) * pp(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 4) * pp(i + 1)
 End If
Next i
ReDim Preserve Ap(1 To r)
'-----------------
dip = 0                   'Remainder of method steps
For i = 1 To r            'For-Next gives internal products.
If Abs(nip) < 1E+100 Then      'Confirm stability.
dip = dip + pp(i) * Ap(i)
```

```
Else
 NegHead = 3
 NegHeadNM = nm + 38      'Intervene if not stable.
 GoTo Line50
End If
Next i
If nip <> 0 Then          'Zero residual check.
alpha = nip / dip
ReDim xp(1 To r)
For i = 1 To r            'Calculate next estimates.
xp(i) = hx(i)
hx(i) = hx(i) + alpha * pp(i)
rr(i) = rr(i) - alpha * Ap(i)
Next i
ReDim Preserve rr(1 To r), xp(1 To r), hx(1 To r)
           If GTCR = 1 Then
            GTCR = 2
            GoTo Line11
           ElseIf GTCR = 2 Then
            GoTo Line11
           End If
Line14: '==========================
Else              'Zero residual upshot.
 ReDim xp(1 To r)
 For i = 1 To r
  xp(i) = hx(i)
  hx(i) = hx(i)
  rr(i) = rr(i)
 Next i
End If            'Zero residual end if.

maxerr = 0
maxdelta = 0
For i = 1 To r                 'Calculate closure difference.
  If Abs(hx(i) - xp(i)) > maxerr Then
    maxerr = Abs(hx(i) - xp(i))
  End If
  If Abs(hx(i) - xp(i)) > maxdelta Then
    maxdelta = Abs(hx(i) - xp(i))
  End If
  If Abs(rr(i)) > maxerr Then
    maxerr = Abs(rr(i))
  End If
Next i
If maxdelta < 0.00000000001 Then
 If maxerr > eps Then
   If kitro + kitrtr < 100 Then
     maxerr = 0.5 * eps
     If kitri = 1 Then
      kitri = 2
     End If
    Else
     resord = -1 * WorksheetFunction.RoundDown(WorksheetFunction.Log10(maxerr), 0)
     eps = 1 * 10 ^ (-resord)
     msgres = "Residual Order = " & resord & ". "
   End If
```

```
 End If
End If

    If maxerrp <= maxerr Then                   'Watches for rounding error; if error evident, recalculates residual, instead of using erroneous
values.
    If GTCR = 0 Then
        GTCR = 1
        For i = 1 To r
            hx(i) = xp(i)
        Next i
        GoTo Line11
      ElseIf kitri > 50 Then
        subsflag = 76
    End If
    End If

Line15: '==========================

TransT = 0
i = 1
For jm = 1 To m
For im = 1 To L
Trans(i) = 0
If CTi(i) <> 4 Then
    Trans(i) = TransTHRS(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), (Hni(i) + UMAi(i)),
(hx(i) + UMAi(i)))
    If Trans(i) <> transp(i) Then
     TransT = 1
    End If
End If
i = i + 1
Next im
Next jm
If subsflag = 1 Then
  subsflag = 0
  GoTo Line48
End If
If subsflag = 76 Then
  subsflag = 0
  GoTo Line16
End If
If subsflag = 77 Then
  subsflag = 0
  GoTo Line16b
End If

If kitro + kitrtr < 200 Then
If kitri > 100 Then
Line15a: '==========================

  If TransT = 1 Then
   GoTo Line16
  End If

i = 1
For jm = 1 To m
```

```
For im = 1 To L
       If CT(jm, im) <> 4 Then
       If hx(i) <= AqBi(i) Then
         GoTo Line16
       End If
       End If
If CTi(i) = 2 Then
        If side11(jm, im) + side12(jm, im) > 0 Then
        If Lons(jm, im) = 2 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
              If SidMat(jm, im, 1) <> sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i)) Then
               GoTo Line16
              End If
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a1k
         End If
         ElseIf Lons(jm, im) = 1 Then
Line9a1k:     '=========================
               hxa = 9999997
               If CT(jm - 1, im) <> 2 Then
                  hxa = hx(i - L)
                Else
                  hxa = 9999992
               End If
               If SidMat(jm, im, 1) <> sThresh(UMA(jm - 1, im), Bed(jm, im), SHD(jm, im), hxa) Then
                  GoTo Line16
               End If
          End If 'Lons
         End If
        If side21(jm, im) + side22(jm, im) > 0 Then
        If Lons(jm, im) = 1 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
              If SidMat(jm, im, 2) <> sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i)) Then
                GoTo Line16
              End If
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a2k
         End If
         ElseIf Lons(jm, im) = 2 Then
Line9a2k:     '=========================
               hxb = 9999997
               If CT(jm, im - 1) <> 2 Then
                 hxb = hx(i - 1)
                Else
                 hxb = 9999992
               End If
              If SidMat(jm, im, 2) <> sThresh(UMA(jm, im - 1), Bed(jm, im), SHD(jm, im), hxb) Then
                 GoTo Line16
              End If
          End If 'Lons
          End If
        If side31(jm, im) + side32(jm, im) > 0 Then
        If Lons(jm, im) = 2 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
              If SidMat(jm, im, 3) <> sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i)) Then
               GoTo Line16
```

```
              End If
           ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
            GoTo Line9a3k
          End If
          ElseIf Lons(jm, im) = 1 Then
Line9a3k:     '==========================
               hxc = 9999997
               If CT(jm + 1, im) <> 2 Then
                 hxc = hx(i + L)
                Else
                 hxc = 9999992
               End If
              If SidMat(jm, im, 3) <> sThresh(UMA(jm + 1, im), Bed(jm, im), SHD(jm, im), hxc) Then
                GoTo Line16
              End If
            End If 'Lons
           End If
         If side41(jm, im) + side42(jm, im) > 0 Then
         If Lons(jm, im) = 1 Then
          If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
              If SidMat(jm, im, 4) <> sThresh(UMA(jm, im), Bed(jm, im), SHD(jm, im), hx(i)) Then
               GoTo Line16
              End If
           ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
            GoTo Line9a4k
          End If
          ElseIf Lons(jm, im) = 2 Then
Line9a4k:     '==========================
               hxd = 9999997
               If CT(jm, im + 1) <> 2 Then
                 hxd = hx(i + 1)
                Else
                 hxd = 9999992
               End If
              If SidMat(jm, im, 4) <> sThresh(UMA(jm, im + 1), Bed(jm, im), SHD(jm, im), hxd) Then
                 GoTo Line16
              End If
           End If 'Lons
          End If
 End If
i = i + 1
Next im
Next jm

End If               'kitri.
Else
If kitri > 4 Then
GoTo Line15a
End If
End If

maxerrp = maxerr  'Set prior maxerr.

Else                                                'Intervene to end iteration if not converging in a reasonable number of cycles.
 NegHead = 3
 NegHeadNM = nm + 38
```

```
 GoTo Line50
End If                                              'Iteration counter end if.

Loop                                           '##############Bottom of method loop.

Line16: '==========================

i = 1
 For jm = 1 To m
  For im = 1 To L
       If CTi(i) Mod 2 = 1 Then
        If hx(i) <= AqBi(i) Then             'Check for dewatering.
          If kitro + kitrtr > 500 Then
           NegHead = 2
           NegHeadNM = nm + 38
           GoTo Line50
          Else
           If ATC = "Confined" Then
            If (Hni(i) - AqBi(i)) > 0.2 * (AqTi(i) - AqBi(i)) Then
              hx(i) = AqBi(i) + 0.2 * (AqTi(i) - AqBi(i))
             Else
              hx(i) = Hni(i)
            End If
           Else                 'Unconfined
            If (Hni(i) - AqBi(i)) > 0.2 * (Hnp(jm, im) - AqBi(i)) Then
              hx(i) = AqBi(i) + 0.2 * (Hnp(jm, im) - AqBi(i))
             Else
              hx(i) = Hni(i)
            End If
           End If
           maxerr = 2 * eps
          End If
        End If
       End If
      i = i + 1
 Next im
Next jm
If maxerr = 2 * eps Then
  subsflag = 77
  GoTo Line15
End If

Line16b: '==========================
For i = 1 To r
    transp(i) = Trans(i)  'Set prior transition code for next coefficient cycle.
Next i

If ThickCon <> "Next Step" Then
ReDim ho(1 To m, 1 To L)           'Thickness update.
Call HoHo(hx(), m, L, jm, im, ovp(), ovw(), beds(), Bed(), hob(), ho(), widpp(), Lons(), widp(), bppb(), bpp(), dx(), dy(), _
          SIT, AqT(), AqB(), UMA(), UMS(), ATC, CT(), lenp(), SB, DEC())
End If

If TransT > 0 Then
  If kitrtr < 1000 Then
     kitrtr = kitrtr + 1
```

```
     GoTo Line9
    Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
ElseIf maxerr > eps Then
  If kitro < 1000 Then
     kitro = kitro + 1
     GoTo Line9
    Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
End If

If GTC = 40 Then
GoTo Line40
End If
'----------------------------------------------------------------------------------------------
'END SIMULTANEOUS SOLUTION by MICCG!!
'
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Recursive portions of solution follow initial solution for time step.
If ThickCon = "Next Step" Then       'ThickCon if.
Else                                 'ThickCon else, "Same Step" update timing option...the recommended setting.
  '##########Begin outside iteration (recursive solution loop) for aquifer thickness.
Line31: '==========================

Do While kitri > 1   '----------------Top of outside iteration loop, continues until convergence on first inner iteration after thickness update.
If kitro < 1000 Then                    'Outside iteration limit check if.
kitro = kitro + 1
  GTC = 40
  GoTo Line9    'Recompute coefficients from new head estimates, from there, repeat simultaneous solution by MICCG.
Line40: '==========================
 Else                                                   'Intervene to end outside iteration if not converging in a reasonable number of cycles.
  NegHead = 3
  NegHeadNM = nm + 38
  GoTo Line50
 End If                                                 'Outside iteration counter end if.
Loop              '----------------Bottom of outside iteration loop.

End If                               'ThickCon end if.
   '##########End outside iteration for aquifer thickness.
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Last of recursion.

Line45: '==========================
 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1
    Hnn(jm, im) = hx(i)                         'Convert next head solution from system domain vector to model domain array.
    Hni(i) = Hno(jm, im)                        'Revert single index vector to step-original head for storage check below.
     If ho(jm, im) = DEC(jm, im) - AqB(jm, im) Then
       If msgc = "" Then
```

```
       msgc = "Moist Surface! "
       End If
       flaw = 4
     End If
    If Trans(i) > Transnm(jm, im) Then
     Transnm(jm, im) = Trans(i)
    End If
              If Transpn(i) \ 100 = 1 Then  'Integer division operator.
               If Trans(i) \ 100 = 2 Then
                If ic = "Instantaneous" Then
                 GoTo Line46
                End If
               End If
              ElseIf Transpn(i) \ 100 = 2 Then
               If Trans(i) \ 100 = 1 Then
                If ic = "Instantaneous" Then   'ic if
Line46: '==========================
                    msgtg = "Status Oscillation (Instantaneous Impulse Arrival). "
                End If                         'ic end if.
               End If
              End If
    If CT(jm, im) = 1 Then
         If ATC = "Confined" Then
           If ITS = "Deep Percolation" Then
             If Hnn(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
              GoTo Line47
             End If
           End If
         End If
        If Transnm(jm, im) > 0 Then
         If Right(IMS, 4) = "Head" Then
          If ic = "Steady" Then
           If msgti = "" Then
            msgti = "Impulse Distribution Through Confinement Transition. "
           End If
          End If
         End If
           If nm > 1 Then
           If Transpnm(jm, im) > 0 Then
           If qs * Qm(nm) > qs * Qm(nm - 1) Then
            If Qc(jm, im) < QCP(jm, im) Then
              msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
            End If
            ElseIf qs * Qm(nm) <= qs * Qm(nm - 1) Then
            If Qc(jm, im) > QCP(jm, im) Then
              msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
            End If
           End If
           End If
         End If
        End If
    End If
    If ATC = "Unconfined" Then
    If CTi(i) <> 4 Then
    If Hnn(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
Line47:   '==========================
```

```
            msgc = "Inundated Ground! "
            msgti = "Impulse Exceeds Aquifer Capacity. "
            flaw = 3
            NegHead = 6
            NegHeadNM = nm + 38
            GoTo Line50
    End If
    End If
    End If
   If CT(jm, im) <> 4 Then
     If MSTA <> 0 Then
        If ho(jm, im) < hoo(jm, im) * MSTA / 100 Then                                   'Check minimum allowed saturated thickness.
          msgmt = "Min Thickness! "                                                          'Update warning message for minimum allowed
saturated thickness.
        End If                                                                            'End if.
     End If
       If msgt = "" Then
       If Trans(i) \ 100 = 1 Then
       msgt = "Confinement Change. "
       ElseIf Trans(i) \ 100 = 2 Then
       msgt = "Confinement Change. "
     End If
     End If
    End If
    If o = nc Then
     QCP(jm, im) = Qc(jm, im)
     Transpnm(jm, im) = Transnm(jm, im)
    End If
    im = im + 1
    i = i + 1
   Loop
   jm = jm + 1
 Loop

'
'OUTPUT  (starts before end of time step loop)
'-----------------------------------------------------------------------
RSP(n) = 0                          'Prepare for response flow calculation.....................................................
RSPA(n) = 0
 ReDim Qnno(1 To m, 1 To L)
 i = 1
 jm = 1
 Do While jm < m + 1
   im = 1
   Do While im < L + 1              'Solved heads allow explicit calculation of flow through streambed.
    If CT(jm, im) = 2 Then              'Restrictive bed flow by Darcy's law.
        If hx(i) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
          Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnn(jm, im) - SHD(jm, im)) / bpp(jm, im)    'Full bed flux
term.
         ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
          Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im) - UMS(jm, im)) /
bpp(jm, im)      'Bed flux limited by gradient to bottom of bed.
          msgbg = "Bed Flux Limited by Desaturation. "
        End If
        If side11(jm, im) + side12(jm, im) > 0 Then
        If Lons(jm, im) = 2 Then
```

```
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 1) - csba(jm, im, 1))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
            GoTo Line9a1q
         End If
         ElseIf Lons(jm, im) = 1 Then
Line9a1q:     '=========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm - 1, im) * csea(jm, im, 1) - csba(jm, im, 1))
          End If 'Lons
         End If
        If side21(jm, im) + side22(jm, im) > 0 Then
        If Lons(jm, im) = 1 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 2) - csba(jm, im, 2))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
            GoTo Line9a2q
         End If
         ElseIf Lons(jm, im) = 2 Then
Line9a2q:     '=========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im - 1) * csea(jm, im, 2) - csba(jm, im, 2))
          End If 'Lons
          End If
        If side31(jm, im) + side32(jm, im) > 0 Then
        If Lons(jm, im) = 2 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 3) - csba(jm, im, 3))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
            GoTo Line9a3q
         End If
         ElseIf Lons(jm, im) = 1 Then
Line9a3q:     '=========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm + 1, im) * csea(jm, im, 3) - csba(jm, im, 3))
           End If 'Lons
          End If
        If side41(jm, im) + side42(jm, im) > 0 Then
        If Lons(jm, im) = 1 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 4) - csba(jm, im, 4))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
            GoTo Line9a4q
         End If
         ElseIf Lons(jm, im) = 2 Then
Line9a4q:     '=========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im + 1) * csea(jm, im, 4) - csba(jm, im, 4))
          End If 'Lons
         End If
     Qnno(jm, im) = qrs * dt * Qnno(jm, im) / UCF  'Flow given correct sign, multiplied by dt to convert to units of volume, converted to acre-feet
if applicable.
    RSP(n) = RSP(n) + Qnno(jm, im) 'Aggregate.
    If zone(jm, im) = 1 Then       'Differentiate.
     RSPA(n) = RSPA(n) + Qnno(jm, im) '.....................................................................................................
    End If
   End If
    im = im + 1
    i = i + 1
   Loop
```

```
   jm = jm + 1
 Loop

subsflag = 1                        'Independent domain budget block.
GoTo Line15
Line48: '==========================
Qtt = qrs * Qtt                     'Volume into aquifer during step.
RSP(n) = (-1) * UCF * RSP(n)        'Volume out.
dS = 0
i = 1
jm = 1
Do While jm < m + 1
  im = 1
  Do While im < L + 1               'Change in storage.
    dS = dS + FuncdVC(Trans(i), dx(im), dy(jm), dt, (Hno(jm, im) + UMA(jm, im)), (Hnn(jm, im) + UMA(jm, im)), CT(jm, im), SIT, ATC, AqT(jm, im),
widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
                    SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2),
_
                    ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRS(jm, im, 1), THRS(jm, im,
2), _
                    THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im))
    im = im + 1
    i = i + 1
  Loop
  jm = jm + 1
Loop
dS = qs * dS
BD = Abs(Qb - (dS + RSP(n)))  'Budget difference: volume in vs. change in storage plus volume out.
BLT = BLTF(Qb, dS, RSP(n), ic, Qm(nm), UCF, nc)
If BLT > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then 'Minimum scale for evaluation.
If BD > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then
If 100 * BD / BLT > 0.1 Then 'Difference divided by largest term, converted to percentage, compared to acceptance level...1/10 of 1%.
 If msgc <> "Moist Surface! " Then
   NegHeadNM = nm + 38
   NegHead = 5
   If Abs(IMPSMX) + Abs(IMPSMN) > 0.00001 Then
     msgep = "Budget Gap. "
    Else
     msgep = "No Volume Budget. "
   End If
   GoTo Line50
  End If
 End If
 End If
 End If
RSP(n) = RSP(n) / ((-1) * UCF)
Qtt = Qtt / qrs

   If HTII > 0 Then
    If o = nc Then
     HTS(nm) = -1 * Hnn(HTIJ, HTII)
    End If
   End If
If nm = nh Then
  If o = nc Then
   im = 1
```

```
   jm = 1
   Do While im < L + 1             'Stash head to plot.
    Do While jm < m + 1
      Hnnp(jm, im) = -1 * Hnn(jm, im)
     jm = jm + 1
    Loop
    jm = 1
    im = im + 1
   Loop
  End If
 End If

n = n + 1                        'TIME STEP PROGRESSION.
o = o + 1
If o = nc + 1 Then
 o = 1
 nm = nm + 1
End If

im = 1
jm = 1
 Do While im < L + 1
  Do While jm < m + 1
   Hn(jm, im) = Hnn(jm, im)      'UPDATE HEAD.
   Hno(jm, im) = Hn(jm, im)
   jm = jm + 1
  Loop
  jm = 1
  im = im + 1
 Loop

Loop                              'Time step loop!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                  'END OF TIME STEP COMPUTATION LOOP (continue output section)
Line50: '========================

 Sheet10.Unprotect Password:=CPC
 Sheet10.Cells(133, 2).Value = "Head Time Series at Node"
 If HTII > 0 Then
   Sheet10.Cells(134, 7).Value = HTII
   Sheet10.Cells(135, 7).Value = HTIJ
    If NegHead > 1 Then
        hrb = NegHeadNM - 38 + 140 - 1
     Else
        hrb = no + 140
    End If
    Sheet10.Range("c140:" & "c" & hrb).Value = WorksheetFunction.Transpose(HTS())        'Output head time series.
   Else
   Sheet10.Cells(135, 3).Value = "To plot an output head time series, please specify, on the Time tab, non-zero node indeces for the desired
location.  Thank you!"
 End If
     Sheet10.Cells(72, 55).Value = ""
     Sheet10.Cells(72, 2).Value = "Head Surface Output"
If nh > 0 Then
  Sheet10.Cells(73, 7).Value = nh
If nh < no + 1 Then
hrb = 76 + m
```

```
hrr = crf(L)
Sheet10.Range("c77:" & hrr & hrb).Value = Hnnp()        'Output head to plot.
Else
GoTo Line60
End If
Else
Line60: '=========================
Sheet10.Cells(75, 3).Value = "To plot head, please specify, on the Time tab, a period number for plotting that is greater than zero and less than or
equal to the number of simulation periods.  Thank you!"
End If
  If NegHead > 1 Then
  If NegHeadNM - 38 - 1 < nh Then
  hrb = 76 + m
  hrr = crf(L)
  Sheet10.Range("c77:" & hrr & hrb).ClearContents
  End If
  End If
 Sheet10.Protect Password:=CPC

RSPT = 0
RSPTA = 0
IMPST = 0
n = 1
nm = 1
Do While n < nn + 1                                     'Compute Impulse and Response Outputs and Totals.
 o = 1
 RSPO(nm) = 0
 RSPOA(nm) = 0
 Do While o < nc + 1
  RSPO(nm) = RSPO(nm) + RSP(n)
  RSPOA(nm) = RSPOA(nm) + RSPA(n)
  o = o + 1
  n = n + 1
 Loop
 IMPST = IMPST + Qm(nm)
 RSPT = RSPT + RSPO(nm)
 RSPTA = RSPTA + RSPOA(nm)
 nm = nm + 1
Loop
  If IMPSMX > 0 Then
   If IMPSMN < 0 Then
    IMPSMX = 999999999.123457
   End If
  End If

If IMPSMX <> 999999999.123457 Then
If IMPST <> 0 Then                  'SECONDARY OUTPUT BLOCK
If SO = "Truncated Response" Then   'Physical residual distributed evenly for single impulse scenario.
 n = 1
 Do While n < no + 1
  SOV(n) = RSPO(n) + (IMPST - RSPT) / no
  n = n + 1
 Loop
End If
If SO = "Cumulative Ratio" Then     'Ratio of cumulative response to impulse through each time step.
 CUMI = 0
```

```
 CUMR = 0
 n = 1
 Do While n < no + 1
  CUMI = CUMI + Qm(n)
  CUMR = CUMR + RSPO(n)
  If CUMI <> 0 Then
  SOV(n) = CUMR / CUMI
  Else
  SOV(n) = "-"
  End If
  n = n + 1
 Loop
End If
If SO = "Response Ratio" Then        'Ratio of response for time step to impulse total.
 n = 1
 Do While n < no + 1
  SOV(n) = RSPO(n) / IMPST
  n = n + 1
 Loop
End If
If SO = "Period Ratio" Then        'Ratio of response for time step to impulse for time step.
 n = 1
 Do While n < no + 1
  If Qm(n) <> 0 Then
  SOV(n) = RSPO(n) / Qm(n)
  Else
  SOV(n) = "-"
  End If
  n = n + 1
 Loop
End If
If SO = "Impact Factor" Then         'Fraction of annual response on symmetrical rolling basis.
 If St = "Annual Pattern" Then
 If NegHead = 1 Then
 IFT = 0
 CUMRn(0) = 0
 n = 1
 Do While n < no + 1
  CUMRn(n) = CUMRn(n - 1) + RSPO(n)
  n = n + 1
 Loop
 If PD = "Days" Then
  tt = 365 * WorksheetFunction.RoundDown(no / 365, 0) - ((2 * 365) - 1)
  n = 1
  Do While n < no + 1
   If n < 365 + 1 Then
    SOV(n) = RSPO(tt + n - 1) / (CUMRn(tt + n - 1 + (365 - 1) / 2) - CUMRn(tt + n - 1 - 1 - (365 - 1) / 2))
    IFT = IFT + SOV(n)
   Else
    SOV(n) = "-"
   End If
   n = n + 1
  Loop
  For n = 1 To 365
   SOV(n) = SOV(n) / IFT
  Next n
```

```
 Else
  tt = 12 * WorksheetFunction.RoundDown(no / 12, 0) - ((2 * 12) - 1)
  n = 1
  Do While n < no + 1
   If n < 12 + 1 Then
    SOV(n) = RSPO(tt + n - 1) / (((CUMRn(tt + n - 1 + 5) - CUMRn(tt + n - 1 - 7)) + (CUMRn(tt + n - 1 + 6) - CUMRn(tt + n - 1 - 6))) / 2)
    IFT = IFT + SOV(n)
   Else
    SOV(n) = "-"
   End If
   n = n + 1
  Loop
  For n = 1 To 12
   SOV(n) = SOV(n) / IFT
  Next n
 End If                         'PD end if.
 End If                         'NegHead end if.
 End If                         'ST end if.
End If                          'SO end if.
FCR = RSPT / IMPST
End If                          'IMPST end if.
End If                          'IMPSMX end if.

Sheet11.Unprotect Password:=CPC

hrb = 38 + no
Sheet11.Cells(6, 22).Value = flaw
Sheet11.Range("e29").ClearContents
If msgim2 = "10" Then                           'Message prep.
msgim2 = "Impulses Net 0! "
End If
If msgim2 = "11" Then
msgim2 = "Impulse Tab 0s! "
hrb = 38
Sheet11.Cells(6, 22).Value = 2
End If
If RSPOFM2 = 2 Then
msgh = msgh & "Bed Level Reset to Stream Head. "
End If
If msgti = "Discrete Impulse Distribution by Head Requires Non-zero Weighting. " Then
hrb = 38
Sheet11.Cells(6, 22).Value = 2
End If
If msgti = "Discrete Impulse Distribution by Head Requires Like Signs Throughout Grid. Distribute by Volume Instead. " Then
hrb = 38
Sheet11.Cells(6, 22).Value = 2
End If
If msgep <> "" Then
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
 If msgep = "Budget Gap. " Then
   Sheet11.Cells(10, 10).Value = epse
  Else
   Sheet11.Cells(10, 10).ClearContents
 End If
Else
```

```
Sheet11.Cells(10, 10).ClearContents
End If
If msgpb <> "" Then
msgh = msgpb & msgh
End If
If RSPOFM = 2 Then
msgh = msgh & "Head Equalization. "
ElseIf RSPOFM = 4 Then
msgh = msgh & "Stream Head Equalization. "
End If
If NegHead = 2 Then
msgtt = "Dewatered Cell(s)! "
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
msgres = ""
End If
Sheet11.Cells(10, 11).Value = 11
If NegHead = 3 Then
msgtt = "Iteration Limit Exceeded! "
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
Sheet11.Cells(10, 11).Value = epse
End If
If NegHead = 6 Then
hrb = NegHeadNM - 1
End If
If msgres <> "" Then
Sheet11.Cells(10, 11).Value = epse
End If
If NegHead = 4 Then
msgtt = "Impulse Distribution Iteration Limit Exceeded! "
hrb = NegHeadNM - 1
Sheet11.Cells(6, 22).Value = 3
End If
If NegHead = 7 Then
hrb = NegHeadNM
End If

If hrb > 38 Then
Sheet11.Range("e39:" & "e" & hrb).Value = WorksheetFunction.Transpose(RSPOA())     'Output response time series, Zone 1, then total, then secondary
output.
Sheet11.Range("f39:" & "f" & hrb).Value = WorksheetFunction.Transpose(RSPO())
 If IMPSMX <> 999999999.123457 Then
   If hrb = 38 + no Then
    If RSPOFM = 1 Then
      Sheet11.Range("g39:" & "g" & hrb).Value = WorksheetFunction.Transpose(SOV())
      Sheet11.Cells(29, 5).Value = FCR                                          'Output final computed cumulative ratio of response to impulse.
    End If
   End If
  Else
  msgh = "Impulse Alternation. "
 End If
End If

Else                                      'Thickness input else.
Sheet11.Unprotect Password:=CPC
```

```
   msgh = msghfun(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1))
   If msgh22(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1)) = 2 Then
   Sheet11.Cells(6, 22).Value = 2
   End If
End If                                    'End thickness input if.
   If TipW(3) = 1 Then
   msgtp = "Aquifer Top at or above Initial Piezometric Surface. "
   End If

Call MessengerA(msg, msgim2, msgbg, msgti, msgmt, msgtg, msgu, msgumat, _
                msgclt, msgwp, msgc, msgtp, msgh, msgim, msgt, CPC, _
                msgtt, msgep, msgres)

End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub IMTDC()                           'Solution of implicit representation Adapted For 2-layer delayed-yield aquifer with complete incision
adjustment.

Dim dt As Double                      'Dimension calculation time step.
Dim dx() As Double, dy() As Double    'Dimension increments.
Dim k() As Double                     'Dimension hydraulic conductivity.
Dim Sy() As Double, Ss() As Double    'Dimension storage properties.
Dim hoo() As Double, ho() As Double   'Dimension initial saturated thickness, transmitting thickness.
Dim AqT() As Double, AqB() As Double  'Dimension aquifer top, bottom.
Dim Hn() As Double, Hno() As Double   'Dimension initial head, original for step.
Dim Hni() As Double                   'Dimension single index initial head.
Dim Hnp() As Double, Hnpat() As Double 'Dimension permanent initial heads.
Dim Qd() As Double, Qm() As Double    'Dimension net flow impulse distribution, impulse time series.
Dim Qdmx As Double, Qdmn As Double    'Dimension impulse distribution max and min.
Dim Qdmna As Double                   'Dimension impulse min absolute value.
Dim Qc() As Double, QCP() As Double   'Dimension impulse for calculation.
Dim AA() As Double, BB() As Double    'Dimension flow coefficients.
Dim CC() As Double, DD() As Double    'Dimension flow coefficients.
Dim EE() As Double, b() As Double     'Dimension volume coefficients.
Dim SE() As Double, SEC() As Double   'Dimension matrix to hold system of equations, redimensioned below with element limits.
Dim Hnn() As Double, Hnnp() As Double 'Dimension next time step differential head, head to plot.
Dim Hnnpat() As Double                'Dimension head to plot.
Dim Qnno() As Double                  'Dimension next time step flow output, to be determined for response cells, converting calculated head by
Storage Coefficient, etc.
Dim RSP() As Double, RSPA() As Double 'Dimension Response, Zone Response.
Dim RSPO() As Double, RSPT As Double  'Dimension Response Output, Response Total.
Dim RSPOA() As Double, RSPTA As Double 'Dimension Response Output, Response Total, both for Zone.
Dim i As Integer, im As Integer       'Dimension matrix row index, model column index.
Dim z As Integer                      'Dimension confinement threshold index.
Dim jm As Integer, L As Integer       'Dimension model row index, number of model columns.
Dim m As Integer, r As Integer        'Dimension number of model columns, matrix rows.
Dim ITS As String, IMS As String      'Dimension Impulse Type Specification, Impulse Magnitude Specification.
Dim CT() As Integer, CTi() As Integer 'Dimension Cell Type.
Dim IA As Double, IAP As Double       'Dimension Input Area effects.
Dim qs As Double, qrs As Double       'Dimension Flow Sign, Flow Response Sign.
Dim PD As String, ic As String        'Dimension Period Denomination, Impulse Character.
Dim u As Long, n As Long, nc As Long  'Dimension units per period, number of periods, number of sub-periods.
```

```
Dim nn As Long, nh As Long            'Dimension total number of steps, period number to plot head.
Dim no As Long, nm As Long, o As Long 'Dimension original number of periods, number of of period in undivided increments, period counter.
Dim deno As Long                      'Dimension denomitor for characteristic impulse calculation.
Dim ATC As String                     'Dimension Aquifer Thickness Character.
Dim UCF As Double                     'Dimension Unit Conversion Factor.
Dim zone() As String                  'Dimension response Zone.
Dim msg As String, msgc As String     'Dimension warning messages.
Dim msgt As String, msgti As String   'Dimension warning messages.
Dim msgbg As String                   'Dimension warning messages.
Dim msgtt As String, msgh As String   'Dimension warning messages.
Dim msgtp As String, msgmt As String  'Dimension warning messages.
Dim msgim As String, msgim2 As String 'Dimension warning messages.
Dim msgpb As String, msgep As String  'Dimension warning messages.
Dim MSTA As Double                    'Dimension minimum saturated thickness allowed.
Dim DEC() As Double                   'Dimension depth of earth cover.
Dim IMPST As Double                   'Dimension impulse total.
Dim CPC As Variant                    'Dimension some dummy text here.
CPC = "*******"                       'Dimension nothing to see here again.
Dim RSPOFM As Integer                 'Dimension response output flag to catch initial head imbalance.
Dim RSPOFM2 As Integer                'Dimension stream head imbalance flag.
Dim count2 As Double, count13 As Double 'Dimension storage and area weighted counters for cell types.
Dim headsum2 As Double                'Dimension initial storage and area weighted head sums and averages by cell type, stream.
Dim headave2 As Double                'Dimension initial averages and sums.
Dim headsum2s As Double               'Dimension initial averages and sums.
Dim headsum13 As Double               'Dimension initial storage and area weighted head sums and averages by cell type.
Dim headave13 As Double               'Dimension initial averages and sums.
Dim SB As String, SIT As String       'Dimesion relative permeability class of streambed, streambed incision type.
Dim kpp() As Double, bpp() As Double  'Dimension discrete streambed conductivity, thickness.
Dim kppb() As Double, bppb() As Double  'Dimension discrete streambank conductivity, thickness.
Dim bppi() As Double, widp() As Double 'Dimension discrete streambed single index thickness, dummy width.
Dim widpp() As Double, lenp() As Double 'Dimension discrete streambed width, length.
Dim hrb As Variant, hrr As Variant    'Dimension output cell range extents.
Dim hrr2 As Variant                   'Dimension cell range extent.
Dim epse As Integer                   'Dimension iteration closure threshold exponent.
Dim IMPSMX As Double, IMPSMN As Double 'Dimension impulse max and min.
Dim NegHead As Integer                'Dimension negative head flag.
Dim NegHeadNM As Integer              'Dimension error period.
Dim AqBi() As Double, AqTi() As Double 'Dimension single-index variables for bottom and top.
Dim AvTp As String                    'Dimension transmission characteristic average type.
Dim Bed() As Double, BEDi() As Double 'Dimension discrete bed elevation.
Dim SHD() As Double                   'Dimension stream head.
Dim TipW() As Integer                 'Dimension thickness input parameter watch variable.
Dim hx() As Double                    'Dimension successive approximation head vector.
Dim eps As Double                     'Dimension convergence threshold for MICCG.
Dim Ax() As Double, rr() As Double    'Dimension MICCG solver variables.
Dim vv() As Double, zz() As Double    'Dimension MICCG solver variables.
Dim UT() As Double, DU() As Double    'Dimension MICCG solver preconditioner matrices.
Dim UM() As Double                    'Dimension MICCG solver preconditioner matrix.
Dim pp() As Double, Ap() As Double    'Dimension MICCG solver variables.
Dim nip As Double, dip As Double      'Dimension MICCG solver variables.
Dim alpha As Double, beta As Double   'Dimension MICCG solver variables.
Dim xp() As Double, maxerr As Double  'Dimension MICCG solver variables.
Dim maxerrp As Double                 'Dimension prior max error.
Dim kitr As Integer, kitri As Integer 'Dimension iteration trackers.
Dim kitrtr As Integer, kitro As Integer 'Dimension iteration trackers.
Dim GTC As Integer, GTCR As Integer   'Dimension GoTo codes.
```

```
Dim St As String                      'Dimension schedule type.
Dim Trans() As Integer                'Dimension pressurization transition flags.
Dim transp() As Integer               'Dimension prior pressurization transition flags.
Dim TransT As Integer                 'Dimension pressurization transition aggregate flag.
Dim Transnm() As Integer              'Dimension period transition.
Dim Transpnm() As Integer             'Dimension prior period transition.
Dim IGTC As Integer                   'Dimension impulse distribution GoTo code.
Dim Qt As Double                      'Dimension Newton solver variables for impulse distribution.
Dim Qtt As Double, Qf As Double       'Dimension impulse distribution variables.
Dim ObF As Double, ObFF As Double     'Dimension Newton solver variables for impulse distribution.
Dim dFdu As Double, Qttp As Double    'Dimension impulse distribution variable, prior Qtt.
Dim dS As Double                      'Dimension change in storage for global budget.
Dim BD As Double, BLT As Double       'Dimension budget difference, budget largest term.
Dim PermFact() As Double              'Dimension permissive bed coefficient factor.
Dim HTS() As Double, HTS2() As Double 'Dimension head time series output.
Dim HTII As Integer, HTIJ As Integer  'Dimension head time series location im index, jm index.
Dim kv() As Double, kh() As Double    'Dimension vertical, horizontal conductivity of aquitard.
Dim kvq() As Double                   'Dimension vertical conductivity of aquifer.
Dim Sigma() As Double, Sigs() As Double 'Dimension specific yield, specific storage of aquitard.
Dim Hnat() As Double, Hnoat() As Double 'Dimension initial head of aquitard.
Dim Hnnat() As Double, hoat() As Double 'Dimension solved head, saturated thickness of aquitard.
Dim Hniat() As Double                 'Dimension single index intial aquitard head.
Dim AAt() As Double, BBt() As Double  'Dimension primary coefficients for aquitard equations.
Dim CCt() As Double, DDt() As Double  'Dimension aquitard flow coefficients.
Dim EET() As Double, bbtt() As Double 'Dimension aquitard coefficients.
Dim GG() As Double, GGT() As Double   'Dimension vertical flow coefficients for flow between layers.
Dim msgat As String, msgres As String 'Dimension aquitard messages.
Dim PermFlag() As Integer             'Dimension permissivefactor flag, residual order.
Dim resord As Integer                 'Dimension residual order.
Dim maxdelta As Double                'Dimension maximum change in head estimates.
Dim LK As Integer                     'Dimension leakance flag.
Dim UMS() As Double, UMA() As Double  'Dimension Ultimate Matric Suction below streambed, aquifer top.
Dim UMAT() As Double                  'Dimension Critical Matric Suction in aquitard, previous impulse.
Dim UMAi() As Double, UMATi() As Double 'Dimension single index variables.
Dim UMSi() As Double                  'Dimension single index suction beneath stream bed.
Dim subsflag As Integer               'Dimension substitution routine flag.
Dim Transpn() As Integer              'Dimension previous step transition.
Dim msgtg As String                   'Dimension g instability message.
Dim kvtb() As Double                  'Dimension average vertical conductivity.
Dim flaw As Integer                   'Dimension flaw flag.
Dim hxa As Double, hxat As Double     'Dimension head place holders.
Dim hxb As Double, hxbt As Double     'Dimension head place holders.
Dim hxc As Double, hxct As Double     'Dimension head place holders.
Dim hxd As Double, hxdt As Double     'Dimension head place holders.
Dim AqTa As Double, AqtB As Double    'Dimension aquifer top place holders.
Dim AqTc As Double, AqTd As Double    'Dimension aquifer top place holders.
Dim side11t() As Double, side12t() As Double 'Dimension wet channel side for complete incision.
Dim side21t() As Double, side22t() As Double 'Dimension wet channel side for complete incision.
Dim side31t() As Double, side32t() As Double 'Dimension wet channel side for complete incision.
Dim side41t() As Double, side42t() As Double 'Dimension wet channel side for complete incision.
Dim side11a() As Double, side12a() As Double 'Dimension wet channel side for complete incision.
Dim side21a() As Double, side22a() As Double 'Dimension wet channel side for complete incision.
Dim side31a() As Double, side32a() As Double 'Dimension wet channel side for complete incision.
Dim side41a() As Double, side42a() As Double 'Dimension wet channel side for complete incision.
Dim Lons() As Integer                  'Dimension long side indicator.
Dim Lon1 As Integer, Lon2 As Integer  'Dimension long side rotation trackers.
```

```
Dim msgu As String, msgumat As String 'Dimension matric suction messages.
Dim xpo() As Double                   'Dimension prior outside iteration hx estimates.
Dim cset() As Double, csbt() As Double 'Dimension channel side e and b coefficient, aquitard.
Dim csea() As Double, csba() As Double 'Dimension channel side e and b coefficient, aquifer.
Dim SidMat() As Integer                'Dimension channel side head condition flag.
Dim BedMat() As Integer                'Dimension channel bed head condition flag.
Dim TransC() As Long                   'Dimension pressurization oscillation counter.
Dim Gflag() As Integer                 'Dimension pressurization oscillation flag.
Dim GCT() As Integer, GCA() As Integer 'Dimension total and above counters for G flag.
Dim qnoc() As Double                   'Dimension channel side e and b coefficient, aquifer.
Dim msgclt As String, msgwp As String  'Dimension Complete override, stream width messages.
Dim ovp() As Double, ovw() As Double   'Dimension bank overlap variables.
Dim beds() As Double                   'Dimension surrounding bed elevation variable.
Dim THRS() As Double, THRSt() As Double 'Dimension storage parameter threshold level stack.
Dim THRCNT() As Integer, THRCNTt() As Integer 'Dimension Storage Parameter Threshold level count.
Dim hob() As Double, hobat() As Double 'Dimension saturated thicknesses from a particular side, for horizontal flow.
Dim dvtc() As Double                   'Dimension volume increment placeholder for depressurization.
Dim rpc() As Integer                   'Dimension repressurization counter.
Dim NBRS() As Integer                 'Dimension nearby response cell counter per side.
Dim Dewflag() As Integer              'Dimension aquitard dewatering flag.
Dim CAS() As Double                   'Dimension cell area times nominal storage.
Dim vfhd As String, vfch As String    'Dimension vertical flow head dissipation options.
Dim Knocker As String                 'Dimension wet cell face seepage option.
Dim trigger As Integer                'Dimension aquitard head adjustment iteration trigger.
Dim GTCRT As Integer                  'Dimension trigger goto code.
Dim KOC As Integer                    'Dimension Kickout Code.
Dim msgsep As String                  'Dimension head separation message.
Dim ATDF As Integer                   'Dimension aquitard head surface dewatering flag.
Dim delta As Double                   'Dimension MICCG upper preconditioner matrix diagonal scale factor to prevent negatives.
Dim PFSflag As Integer                'Dimension PermFact scale flag.
Dim MaxTerm As Double                 'Dimension maximum augmented matrix entry value for aquifer rows.
Dim MaxTermT As Double                'Dimension maximum augmented matrix entry value for aquitard rows.
Dim Qb As Double                      'Dimension domain step impulse for budget block.

Dim SEB As String                     'Dimension Stream End Bank option.

Line0:   '=========================

ATDF = 0

L = Sheet3.Cells(12, 8).Value         'Load number of cells on one side of model.
m = Sheet3.Cells(15, 8).Value         'Load number of cells on other side of model.
r = L * m * 2                         'Number of rows in equation matrix.

no = Sheet11.Cells(13, 5).Value       'Load number of periods.
nc = Sheet11.Cells(16, 5).Value       'Load number of sub-period steps.
nn = no * nc                          'Set number of total steps.

ReDim dx(1 To L), dy(1 To m)           'Set custom array dimensions.
ReDim k(1 To m, 1 To L)
ReDim hoo(1 To m, 1 To L), ho(1 To m, 1 To L)
ReDim Hno(1 To m, 1 To L), Hn(1 To m, 1 To L)
ReDim Hnp(1 To m, 1 To L), Hnpat(1 To m, 1 To L)
ReDim Qd(1 To m, 1 To L)
ReDim Qm(1 To no)
ReDim Hnn(1 To m, 1 To L), Hnnp(1 To m, 1 To L), Hnnpat(1 To m, 1 To L)
```

```
ReDim Qnno(1 To m, 1 To L)
ReDim RSP(1 To nn), RSPA(1 To nn)
ReDim RSPO(1 To no)
ReDim RSPOA(1 To no)
ReDim CT(1 To m, 1 To L)
ReDim zone(1 To m, 1 To L)
ReDim DEC(1 To m, 1 To L)
ReDim Sy(1 To m, 1 To L), Ss(1 To m, 1 To L)
ReDim kpp(1 To m, 1 To L), bpp(1 To m, 1 To L), bppi(1 To r / 2)
ReDim kppb(1 To m, 1 To L), bppb(1 To m, 1 To L)
ReDim widp(1 To m, 1 To L), widpp(1 To m, 1 To L)
ReDim lenp(1 To m, 1 To L)
ReDim AqBi(1 To r / 2), AqTi(1 To r / 2)
ReDim Bed(1 To m, 1 To L), BEDi(1 To r / 2), SHD(1 To m, 1 To L)
ReDim AqT(1 To m, 1 To L), AqB(1 To m, 1 To L)
ReDim TipW(1 To 5)
ReDim Transpnm(1 To m, 1 To L)
ReDim Hni(r / 2 + 1 To r)
ReDim CTi(1 To r / 2)
ReDim Qc(1 To m, 1 To L), QCP(1 To m, 1 To L)
ReDim HTS(0 To no), HTS2(0 To no)
ReDim kv(1 To m, 1 To L), kh(1 To m, 1 To L), kvq(1 To m, 1 To L)
ReDim Sigma(1 To m, 1 To L), Sigs(1 To m, 1 To L)
ReDim Hnat(1 To m, 1 To L), Hnoat(1 To m, 1 To L)
ReDim Hnnat(1 To m, 1 To L), hoat(1 To m, 1 To L)
ReDim Hniat(1 To r / 2)
ReDim Transpn(1 To r)
ReDim UMS(1 To m, 1 To L), UMA(1 To m, 1 To L)
ReDim UMAT(1 To m, 1 To L)
ReDim UMAi(r / 2 + 1 To r), UMATi(1 To r / 2), UMSi(1 To r / 2)
ReDim PermFact(1 To m, 1 To L, 1 To 2), Lons(1 To m, 1 To L)
ReDim ovp(1 To m, 1 To L, 1 To 4), ovw(1 To m, 1 To L, 1 To 4)
ReDim beds(1 To m, 1 To L, 1 To 4)
ReDim THRS(1 To m, 1 To L, 1 To 5), THRSt(1 To m, 1 To L, 1 To 5)
ReDim THRCNT(1 To m, 1 To L), THRCNTt(1 To m, 1 To L)
ReDim hob(1 To m, 1 To L, 1 To 4), hobat(1 To m, 1 To L, 1 To 4)
ReDim NBRS(1 To m, 1 To L, 1 To 4)
'
'INPUT
'-----------------------------------------------------------------------
RSPOFM = 1                          'Initialize flags all clear.
RSPOFM2 = 1

Call pload(St, SB, SIT, ITS, IMS, ic, ATC, AvTp, HTII, HTIJ, flaw)
vfch = WorksheetFunction.Proper(Sheet6.Cells(8, 37).Value)
vfhd = WorksheetFunction.Proper(Sheet6.Cells(10, 37).Value)
Knocker = WorksheetFunction.Proper(Sheet6.Cells(9, 30).Value)
SEB = WorksheetFunction.Proper(Sheet12.Cells(12, 14).Value)     'Load Stream End Bank option.

epse = Sheet11.Cells(10, 5).Value    'Load closure threshold exponent.
eps = 1 * 10 ^ (-epse)                     'Set convergence threshold value.

msg = Sheet11.Cells(30, 5).Value      'Load warning message, if any.

If ITS = "Well Pumping" Then          'Set flow signs for practical conventions.
 qs = 1
```

```
 qrs = -1
 Else
 qs = -1
 qrs = 1
End If
If WorksheetFunction.Proper(Sheet3.Cells(8, 8).Value) = "Acre-Feet" Then   'Set Unit Conversion Factor.
 UCF = 43560
 Else
 UCF = 1
End If
If WorksheetFunction.Proper(Sheet4.Cells(4, 4).Value) = "A" Then  'Load Min Sat Thick Allow.
 MSTA = Sheet4.Cells(15, 5).Value
End If

PD = WorksheetFunction.Proper(Sheet11.Cells(14, 5).Value) 'Load time unit denomination.
u = Sheet11.Cells(15, 5).Value       'Load units per period.
nh = Sheet11.Cells(23, 5).Value      'Load period to plot heads.
If PD = "Days" Then                  'Set time step.
 dt = u / nc
 ElseIf PD = "Months" Then
 dt = u * 365.25 / 12 / nc
 ElseIf PD = "Hours" Then
 dt = u / nc / 24
End If
If ic = "Steady" Then
 deno = nc
 Else
 deno = 1
End If

For im = 1 To L
For jm = 1 To m
   CT(jm, im) = Sheet3.Cells(26 + jm, 3 + im).Value
Next jm
Next im

'Load initial parameter values.
Call LoadC(RSPOFM2, SB, SIT, IMS, im, L, jm, m, msgwp, _
          dx(), dy(), CT(), widp(), widpp(), lenp(), Lons(), AqT(), _
          kv(), kh(), kvq(), LK, Sigma(), Sigs(), AqB(), DEC(), _
          k(), zone(), Sy(), Ss(), Hn(), Hno(), Hnp(), Hnat(), _
          Hnoat(), Hnpat(), UMA(), UMAT(), kpp(), bpp(), kppb(), bppb(), _
          SHD(), Bed(), msgpb, msgh, TipW(), UMS(), _
          Qd(), Qdmx, Qdmn, Qdmna, msgu, Lon1, Lon2, ATC)

Call Neigh(CT(), L, m, NBRS(), widp(), widpp(), bppb())

Call OVPERer(m, L, jm, im, ovp(), ovw(), Bed(), beds(), widpp(), Lons(), widp(), bppb(), dx(), dy(), SIT, msgclt, NBRS(), SEB)
If Left(msgclt, 2) = "Ex" Then
 NegHead = 7
 NegHeadNM = 38
 flaw = 3
 GoTo Line50
End If
                                        'Preliminary processing of parameters.
Call PrelimC(jm, m, im, L, CT(), Bed(), bpp(), _
```

```
            AqT(), DEC(), TipW(), Hn(), Hnat(), AqB(), AqBi(), _
            AqTi(), BEDi(), CTi(), bppi(), UMAi(), UMATi(), UMSi(), _
            UMA(), UMAT(), UMS(), ATC, r, PermFact(), ovp(), ovw(), beds(), _
            SIT, dx(), dy(), headsum2, count2, headsum2s, headsum13, count13, _
            widp(), widpp(), bppb(), lenp(), Ss(), Sy(), SB, SHD(), hx(), Sigs(), _
            Sigma(), msgsep, flaw)

Call HoHo3(hx(), m, L, jm, im, ovp(), ovw(), beds(), Bed(), hob(), ho(), widpp(), Lons(), widp(), bppb(), bpp(), dx(), dy(), _
          SIT, AqT(), AqB(), UMA(), UMS(), ATC, CT(), lenp(), SB)
Call Hoot(m, L, jm, im, ho(), hoo())

For nm = 1 To no
 Qm(nm) = Sheet11.Cells(38 + nm, 4).Value  'Load impulse magnitude time series.
 If nm = 1 Then
 IMPSMX = Qm(1)
 IMPSMN = Qm(1)
 End If
 If IMPSMX < Qm(nm) Then
  IMPSMX = Qm(nm)
 End If
 If IMPSMN > Qm(nm) Then
  IMPSMN = Qm(nm)
 End If
Next nm

 If HTII > 0 Then
         HTS(0) = -1 * Hn(HTIJ, HTII)
         HTS2(0) = -1 * Hnat(HTIJ, HTII)
 End If

 If Qdmx > 0 Then
  If Qdmn < 0 Then
   msgim = "Mixed Impulse. "
  End If
 End If

If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
headave2 = headsum2 / count2
headave13 = headsum13 / count13
 If Abs(headave2 - headave13) > 0.000001 Then             'Check for initial head balance.
  msgh = msgh & "Head Equalization. "
  RSPOFM = 2
 End If
End If
If Abs(headsum2s) > 0.000001 Then
 msgh = msgh & "Stream Head Equalization. "
 RSPOFM = 4
End If

'Establish and sort pressurization thresholds.
Call THstackT(ovp(), ovw(), beds(), dx(), dy(), _
             AqT(), Bed(), bpp(), UMS(), UMAT(), _
             THRSt(), THRCNTt(), m, L, CT(), SIT, _
             widp(), bppb(), widpp(), UMA())
Call THstack(ovp(), ovw(), beds(), dx(), dy(), AqB(), _
             AqT(), Bed(), bpp(), UMS(), UMA(), ATC, _
```

```
             THRS(), THRCNT(), m, L, CT(), SIT, widp(), _
             bppb(), widpp())

For jm = 1 To m        'Check that aquifer bottom beneath bed and banks.
For im = 1 To L
 If THRCNT(jm, im) > 0 Then
  If CT(jm, im) = 2 Then
     If AqB(jm, im) >= THRS(jm, im, THRCNT(jm, im)) Then
       TipW(1) = 3
     End If
   ElseIf CT(jm, im) <> 4 Then
     If AqB(jm, im) > THRS(jm, im, THRCNT(jm, im)) Then
       TipW(1) = 5
     End If
  End If
 End If
Next im
Next jm

If TipW(1) + TipW(2) + TipW(4) + TipW(5) = 0 Then                                'Condition routine on valid thickness input, end if near end of
subroutine.

'
'TIME STEP COMPUTATION LOOP
'-----------------------------------------------------------------------
NegHead = 1

  n = 1                               'Initialize time step counter.
  o = 1
  nm = 1

Do While n < nn + 1                'Loop for time steps!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

GTC = 0                             'Initialize codes for each time step.
kitro = 0
kitrtr = 0
TransT = 0
delta = 0
PFSflag = 0
MaxTerm = 0
MaxTermT = 0

If n > 1 Then
   Qttp = Qtt
   For i = 1 To r
    Transpn(i) = Trans(i)
   Next i
End If
ReDim xpo(1 To r)
ReDim rpc(1 To m, 1 To L)
ReDim AvTpT(1 To m, 1 To L, 1 To 4)
ReDim AvTpA(1 To m, 1 To L, 1 To 4)
ReDim Trans(1 To r)
ReDim transp(1 To r)
ReDim TransC(1 To r)
ReDim Dewflag(1 To r / 2)
```

```
ReDim Gflag(1 To m, 1 To L)
ReDim GCT(1 To m, 1 To L)
ReDim GCA(1 To m, 1 To L)
ReDim CAS(1 To r / 2)
ReDim kvtb(1 To r / 2)
ReDim dvtc(1 To r / 2)
ReDim Preserve Hn(1 To m, 1 To L)
ReDim Preserve Hnat(1 To m, 1 To L)
If o = 1 Then
 ReDim Transnm(1 To m, 1 To L)
End If

'IMPULSE DISTRIBUTION BLOCK.+++++++++++++
Call IDBC(o, ic, Qc(), Qtt, Qm(), Qf, IA, _
        IAP, m, L, CT(), IMS, _
        Qd(), ITS, dx(), dy(), Sigma(), Sigs(), SIT, _
        Hn(), AqT(), Bed(), Ss(), Sy(), AqB(), bpp(), SB, _
        UMA(), UMS(), msgim, msgti, UCF, qs, Qdmna, _
        msgim2, Hnat(), msgc, NegHead, NegHeadNM, _
        Qt, msgtt, subsflag, DEC(), nm, nc, _
        dt, flaw, THRS(), THRCNT(), THRSt(), THRCNTt(), _
        widp(), ovp(), ovw(), beds(), lenp(), widpp(), bppb(), UMAT(), Qb)
If subsflag = 8 Then
 subsflag = 0
 GoTo Line8
 ElseIf subsflag = 50 Then
 subsflag = 0
 GoTo Line50
End If

Line8: '==========================

i = 1                                          'Re-initialize head vectors.
 For jm = 1 To m
  For im = 1 To L
Hni(i + r / 2) = Hn(jm, im)                                  'Fill vectors.
Hniat(i) = Hnat(jm, im)
hx(i + r / 2) = Hn(jm, im)
hx(i) = Hnat(jm, im)
xpo(i) = hx(i)
xpo(i + r / 2) = hx(i + r / 2)
If CT(jm, im) <> 4 Then
  If Hn(jm, im) <= AqB(jm, im) Then                        'Check that head not beneath aquifer.
Line8b: '==========================
    NegHead = 2
    NegHeadNM = nm + 38
    GoTo Line50
  End If
  If Hnat(jm, im) <= AqB(jm, im) Then
  GoTo Line8b
  End If
End If
    CAS(i) = dx(im) * dy(jm) * SfuncCt(CT(jm, im), SIT, "Confined", (AqT(jm, im) + 0.01), AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im),
Bed(jm, im), _
                                    bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im), SB, UMAT(jm, im), UMS(jm, im), ovp(jm, im, 1), _
                                    ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), _
```

```
                                    ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
    kvtb(i) = fkvtbc(SIT, kv(jm, im), dx(im), dy(jm), CT(jm, im), Bed(jm, im), bpp(jm, im), AqT(jm, im), kvq(jm, im), _
                     LK, widpp(jm, im), widp(jm, im), lenp(jm, im), bppb(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                     ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), _
                     ovw(jm, im, 4), beds(jm, im, 4))
    i = i + 1
 Next im
Next jm
subsflag = 1
GoTo Line17                             'Thickness update.

Line9: '=========================       'Line label for point of return, if pressurization transition in first set of iterations, and for outside
iteration.
'
'FILL PRIMARY COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------------------------
ReDim AA(1 To m, 1 To L), BB(1 To m, 1 To L), CC(1 To m, 1 To L)  'Redimension A, B, C to reset all to empty.
ReDim DD(1 To m, 1 To L), EE(1 To m, 1 To L), b(1 To m, 1 To L)   'Redimension D, E, b to reset all to empty.
ReDim AAt(1 To m, 1 To L), BBt(1 To m, 1 To L), CCt(1 To m, 1 To L)
ReDim DDt(1 To m, 1 To L), EET(1 To m, 1 To L), bbtt(1 To m, 1 To L)
ReDim GG(1 To m, 1 To L), GGT(1 To m, 1 To L)
ReDim side11t(1 To m, 1 To L), side12t(1 To m, 1 To L)
ReDim side21t(1 To m, 1 To L), side22t(1 To m, 1 To L)
ReDim side31t(1 To m, 1 To L), side32t(1 To m, 1 To L)
ReDim side41t(1 To m, 1 To L), side42t(1 To m, 1 To L)
ReDim side11a(1 To m, 1 To L), side12a(1 To m, 1 To L)
ReDim side21a(1 To m, 1 To L), side22a(1 To m, 1 To L)
ReDim side31a(1 To m, 1 To L), side32a(1 To m, 1 To L)
ReDim side41a(1 To m, 1 To L), side42a(1 To m, 1 To L)
ReDim Preserve CT(1 To m, 1 To L), xpo(1 To r)
ReDim cset(1 To m, 1 To L, 1 To 4)
ReDim csbt(1 To m, 1 To L, 1 To 4)
ReDim csea(1 To m, 1 To L, 1 To 4)
ReDim csba(1 To m, 1 To L, 1 To 4)
ReDim SidMat(1 To m, 1 To L, 1 To 2, 1 To 4)
ReDim BedMat(1 To m, 1 To L)
ReDim PermFlag(1 To m, 1 To L)
ReDim qnoc(1 To m, 1 To L, 1 To 4)
 i = 1
 For jm = 1 To m
  For im = 1 To L                       'Calculate coefficient values.
   If CT(jm, im) <> 4 Then
    dvtc(i) = FuncdVatC(Trans(i), dx(im), dy(jm), dt, (Hnat(jm, im) + UMAT(jm, im)), (AqT(jm, im) - UMA(jm, im) + UMAT(jm, im)), CT(jm, im), SIT,
ATC, AqT(jm, im), widp(jm, im), lenp(jm, im), _
                        widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im), AqB(jm, im), SB, UMAT(jm, im), UMS(jm,
im), _
                        ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), _
                        ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRSt(jm, im, 1), THRSt(jm, im, 2), _
                        THRSt(jm, im, 3), THRSt(jm, im, 4), THRSt(jm, im, 5), THRCNTt(jm, im), DEC(jm, im))
    hxa = 9999997
    hxb = 9999997
    hxc = 9999997
    hxd = 9999997
    hxat = 9999997
    hxbt = 9999997
    hxct = 9999997
```

```
    hxdt = 9999997
    If jm - 1 > 0 Then
     If CT(jm - 1, im) < 4 Then
      AA(jm, im) = khbarC(AvTp, k(jm - 1, im), hob(jm - 1, im, 3), dy(jm - 1), _
                                k(jm, im), hob(jm, im, 1), dy(jm), dx(im))
      AAt(jm, im) = khbarC(AvTp, kh(jm - 1, im), hobat(jm - 1, im, 3), dy(jm - 1), _
                                 kh(jm, im), hobat(jm, im, 1), dy(jm), dx(im))
                qnoc(jm, im, 1) = QnoCsideC(AvTp, CT(jm, im), AqT(jm, im), AqT(jm - 1, im), hx(i - L), hobat(jm - 1, im, 3), UMAT(jm, im), kh(jm,
im), _
                                            kh(jm - 1, im), dx(im), dy(jm), dy(jm - 1), hobat(jm, im, 1), SHD(jm - 1, im), SHD(jm, im), _
                                            DEC(jm, im), UMA(jm, im), hx(i), Knocker)
                  b(jm, im) = b(jm, im) - qnoc(jm, im, 1)
                  bbtt(jm - 1, im) = bbtt(jm - 1, im) + qnoc(jm, im, 1)
      If CT(jm - 1, im) <> 2 Then
        hxa = hx(r / 2 + i - L)
        hxat = hx(i - L)
        AqTa = AqT(jm - 1, im)
       Else
        hxa = 9999992
        hxat = 9999992
      End If
    End If
   End If
   If im - 1 > 0 Then
    If CT(jm, im - 1) < 4 Then
      BB(jm, im) = khbarC(AvTp, k(jm, im - 1), hob(jm, im - 1, 4), dx(im - 1), _
                                k(jm, im), hob(jm, im, 2), dx(im), dy(jm))
      BBt(jm, im) = khbarC(AvTp, kh(jm, im - 1), hobat(jm, im - 1, 4), dx(im - 1), _
                                 kh(jm, im), hobat(jm, im, 2), dx(im), dy(jm))
                qnoc(jm, im, 2) = QnoCsideC(AvTp, CT(jm, im), AqT(jm, im), AqT(jm, im - 1), hx(i - 1), hobat(jm, im - 1, 4), UMAT(jm, im), kh(jm,
im), _
                                            kh(jm, im - 1), dy(jm), dx(im), dx(im - 1), hobat(jm, im, 2), SHD(jm, im - 1), SHD(jm, im), _
                                            DEC(jm, im), UMA(jm, im), hx(i), Knocker)
                  b(jm, im) = b(jm, im) - qnoc(jm, im, 2)
                  bbtt(jm, im - 1) = bbtt(jm, im - 1) + qnoc(jm, im, 2)
      If CT(jm, im - 1) <> 2 Then
        hxb = hx(r / 2 + i - 1)
        hxbt = hx(i - 1)
        AqtB = AqT(jm, im - 1)
       Else
        hxb = 9999992
        hxbt = 9999992
      End If
    End If
   End If
   If jm + 1 < m + 1 Then
    If CT(jm + 1, im) < 4 Then
      CC(jm, im) = khbarC(AvTp, k(jm + 1, im), hob(jm + 1, im, 1), dy(jm + 1), _
                                k(jm, im), hob(jm, im, 3), dy(jm), dx(im))
      CCt(jm, im) = khbarC(AvTp, kh(jm + 1, im), hobat(jm + 1, im, 1), dy(jm + 1), _
                                 kh(jm, im), hobat(jm, im, 3), dy(jm), dx(im))
                qnoc(jm, im, 3) = QnoCsideC(AvTp, CT(jm, im), AqT(jm, im), AqT(jm + 1, im), hx(i + L), hobat(jm + 1, im, 1), UMAT(jm, im), kh(jm,
im), _
                                            kh(jm + 1, im), dx(im), dy(jm), dy(jm + 1), hobat(jm, im, 3), SHD(jm + 1, im), SHD(jm, im), _
                                            DEC(jm, im), UMA(jm, im), hx(i), Knocker)
                  b(jm, im) = b(jm, im) - qnoc(jm, im, 3)
```

```
                bbtt(jm + 1, im) = bbtt(jm + 1, im) + qnoc(jm, im, 3)
      If CT(jm + 1, im) <> 2 Then
        hxc = hx(r / 2 + i + L)
        hxct = hx(i + L)
        AqTc = AqT(jm + 1, im)
       Else
        hxc = 9999992
        hxct = 9999992
      End If
    End If
   End If
   If im + 1 < L + 1 Then
    If CT(jm, im + 1) < 4 Then
      DD(jm, im) = khbarC(AvTp, k(jm, im + 1), hob(jm, im + 1, 2), dx(im + 1), _
                                k(jm, im), hob(jm, im, 4), dx(im), dy(jm))
      DDt(jm, im) = khbarC(AvTp, kh(jm, im + 1), hobat(jm, im + 1, 2), dx(im + 1), _
                                 kh(jm, im), hobat(jm, im, 4), dx(im), dy(jm))
                qnoc(jm, im, 4) = QnoCsideC(AvTp, CT(jm, im), AqT(jm, im), AqT(jm, im + 1), hx(i + 1), hobat(jm, im + 1, 2), UMAT(jm, im), kh(jm,
im), _
                                            kh(jm, im + 1), dy(jm), dx(im), dx(im + 1), hobat(jm, im, 4), SHD(jm, im + 1), SHD(jm, im), _
                                            DEC(jm, im), UMA(jm, im), hx(i), Knocker)
                b(jm, im) = b(jm, im) - qnoc(jm, im, 4)
                bbtt(jm, im + 1) = bbtt(jm, im + 1) + qnoc(jm, im, 4)
      If CT(jm, im + 1) <> 2 Then
        hxd = hx(r / 2 + i + 1)
        hxdt = hx(i + 1)
        AqTd = AqT(jm, im + 1)
       Else
        hxd = 9999992
        hxdt = 9999992
      End If
     End If
   End If
   If Gflag(jm, im) < 1 Then
      GG(jm, im) = FGGc(rpc(jm, im), kv(jm, im), dx(im), dy(jm), CT(jm, im), SB, Bed(jm, im), bpp(jm, im), AqT(jm, im), AqB(jm, im), kvq(jm, im),
AvTp, _
                        hx(i + r / 2), LK, SIT, hx(i), Hn(jm, im), UMA(jm, im), UMS(jm, im), GCT(jm, im), GCA(jm, im), _
                        widpp(jm, im), widp(jm, im), lenp(jm, im), bppb(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                        ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), _
                        ovw(jm, im, 4), beds(jm, im, 4), UMAT(jm, im), vfhd, vfch)
                  If Gflag(jm, im) = -1 Then
                    GCT(jm, im) = GCT(jm, im) + 1
                    If hx(i + r / 2) >= AqT(jm, im) - UMA(jm, im) Then
                     GCA(jm, im) = GCA(jm, im) + 1
                    End If
                  End If
    ElseIf Gflag(jm, im) = 1 Then
      GG(jm, im) = FGGc(rpc(jm, im), kv(jm, im), dx(im), dy(jm), CT(jm, im), SB, Bed(jm, im), bpp(jm, im), AqT(jm, im), AqB(jm, im), kvq(jm, im),
AvTp, _
                        797979, LK, SIT, hx(i), Hn(jm, im), UMA(jm, im), UMS(jm, im), GCT(jm, im), GCA(jm, im), _
                        widpp(jm, im), widp(jm, im), lenp(jm, im), bppb(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                        ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), _
                        ovw(jm, im, 4), beds(jm, im, 4), UMAT(jm, im), vfhd, vfch)
   End If
    GGT(jm, im) = GG(jm, im)
  End If                            'CT = 4 end if.
```

```
  EE(jm, im) = EE(jm, im) + GG(jm, im) + FuncEC(AA(jm, im), BB(jm, im), CC(jm, im), DD(jm, im), Trans(i + r / 2), dx(im), dy(jm), dt, Hn(jm, im),
hx(i + r / 2), CT(jm, im), SIT, _
                ATC, AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm,
im), SB, UMA(jm, im), UMS(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), _
                beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4),
_
                THRS(jm, im, 5), THRCNT(jm, im))
  EET(jm, im) = EET(jm, im) + GGT(jm, im) + FuncETc(AAt(jm, im), BBt(jm, im), CCt(jm, im), DDt(jm, im), Trans(i), dx(im), dy(jm), dt, Hnat(jm, im),
hx(i), CT(jm, im), SIT, _
                ATC, AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im),
AqB(jm, im), SB, UMAT(jm, im), UMS(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), _
                beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRSt(jm, im, 1), THRSt(jm, im, 2), THRSt(jm, im, 3), THRSt(jm, im,
4), _
                THRSt(jm, im, 5), THRCNTt(jm, im), DEC(jm, im))
  b(jm, im) = b(jm, im) + FuncBBC(Qc(jm, im), Trans(i + r / 2), dx(im), dy(jm), dt, Hn(jm, im), hx(i + r / 2), CT(jm, im), SIT, _
                ATC, AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm,
im), SB, UMA(jm, im), UMS(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), _
                beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4),
_
                THRS(jm, im, 5), THRCNT(jm, im), ITS)
  bbtt(jm, im) = bbtt(jm, im) + FuncBTc(Qc(jm, im), Trans(i), dx(im), dy(jm), dt, Hnat(jm, im), hx(i), CT(jm, im), SIT, _
                ATC, AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im),
AqB(jm, im), SB, UMAT(jm, im), UMS(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), _
                beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRSt(jm, im, 1), THRSt(jm, im, 2), THRSt(jm, im, 3), THRSt(jm, im,
4), _
                THRSt(jm, im, 5), THRCNTt(jm, im), ITS, hoat(jm, im), DEC(jm, im))
  If bbtt(jm, im) = 9.99901110999 Then
       NegHead = 6
       NegHeadNM = nm + 38
        msgti = "Impulse Exceeds Aquitard Capacity. "
        flaw = 3
       GoTo Line50
  End If
  If hx(i + r / 2) <= AqTi(i) - UMA(jm, im) Then                                    'Aquitard flux limited (aquifer head drawn down below top).
    If hx(i) > AqTi(i) - UMA(jm, im) Then
         bbtt(jm, im) = bbtt(jm, im) - GG(jm, im) * ((AqT(jm, im) - UMA(jm, im)) - hx(i + r / 2))
         b(jm, im) = b(jm, im) + GG(jm, im) * ((AqT(jm, im) - UMA(jm, im)) - hx(i + r / 2))
    End If
  End If
   If CT(jm, im) = 2 Then
     If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
      If hx(i) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        EET(jm, im) = EET(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) / bpp(jm, im)
        bbtt(jm, im) = bbtt(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * SHD(jm, im) / bpp(jm, im)
        BedMat(jm, im) = 1
       ElseIf hx(i) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        If hx(i) > AqT(jm, im) - UMA(jm, im) Then
          bbtt(jm, im) = bbtt(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) /
bpp(jm, im)
          BedMat(jm, im) = 2
         ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
```

```
          b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) / bpp(jm,
im)
          BedMat(jm, im) = 5
        End If
       End If
      ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
      If hx(i + r / 2) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        EE(jm, im) = EE(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) / bpp(jm, im)                          'Includes full bed flux term.
        b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * SHD(jm, im) / bpp(jm, im)              'Includes full bed flux term.
        BedMat(jm, im) = 3
       ElseIf hx(i + r / 2) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
        b(jm, im) = b(jm, im) - kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (SHD(jm, im) + (bpp(jm, im) - Bed(jm, im)) + UMS(jm, im)) / bpp(jm, im)
'Bed flux limited (drawn down below bottom).
        BedMat(jm, im) = 4
       End If
     End If
      Call SidesC(side11t(jm, im), side12t(jm, im), side21t(jm, im), side22t(jm, im), side31t(jm, im), side32t(jm, im), _
                  side41t(jm, im), side42t(jm, im), side11a(jm, im), side12a(jm, im), side21a(jm, im), side22a(jm, im), _
                  side31a(jm, im), side32a(jm, im), side41a(jm, im), side42a(jm, im), Bed(jm, im), AqT(jm, im), AqTa, AqtB, _
                  AqTc, AqTd, SHD(jm, im), hx(i), hx(i + r / 2), hxat, hxbt, hxct, hxdt, dx(im), dy(jm), Lons(jm, im), _
                  widpp(jm, im), lenp(jm, im), hxa, hxb, hxc, hxd, widp(jm, im), bppb(jm, im), NBRS(jm, im, 1), NBRS(jm, im, 2), _
                  NBRS(jm, im, 3), NBRS(jm, im, 4), SEB)
        If side11a(jm, im) + side12a(jm, im) + side11t(jm, im) + side12t(jm, im) > 0 Then
        If Lons(jm, im) = 2 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
           SidMat(jm, im, 1, 1) = sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i))
           SidMat(jm, im, 2, 1) = sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2))
             csea(jm, im, 1) = sideEE(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side11a(jm, im),
side12a(jm, im))
             csba(jm, im, 1) = sideBB(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side11a(jm, im),
side12a(jm, im))
             cset(jm, im, 1) = sideEET(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side11t(jm, im),
side12t(jm, im))
             csbt(jm, im, 1) = sideBBtt(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side11t(jm, im),
side12t(jm, im))
           EE(jm, im) = EE(jm, im) + csea(jm, im, 1)
           b(jm, im) = b(jm, im) - csba(jm, im, 1)
           If hx(i) > AqT(jm, im) - UMA(jm, im) Then
             EET(jm, im) = EET(jm, im) + cset(jm, im, 1)
             bbtt(jm, im) = bbtt(jm, im) - csbt(jm, im, 1)
            ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
             b(jm, im) = b(jm, im) + (cset(jm, im, 1) * (AqT(jm, im) - UMA(jm, im)) - csbt(jm, im, 1))
           End If
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a1
         End If
         ElseIf Lons(jm, im) = 1 Then
Line9a1:     '=========================
           SidMat(jm, im, 1, 1) = sThreshT(UMAT(jm - 1, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm - 1, im), hxat)
           SidMat(jm, im, 2, 1) = sThreshQ(UMA(jm - 1, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm - 1, im), hxa)
             csea(jm, im, 1) = sideEE(AqTa, Bed(jm, im), UMA(jm - 1, im), SHD(jm, im), hxa, kppb(jm, im), bppb(jm, im), side11a(jm, im), side12a(jm,
im))
             csba(jm, im, 1) = sideBB(AqTa, Bed(jm, im), UMA(jm - 1, im), SHD(jm, im), hxa, kppb(jm, im), bppb(jm, im), side11a(jm, im), side12a(jm,
im))
             cset(jm, im, 1) = sideEET(AqTa, Bed(jm, im), UMAT(jm - 1, im), SHD(jm, im), hxat, kppb(jm, im), bppb(jm, im), side11t(jm, im),
side12t(jm, im))
```

```
            csbt(jm, im, 1) = sideBBtt(AqTa, Bed(jm, im), UMAT(jm - 1, im), SHD(jm, im), hxat, kppb(jm, im), bppb(jm, im), side11t(jm, im),
side12t(jm, im))
          EE(jm - 1, im) = EE(jm - 1, im) + csea(jm, im, 1)
          b(jm - 1, im) = b(jm - 1, im) - csba(jm, im, 1)
          If hxat > AqTa - UMA(jm - 1, im) Then
            EET(jm - 1, im) = EET(jm - 1, im) + cset(jm, im, 1)
            bbtt(jm - 1, im) = bbtt(jm - 1, im) - csbt(jm, im, 1)
           ElseIf hxat <= AqTa - UMA(jm - 1, im) Then
            b(jm - 1, im) = b(jm - 1, im) + (cset(jm, im, 1) * (AqTa - UMA(jm - 1, im)) - csbt(jm, im, 1))
          End If
         End If 'Lons
        End If
       If side21a(jm, im) + side22a(jm, im) + side21t(jm, im) + side22t(jm, im) > 0 Then
       If Lons(jm, im) = 1 Then
        If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
          SidMat(jm, im, 1, 2) = sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i))
          SidMat(jm, im, 2, 2) = sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2))
            csea(jm, im, 2) = sideEE(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side21a(jm, im),
side22a(jm, im))
            csba(jm, im, 2) = sideBB(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side21a(jm, im),
side22a(jm, im))
            cset(jm, im, 2) = sideEET(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side21t(jm, im),
side22t(jm, im))
            csbt(jm, im, 2) = sideBBtt(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side21t(jm, im),
side22t(jm, im))
          EE(jm, im) = EE(jm, im) + csea(jm, im, 2)
          b(jm, im) = b(jm, im) - csba(jm, im, 2)
          If hx(i) > AqT(jm, im) - UMA(jm, im) Then
            EET(jm, im) = EET(jm, im) + cset(jm, im, 2)
            bbtt(jm, im) = bbtt(jm, im) - csbt(jm, im, 2)
           ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
            b(jm, im) = b(jm, im) + (cset(jm, im, 2) * (AqT(jm, im) - UMA(jm, im)) - csbt(jm, im, 2))
           End If
         ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
          GoTo Line9a2
        End If
        ElseIf Lons(jm, im) = 2 Then
Line9a2:     '=========================
          SidMat(jm, im, 1, 2) = sThreshT(UMAT(jm, im - 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im - 1), hxbt)
          SidMat(jm, im, 2, 2) = sThreshQ(UMA(jm, im - 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im - 1), hxb)
            csea(jm, im, 2) = sideEE(AqtB, Bed(jm, im), UMA(jm, im - 1), SHD(jm, im), hxb, kppb(jm, im), bppb(jm, im), side21a(jm, im), side22a(jm,
im))
            csba(jm, im, 2) = sideBB(AqtB, Bed(jm, im), UMA(jm, im - 1), SHD(jm, im), hxb, kppb(jm, im), bppb(jm, im), side21a(jm, im), side22a(jm,
im))
            cset(jm, im, 2) = sideEET(AqtB, Bed(jm, im), UMAT(jm, im - 1), SHD(jm, im), hxbt, kppb(jm, im), bppb(jm, im), side21t(jm, im),
side22t(jm, im))
            csbt(jm, im, 2) = sideBBtt(AqtB, Bed(jm, im), UMAT(jm, im - 1), SHD(jm, im), hxbt, kppb(jm, im), bppb(jm, im), side21t(jm, im),
side22t(jm, im))
          EE(jm, im - 1) = EE(jm, im - 1) + csea(jm, im, 2)
          b(jm, im - 1) = b(jm, im - 1) - csba(jm, im, 2)
          If hxbt > AqtB - UMA(jm, im - 1) Then
            EET(jm, im - 1) = EET(jm, im - 1) + cset(jm, im, 2)
            bbtt(jm, im - 1) = bbtt(jm, im - 1) - csbt(jm, im, 2)
           ElseIf hxbt <= AqtB - UMA(jm, im - 1) Then
            b(jm, im - 1) = b(jm, im - 1) + (cset(jm, im, 2) * (AqtB - UMA(jm, im - 1)) - csbt(jm, im, 2))
           End If
```

```
          End If 'Lons
          End If
        If side31a(jm, im) + side32a(jm, im) + side31t(jm, im) + side32t(jm, im) > 0 Then
        If Lons(jm, im) = 2 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
           SidMat(jm, im, 1, 3) = sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i))
           SidMat(jm, im, 2, 3) = sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2))
             csea(jm, im, 3) = sideEE(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side31a(jm, im),
side32a(jm, im))
             csba(jm, im, 3) = sideBB(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side31a(jm, im),
side32a(jm, im))
             cset(jm, im, 3) = sideEET(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side31t(jm, im),
side32t(jm, im))
             csbt(jm, im, 3) = sideBBtt(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side31t(jm, im),
side32t(jm, im))
           EE(jm, im) = EE(jm, im) + csea(jm, im, 3)
           b(jm, im) = b(jm, im) - csba(jm, im, 3)
            If hx(i) > AqT(jm, im) - UMA(jm, im) Then
             EET(jm, im) = EET(jm, im) + cset(jm, im, 3)
             bbtt(jm, im) = bbtt(jm, im) - csbt(jm, im, 3)
             ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
             b(jm, im) = b(jm, im) + (cset(jm, im, 3) * (AqT(jm, im) - UMA(jm, im)) - csbt(jm, im, 3))
            End If
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a3
         End If
         ElseIf Lons(jm, im) = 1 Then
Line9a3:     '==========================
           SidMat(jm, im, 1, 3) = sThreshT(UMAT(jm + 1, im), Bed(jm, im), SHD(jm, im), AqT(jm + 1, im), AqT(jm, im), hxct)
           SidMat(jm, im, 2, 3) = sThreshQ(UMA(jm + 1, im), Bed(jm, im), SHD(jm, im), AqT(jm + 1, im), AqT(jm, im), hxc)
             csea(jm, im, 3) = sideEE(AqTc, Bed(jm, im), UMA(jm + 1, im), SHD(jm, im), hxc, kppb(jm, im), bppb(jm, im), side31a(jm, im), side32a(jm,
im))
             csba(jm, im, 3) = sideBB(AqTc, Bed(jm, im), UMA(jm + 1, im), SHD(jm, im), hxc, kppb(jm, im), bppb(jm, im), side31a(jm, im), side32a(jm,
im))
             cset(jm, im, 3) = sideEET(AqTc, Bed(jm, im), UMAT(jm + 1, im), SHD(jm, im), hxct, kppb(jm, im), bppb(jm, im), side31t(jm, im),
side32t(jm, im))
             csbt(jm, im, 3) = sideBBtt(AqTc, Bed(jm, im), UMAT(jm + 1, im), SHD(jm, im), hxct, kppb(jm, im), bppb(jm, im), side31t(jm, im),
side32t(jm, im))
           EE(jm + 1, im) = EE(jm + 1, im) + csea(jm, im, 3)
           b(jm + 1, im) = b(jm + 1, im) - csba(jm, im, 3)
            If hxct > AqTc - UMA(jm + 1, im) Then
             EET(jm + 1, im) = EET(jm + 1, im) + cset(jm, im, 3)
             bbtt(jm + 1, im) = bbtt(jm + 1, im) - csbt(jm, im, 3)
             ElseIf hxct <= AqTc - UMA(jm + 1, im) Then
             b(jm + 1, im) = b(jm + 1, im) + (cset(jm, im, 3) * (AqTc - UMA(jm + 1, im)) - csbt(jm, im, 3))
            End If
           End If 'Lons
         End If
        If side41a(jm, im) + side42a(jm, im) + side41t(jm, im) + side42t(jm, im) > 0 Then
        If Lons(jm, im) = 1 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
           SidMat(jm, im, 1, 4) = sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i))
           SidMat(jm, im, 2, 4) = sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2))
             csea(jm, im, 4) = sideEE(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side41a(jm, im),
side42a(jm, im))
```

```
            csba(jm, im, 4) = sideBB(AqT(jm, im), Bed(jm, im), UMA(jm, im), SHD(jm, im), hx(i + r / 2), kppb(jm, im), bppb(jm, im), side41a(jm, im),
side42a(jm, im))
            cset(jm, im, 4) = sideEET(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side41t(jm, im),
side42t(jm, im))
            csbt(jm, im, 4) = sideBBtt(AqT(jm, im), Bed(jm, im), UMAT(jm, im), SHD(jm, im), hx(i), kppb(jm, im), bppb(jm, im), side41t(jm, im),
side42t(jm, im))
          EE(jm, im) = EE(jm, im) + csea(jm, im, 4)
          b(jm, im) = b(jm, im) - csba(jm, im, 4)
          If hx(i) > AqT(jm, im) - UMA(jm, im) Then
            EET(jm, im) = EET(jm, im) + cset(jm, im, 4)
            bbtt(jm, im) = bbtt(jm, im) - csbt(jm, im, 4)
           ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
            b(jm, im) = b(jm, im) + (cset(jm, im, 4) * (AqT(jm, im) - UMA(jm, im)) - csbt(jm, im, 4))
          End If
         ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
          GoTo Line9a4
        End If
        ElseIf Lons(jm, im) = 2 Then
Line9a4:     '=========================
          SidMat(jm, im, 1, 4) = sThreshT(UMAT(jm, im + 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im + 1), hxdt)
          SidMat(jm, im, 2, 4) = sThreshQ(UMA(jm, im + 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im + 1), hxd)
            csea(jm, im, 4) = sideEE(AqTd, Bed(jm, im), UMA(jm, im + 1), SHD(jm, im), hxd, kppb(jm, im), bppb(jm, im), side41a(jm, im), side42a(jm,
im))
            csba(jm, im, 4) = sideBB(AqTd, Bed(jm, im), UMA(jm, im + 1), SHD(jm, im), hxd, kppb(jm, im), bppb(jm, im), side41a(jm, im), side42a(jm,
im))
            cset(jm, im, 4) = sideEET(AqTd, Bed(jm, im), UMAT(jm, im + 1), SHD(jm, im), hxdt, kppb(jm, im), bppb(jm, im), side41t(jm, im),
side42t(jm, im))
            csbt(jm, im, 4) = sideBBtt(AqTd, Bed(jm, im), UMAT(jm, im + 1), SHD(jm, im), hxdt, kppb(jm, im), bppb(jm, im), side41t(jm, im),
side42t(jm, im))
          EE(jm, im + 1) = EE(jm, im + 1) + csea(jm, im, 4)
          b(jm, im + 1) = b(jm, im + 1) - csba(jm, im, 4)
          If hxdt > AqTd - UMA(jm, im + 1) Then
            EET(jm, im + 1) = EET(jm, im + 1) + cset(jm, im, 4)
            bbtt(jm, im + 1) = bbtt(jm, im + 1) - csbt(jm, im, 4)
           ElseIf hxdt <= AqTd - UMA(jm, im + 1) Then
            b(jm, im + 1) = b(jm, im + 1) + (cset(jm, im, 4) * (AqTd - UMA(jm, im + 1)) - csbt(jm, im, 4))
          End If
         End If 'Lons
        End If
   End If                          'CT end if.
    i = i + 1
  Next im
Next jm
i = 1
 For jm = 1 To m
  For im = 1 To L
   If Trans(i) \ 100 = 1 Then                                          'Aquitard flux limited (aquitard dewatered).
    If (Trans(i) - 100) Mod 10 = THRCNTt(jm, im) Then
        Dewflag(i) = 1
        PermFlag(jm, im) = 1
        EET(jm, im) = PermFact(jm, im, 1) * dx(im) * dy(jm) / dt
        bbtt(jm, im) = -EET(jm, im) * (AqT(jm, im) - UMA(jm, im))
        If ITS = "Deep Percolation" Then
         If CT(jm, im) = 1 Then
            If Qc(jm, im) < 0 Then                                   'Water infiltrates to layer.
              b(jm, im) = b(jm, im) + Qc(jm, im)
```

```
              b(jm, im) = b(jm, im) + dvtc(i) / dt
             ElseIf Qc(jm, im) = 0 Then                                'Exfiltration case (>) residual not credited to lower layer.
              b(jm, im) = b(jm, im) + dvtc(i) / dt
            End If
           ElseIf CT(jm, im) <> 1 Then
            b(jm, im) = b(jm, im) + dvtc(i) / dt
          End If
         Else
           b(jm, im) = b(jm, im) + dvtc(i) / dt
        End If
   End If
  End If
   If CT(jm, im) = 4 Then
    GoTo Line9d
   ElseIf CT(jm, im) = 2 Then
   If SB = "Permissive" Then
Line9d:     '==========================
     AAt(jm, im) = 0
     BBt(jm, im) = 0
     CCt(jm, im) = 0
     DDt(jm, im) = 0
     GGT(jm, im) = 0
     EET(jm, im) = PermFact(jm, im, 1) * dx(im) * dy(jm) / dt
     bbtt(jm, im) = -EET(jm, im) * Hnpat(jm, im)
     PermFlag(jm, im) = 1
     If subsflag = 1 Then
      subsflag = 0
      GoTo line9e
     End If
   If CT(jm, im) = 4 Then
    GoTo Line9dd
   ElseIf Bed(jm, im) <= AqT(jm, im) Then
Line9dd:     '==========================
     AA(jm, im) = 0
     BB(jm, im) = 0
     CC(jm, im) = 0
     DD(jm, im) = 0
     GG(jm, im) = 0
     EE(jm, im) = PermFact(jm, im, 2) * dx(im) * dy(jm) / dt                    'Factor scales coefficients for procedure stability without
affecting result.
     b(jm, im) = -EE(jm, im) * Hnp(jm, im)
    ElseIf Bed(jm, im) > AqT(jm, im) Then
    End If
   End If                                   'SB end if.
   End If                                   'CT end if.
    If EET(jm, im) = 0 Then
      subsflag = 1
      GoTo Line9d
line9e:     '==========================
      bbtt(jm, im) = -EET(jm, im) * Hnat(jm, im)
      GoTo Line9f
    End If
    If AAt(jm, im) + BBt(jm, im) + CCt(jm, im) + DDt(jm, im) + GGT(jm, im) = 0 Then
     If Dewflag(i) <> 1 Then
              Dewflag(i) = 2
              If ITS = "Deep Percolation" Then
```

```
              If Qc(jm, im) < 0 Then                                    'Water infiltrates to layer.
                If Qc(jm, im) >= -1 * kvtb(i) Then                      'At negative unit hydraulic gradient, set limiting condition, (percolation
not keeping aquitard saturated).  [Bed flux not checked, as implicit, since it does not jump around like impulse can.]
                  b(jm, im) = b(jm, im) + Qc(jm, im)                        'Lands at PermFlag.
                 Else
                  GoTo Line9f
                End If
               ElseIf Qc(jm, im) = 0 Then                               'Lands at PermFlag.
               ElseIf Qc(jm, im) > 0 Then
                 GoTo Line9f
              End If
             End If
             PermFlag(jm, im) = 1
             EET(jm, im) = PermFact(jm, im, 1) * dx(im) * dy(jm) / dt
               If hx(i) <= AqTi(i) - UMA(jm, im) Then
                 bbtt(jm, im) = -EET(jm, im) * (AqTi(i) - UMA(jm, im))
                Else
                 bbtt(jm, im) = -EET(jm, im) * Hnat(jm, im)
               End If
     End If
    End If
Line9f:     '=========================
    If PFSflag < 1 Then
     If MaxTerm < Abs(EE(jm, im)) Then
      MaxTerm = Abs(EE(jm, im))
     End If
     If MaxTermT < Abs(EET(jm, im)) Then
      MaxTermT = Abs(EET(jm, im))
     End If
    End If
    i = i + 1
  Next im
Next jm

If PFSflag < 1 Then
   PFSflag = 1
End If

'
'WRITE AUGMENTED MATRIX BY POPULATING COEFFICIENT AND INTERCEPT VALUES FOR SYSTEM OF EQUATIONS OF IMPLICIT REPRESENTATION
'-------------------------------------------------------------------------------------------------------------------------------
ReDim SE(1 To r, 1 To 7), SEC(1 To r), UM(1 To r, 1 To 4), UT(1 To r, 1 To 4), DU(1 To r, 1 To 4) 'Redimension matrices to obviate memory errors(row
limit, column limit).
i = 1                               'Initialize system of equations row index, also index for model cell identification serial.
For jm = 1 To m                             'Initialize model space row index.  Note: convention for matrix indeces is different than model.  Model
columns are im, Matrix columns are j.
  For im = 1 To L                           'Initialize and reset model space column index.
   SE(r / 2 + i, 4) = EE(jm, im)         'Coefficient for cell's own next-step head.
   SE(i, 4) = EET(jm, im)
   SEC(r / 2 + i) = -1 * b(jm, im)             'Constant or intercept value for row.
   SEC(i) = -1 * bbtt(jm, im)
   If m > 1 Then
   If L > 1 Then
   If jm + 1 < m + 1 Then
    SE(r / 2 + i, 6) = -1 * CC(jm, im)           'Coefficient for next-step head in model cell beneath, matrix place L to the right.
    SE(i, 6) = -1 * CCt(jm, im)       'Aquitard cell overlying main model.
```

```
   End If
   If jm - 1 > 0 Then
    SE(r / 2 + i, 2) = -1 * AA(jm, im)           'Coefficient for next-step head in model cell above, matrix place L to the left.
    SE(i, 2) = -1 * AAt(jm, im)
   End If
   End If
   End If
   If L > 1 Then
    If im + 1 < L + 1 Then
     SE(r / 2 + i, 5) = -1 * DD(jm, im)           'Coefficient for next-step head in model cell to right, matrix place one to the right.
     SE(i, 5) = -1 * DDt(jm, im)
    End If
    If im - 1 > 0 Then
     SE(r / 2 + i, 3) = -1 * BB(jm, im)           'Coefficient for next-step head in model cell to the left, matrix place one to the left.
     SE(i, 3) = -1 * BBt(jm, im)
    End If
   Else
    If jm + 1 < m + 1 Then
     SE(r / 2 + i, 5) = -1 * CC(jm, im)           'Coefficient for next-step head in model cell beneath, matrix place one to the right.
     SE(i, 5) = -1 * CCt(jm, im)
    End If
    If jm - 1 > 0 Then
     SE(r / 2 + i, 3) = -1 * AA(jm, im)           'Coefficient for next-step head in model cell above, matrix place one to the left.
     SE(i, 3) = -1 * AAt(jm, im)
    End If
   End If
     SE(r / 2 + i, 1) = -1 * GG(jm, im)
     SE(i, 7) = -1 * GGT(jm, im)
      If PermFlag(jm, im) = 1 Then
       If PFSflag < 2 Then
        If SE(i, 4) < 0.75 * MaxTermT Then
         PermFact(jm, im, 1) = PermFact(jm, im, 1) * WorksheetFunction.RoundUp(0.75 * MaxTermT / SE(i, 4), 0) 'Increment factor to recompute
coefficient for mathematical stability.
         subsflag = 1
        End If
        If CT(jm, im) = 4 Then
         GoTo line9ff
        ElseIf SB = "Permissive" Then
        If Bed(jm, im) <= AqT(jm, im) Then
line9ff:     '=========================
         If SE(i + r / 2, 4) < 0.75 * MaxTerm Then
          PermFact(jm, im, 2) = PermFact(jm, im, 2) * WorksheetFunction.RoundUp(0.75 * MaxTerm / SE(i + r / 2, 4), 0) 'Increment factor to recompute
coefficient for mathematical stability.
          subsflag = 1
         End If
        End If
        End If
       End If
     End If
   i = i + 1
  Next im
Next jm
If subsflag = 1 Then
 PFSflag = 2
 subsflag = 0
 GoTo Line9
```

```
End If

Line9g:     '=========================
For i = 1 To r
 UM(i, 1) = (1 + delta) * SE(i, 4)           'Modified incomplete Cholesky factorization upper conditioner matrix.
Next i                                            'Initial loop sets all uii values to aii, unless delta non-zero.
For i = 1 To r
 If i - 1 > 0 Then
  UM(i, 1) = UM(i, 1) - (SE(i - 1, 5) ^ 2) / UM(i - 1, 1)                                    'Xi, Hill pg. 10.
  UM(i, 1) = UM(i, 1) - SE(i - 1, 5) * SE(i - 1, 7) / UM(i - 1, 1)                           'Theta.
     If i + r / 2 - 1 <= r Then
       UM(i + r / 2 - 1, 1) = UM(i + r / 2 - 1, 1) - SE(i - 1, 5) * SE(i - 1, 7) / UM(i - 1, 1)
     End If
  UM(i, 1) = UM(i, 1) - SE(i - 1, 5) * SE(i - 1, 6) / UM(i - 1, 1)                           'Phi.
     If i + L - 1 <= r Then
       UM(i + L - 1, 1) = UM(i + L - 1, 1) - SE(i - 1, 5) * SE(i - 1, 6) / UM(i - 1, 1)
     End If
  If i - L > 0 Then
   UM(i, 1) = UM(i, 1) - (SE(i - L, 6) ^ 2) / UM(i - L, 1)                                   'Tau.
   UM(i, 1) = UM(i, 1) - SE(i - L, 6) * SE(i - L, 7) / UM(i - L, 1)                          'Psi.
     If i + r / 2 - L <= r Then
       UM(i + r / 2 - L, 1) = UM(i + r / 2 - L, 1) - SE(i - L, 6) * SE(i - L, 7) / UM(i - L, 1)
     End If
  End If
  If i - r / 2 > 0 Then
   UM(i, 1) = UM(i, 1) - (SE(i - r / 2, 7) ^ 2) / UM(i - r / 2, 1)                           'Gamma.
  End If
 End If
 If UM(i, 1) < 0.001 Then
   If delta < 1000000 Then
     delta = 1.5 * delta + 0.001
     GoTo Line9g
   End If
 End If
 If i < r Then
  UM(i, 2) = SE(i, 5)
 End If
 If i <= r - L Then
  UM(i, 3) = SE(i, 6)
 End If
 If i <= r - r / 2 Then
  UM(i, 4) = SE(i, 7)
 End If
 DU(i, 1) = 1 'UM(i, 1) * (1 / UM(i, 1)) 'Diagonal matrix of 1/UM(i,1) times UM.
  DU(i, 2) = UM(i, 2) / UM(i, 1)
  DU(i, 3) = UM(i, 3) / UM(i, 1)
  DU(i, 4) = UM(i, 4) / UM(i, 1)
Next i

For i = 1 To r                          'Transpose of UM carried out for condensed format.
UT(i, 4) = UM(i, 1)
If i - 1 > 0 Then
UT(i, 3) = UM(i - 1, 2)
End If
If i - L > 0 Then
UT(i, 2) = UM(i - L, 3)
```

```
End If
If i - r / 2 > 0 Then
UT(i, 1) = UM(i - r / 2, 4)
End If
Next i

'SIMULTANEOUS SOLUTION OF SYSTEM OF EQUATIONS
'-----------------------------------------------------------------------------------------
'Modified Incomplete Cholesky Preconditioned CONJUGATE GRADIENT METHOD (MICCG)!!
'SEE "Preconditioned Conjugate-Gradient 2 (PCG2), A Computer Program for Solving Ground-Water flow Equations"
'by Mary C. Hill, USGS Water Resources Investigations Report 90-4048. 1990.
kitri = 0                     'Initialize iteration counter.
maxerrp = 1E+99               'Initialize prior iteration closure.
maxerr = 1                    'Initialize iteration closure.
GTCR = 0                      'Initialize GoTo Code for Residual Rounding.
trigger = 0                   'Initialize head adjustment trigger.
GTCRT = 0                     'Initialize trigger goto code.
Line10: '=========================       'Line label for top of solver.
ReDim Preserve SE(1 To r, 1 To 7), SEC(1 To r), hx(1 To r)
'Initialize Ax, rr, zz.
'-----------------quick homemade array multiplication
ReDim Ax(1 To r)
For i = 1 To r
Ax(i) = SE(i, 4) * hx(i)
If m > 1 Then
If L > 1 Then
 If i - L > 0 Then
  Ax(i) = Ax(i) + SE(i, 2) * hx(i - L)
 End If
 If i + L < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 6) * hx(i + L)
 End If
End If
End If
 If i - 1 > 0 Then
  Ax(i) = Ax(i) + SE(i, 3) * hx(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 5) * hx(i + 1)
 End If
 If i - r / 2 > 0 Then
  Ax(i) = Ax(i) + SE(i, 1) * hx(i - r / 2)
 End If
 If i + r / 2 < r + 1 Then
  Ax(i) = Ax(i) + SE(i, 7) * hx(i + r / 2)
 End If
 Next i
'-----------------
ReDim Preserve Ax(1 To r)
ReDim rr(1 To r)
For i = 1 To r          'method step
rr(i) = SEC(i) - Ax(i)
Next i
          If GTCR = 2 Then
            GoTo Line14
           ElseIf GTCRT = 2 Then
```

```
           GoTo Line14
         End If
Do While maxerr > eps                        '#############Top of Method loop.
If kitri < 1000 Then                          'Iteration limit check.
Line11: '==========================
ReDim Preserve UT(1 To r, 1 To 4), rr(1 To r)
ReDim vv(1 To r)
For i = 1 To r          'forward sub
vv(i) = rr(i)
  If i - r / 2 > 0 Then
   vv(i) = vv(i) - UT(i, 1) * vv(i - r / 2)
  End If
  If L > 1 Then
   If m > 1 Then
    If i - L > 0 Then
      vv(i) = vv(i) - UT(i, 2) * vv(i - L)
    End If
   End If
  End If
  If i - 1 > 0 Then
   vv(i) = vv(i) - UT(i, 3) * vv(i - 1)
  End If
vv(i) = vv(i) / UT(i, 4)
Next i
ReDim Preserve DU(1 To r, 1 To 4), vv(1 To r)
ReDim zz(1 To r)
i = r
Do While i > 0           'back sub
zz(i) = vv(i)
  If i + r / 2 < r + 1 Then
   zz(i) = zz(i) - DU(i, 4) * zz(i + r / 2)
  End If
  If L > 1 Then
   If m > 1 Then
    If i + L < r + 1 Then
      zz(i) = zz(i) - DU(i, 3) * zz(i + L)
    End If
   End If
  End If
  If i + 1 < r + 1 Then
   zz(i) = zz(i) - DU(i, 2) * zz(i + 1)
  End If
'zz(i) = zz(i) / DU(i, 1)  DU(i,1)=1, so no need to divide.
i = i - 1
Loop
ReDim Preserve zz(1 To r)
          If GTCR = 1 Then
            GoTo Line12
          End If
If kitri = 0 Then
Line12: '==========================
 ReDim pp(1 To r)
 nip = 0
 For i = 1 To r
 pp(i) = zz(i)
 nip = nip + zz(i) * rr(i)
```

```
 Next i
           If GTCR = 1 Then
            ReDim Preserve pp(1 To r)
            GoTo Line13
           End If
ElseIf kitri > 0 Then
 dip = nip
 nip = 0
 For i = 1 To r
 nip = nip + zz(i) * rr(i)
 Next i
 beta = nip / dip
 For i = 1 To r
 pp(i) = zz(i) + beta * pp(i)
 Next i
End If
ReDim Preserve pp(1 To r)
kitri = kitri + 1         'Update iteration counter.
'-----------------array multiplication
Line13: '==========================
ReDim Preserve SE(1 To r, 1 To 7)
ReDim Ap(1 To r)
For i = 1 To r
Ap(i) = SE(i, 4) * pp(i)
If m > 1 Then
If L > 1 Then
 If i - L > 0 Then
  Ap(i) = Ap(i) + SE(i, 2) * pp(i - L)
 End If
 If i + L < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 6) * pp(i + L)
 End If
 End If
 End If
 If i - 1 > 0 Then
  Ap(i) = Ap(i) + SE(i, 3) * pp(i - 1)
 End If
 If i + 1 < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 5) * pp(i + 1)
 End If
 If i - r / 2 > 0 Then
  Ap(i) = Ap(i) + SE(i, 1) * pp(i - r / 2)
 End If
 If i + r / 2 < r + 1 Then
  Ap(i) = Ap(i) + SE(i, 7) * pp(i + r / 2)
 End If
 Next i
 ReDim Preserve Ap(1 To r)
'-----------------
dip = 0                   'Remainder of method steps
For i = 1 To r            'For-Next gives internal products.
If Abs(nip) < 1E+100 Then      'Confirm stability.
dip = dip + pp(i) * Ap(i)
Else
 NegHead = 3
 NegHeadNM = nm + 38      'Intervene if not stable.
```

```
 GoTo Line50
End If
Next i
If nip <> 0 Then          'Zero residual check.
alpha = nip / dip
ReDim xp(1 To r)
For i = 1 To r            'Calculate next estimates.
xp(i) = hx(i)
hx(i) = hx(i) + alpha * pp(i)
rr(i) = rr(i) - alpha * Ap(i)
Next i
ReDim Preserve rr(1 To r), xp(1 To r), hx(1 To r)
           If GTCR = 1 Then
            GTCR = 2
            GoTo Line10
           ElseIf GTCR = 2 Then
            GoTo Line10
           End If
Line14: '==========================
Else              'Zero residual upshot.
 ReDim xp(1 To r)
 For i = 1 To r
  xp(i) = hx(i)
  hx(i) = hx(i)
  rr(i) = rr(i)
 Next i
End If            'Zero residual end if.

If trigger = 1 Then
 trigger = 0
 GTCRT = 0
 GoTo Line14a
End If
For i = 1 To r / 2
 If hx(i) < AqTi(i) - UMAi(i + r / 2) Then
  If -1 * dvtc(i) / dt > kvtb(i) Then
    hx(i) = (AqTi(i) - UMAi(i + r / 2)) + ((-1 * dvtc(i) / dt) - kvtb(i)) * dt / CAS(i)
   Else
    hx(i) = AqTi(i) - UMAi(i + r / 2)
  End If
  trigger = 1
 End If
Next i
If trigger = 1 Then
   GTCRT = 2
   GoTo Line10
End If
Line14a: '==========================

maxerr = 0
maxdelta = 0
For i = 1 To r                'Calculate closure difference.
  If Abs(hx(i) - xp(i)) > maxerr Then
    maxerr = Abs(hx(i) - xp(i))
  End If
  If Abs(hx(i) - xp(i)) > maxdelta Then
```

```
    maxdelta = Abs(hx(i) - xp(i))
  End If
  If Abs(rr(i)) > maxerr Then
    maxerr = Abs(rr(i))
  End If
Next i
If maxdelta < 0.00000000001 Then
 If maxerr > eps Then
   If kitro + kitrtr < 100 Then
     maxerr = 0.5 * eps
     If kitri = 1 Then
      kitri = 2
     End If
    Else
     resord = -1 * WorksheetFunction.RoundDown(WorksheetFunction.Log10(maxerr), 0)
     eps = 1 * 10 ^ (-resord)
     msgres = "Residual Order = " & resord & ". "
   End If
 End If
End If

    If maxerrp <= maxerr Then                   'Watches for rounding error; if error evident, recalculates residual, instead of using erroneous
values.
    If GTCR = 0 Then
        GTCR = 1
        For i = 1 To r
            hx(i) = xp(i)
        Next i
        GoTo Line10
     ElseIf kitri > 50 Then
        subsflag = 76
    End If
    End If

Line15: '==========================

TransT = 0
Call TransitionC(r, Trans(), Hniat(), Hni(), hx(), THRSt(), THRCNTt(), THRS(), THRCNT(), m, L, AqT(), UMA(), UMAT(), CTi())
For i = 1 To r / 2
    If Trans(i) <> transp(i) Then
     TransT = 1
    End If
    If Trans(i + r / 2) <> transp(i + r / 2) Then
     TransT = 1
    End If
Next i
If subsflag = 1 Then
 subsflag = 0
 GoTo Line48
End If
If subsflag = 76 Then
  subsflag = 0
  GoTo Line16
End If
If subsflag = 77 Then
  subsflag = 0
```

```
  GoTo Line17
End If

If kitro + kitrtr < 200 Then
If kitri > 100 Then
Line15a: '==========================

If TransT = 1 Then
 GoTo Line16
End If

i = 1
For jm = 1 To m
For im = 1 To L
       If CT(jm, im) <> 4 Then
       If hx(i + r / 2) <= AqBi(i) Then
        GoTo Line16
       End If
       If ITS = "Deep Percolation" Then
        If hx(i) <= AqT(jm, im) - UMA(jm, im) Then
         GoTo Line16
        End If
       End If
       End If
i = i + 1
Next im
Next jm

Call KickOutC(r, m, L, CT(), xpo(), hx(), rpc(), SB, SIT, _
            UMS(), BedMat(), UMA(), AqT(), Bed(), side11a(), side12a(), _
            side11t(), side12t(), side21a(), side22a(), side21t(), side22t(), _
            side31a(), side32a(), side31t(), side32t(), side41a(), side42a(), _
            side41t(), side42t(), Lons(), widpp(), bpp(), bppb(), widp(), _
            SidMat(), UMAT(), SHD(), KOC, AqBi(), ITS)
If KOC = 1 Then
   GoTo Line16
End If

End If                           'kitri.
Else
If kitri > 4 Then
GoTo Line15a
End If
End If

maxerrp = maxerr  'Set prior maxerr.

Else                                                 'Intervene to end iteration if not converging in a reasonable number of cycles.
 NegHead = 3
 NegHeadNM = nm + 38
 GoTo Line50
End If                                               'Iteration counter end if.

Loop                                        '##############Bottom of method loop.

Line16: '==========================
```

```
For i = 1 To r / 2
       If CTi(i) Mod 2 = 1 Then
        If hx(i + r / 2) <= AqBi(i) Then         'Check for dewatering.
          If kitro + kitrtr > 500 Then
           NegHead = 2
           NegHeadNM = nm + 38
           GoTo Line50
          Else
            If (Hni(i + r / 2) - AqBi(i)) > 0.2 * (AqTi(i) - AqBi(i)) Then
              hx(i + r / 2) = AqBi(i) + 0.2 * (AqTi(i) - AqBi(i))
             Else
              hx(i + r / 2) = Hni(i + r / 2)
            End If
            maxerr = 2 * eps
          End If
        End If
        If ITS = "Deep Percolation" Then
          If hx(i) <= AqTi(i) - UMAi(i + r / 2) Then
             If hx(i + r / 2) > AqTi(i) - UMAi(i + r / 2) Then
              If kitro + kitrtr < 500 Then
                hx(i) = hx(i + r / 2)
                maxerr = 2 * eps
               Else
                GoTo Line16a
              End If
              Else
Line16a: '==========================
               msgti = "Impulse Exceeds Aquitard Capacity. "
               flaw = 3
               NegHead = 46
               NegHeadNM = nm + 38
               GoTo Line50
             End If
            End If
          End If
        End If
Next i
If maxerr = 2 * eps Then
  subsflag = 77
  GoTo Line15
End If

Line17: '==========================
ReDim ho(1 To m, 1 To L), hoat(1 To m, 1 To L)
ReDim hob(1 To m, 1 To L, 1 To 4), hobat(1 To m, 1 To L, 1 To 4)  'Thickness update.
Call HoHo3(hx(), m, L, jm, im, ovp(), ovw(), beds(), Bed(), hob(), ho(), widpp(), Lons(), widp(), bppb(), bpp(), dx(), dy(), _
          SIT, AqT(), AqB(), UMA(), UMS(), ATC, CT(), lenp(), SB)
Call HoHoat(i, r, hx(), Hn(), m, L, jm, im, ovp(), ovw(), beds(), hobat(), hoat(), widpp(), Lons(), widp(), bppb(), dx(), dy(), SIT, _
            AqT(), AqB(), UMAT(), UMS(), DEC(), CT(), lenp(), Bed(), bpp(), SB, rpc(), UMA())
If subsflag = 1 Then
  subsflag = 0
  GoTo Line9
End If

Line17b: '==========================
```

```
ReDim Dewflag(1 To r / 2)
i = 1
For jm = 1 To m
For im = 1 To L
    If Trans(i) <> transp(i) Then
     TransC(i) = TransC(i) + 1
    End If
    If Trans(i + r / 2) <> transp(i + r / 2) Then
     TransC(i + r / 2) = TransC(i + r / 2) + 1
    End If
      If TransC(i + r / 2) = 20 Then
       If Gflag(jm, im) = 0 Then
        Gflag(jm, im) = -1
       End If
      ElseIf TransC(i + r / 2) = 40 Then
       If Gflag(jm, im) < 1 Then
        Gflag(jm, im) = 1
        If xp(i) > AqTi(i) - UMAi(i + r / 2) Then
         GoTo Line17c
        ElseIf hx(i) > AqTi(i) - UMAi(i + r / 2) Then
Line17c: '=========================
        'flaw = 5
        If msgtg = "" Then
         If kvq(jm, im) <> 999999.12321 Then
          msgtg = "Vertical Flux Distance Modulation. "
         End If
        End If
        End If
       End If
       End If
  xpo(i) = hx(i)
  xpo(i + r / 2) = hx(i + r / 2)
transp(i) = Trans(i)                                    'Set prior transition for next coefficient cycle.
transp(i + r / 2) = Trans(i + r / 2)
i = i + 1
Next im
Next jm

If TransT > 0 Then
  If kitrtr < 1000 Then
     kitrtr = kitrtr + 1
     GoTo Line9
    Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
 ElseIf maxerr > eps Then
  If kitro < 1000 Then
     kitro = kitro + 1
     GoTo Line9
   Else
     NegHead = 3
     NegHeadNM = nm + 38
     GoTo Line50
  End If
```

```
End If

If GTC = 40 Then
GoTo Line40
End If
'---------------------------------------------------------------------------------------------
'END SIMULTANEOUS SOLUTION by MICCG!!
'
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Recursive portions of solution follow initial solution for time step.
  '##########Begin outside iteration (recursive solution loop) for aquifer thickness.
Line31: '==========================

Do While kitri > 1   '----------------Top of outside iteration loop, continues until convergence on first inner iteration after thickness update.
If kitro < 1000 Then                    'Outside iteration limit check if.
kitro = kitro + 1
  GTC = 40
  GoTo Line9    'Recompute coefficients from new head estimates, from there, repeat simultaneous solution by MICCG.
Line40: '==========================
 Else                                                     'Intervene to end outside iteration if not converging in a reasonable number of cycles.
  NegHead = 3
  NegHeadNM = nm + 38
  GoTo Line50
 End If                                                   'Outside iteration counter end if.
Loop               '----------------Bottom of outside iteration loop.
   '##########End outside iteration for aquifer thickness.
'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Last of recursion.

Line45: '==========================
Call FortyFiveC(AqT(), CT(), DEC(), flaw, Hni(), Hniat(), Hnn(), Hnnat(), Hno(), Hnoat(), _
               ho(), hoat(), hoo(), hx(), ic, IMS, ITS, msgat, msgc, msgmt, msgt, msgtg, _
               msgti, msgumat, MSTA, nc, nm, o, Qc(), QCP(), Qm(), qs, Trans(), Transnm(), _
               Transpn(), Transpnm(), m, L, r, NegHead, NegHeadNM, subsflag, Dewflag())
If subsflag = 50 Then
 subsflag = 0
 GoTo Line50
End If

'
'OUTPUT  (starts before end of time step loop)
'-------------------------------------------------------------------
RSP(n) = 0                          'Prepare for response flow calculation...............................................................
RSPA(n) = 0
ReDim Qnno(1 To m, 1 To L)
 Call QnoutC(RSP(), RSPA(), Qnno(), i, jm, im, m, L, CT(), SB, Bed(), bpp(), AqT(), hx(), UMS(), kpp(), widpp(), lenp(), SHD(), msgbg, _
            SIT, Hnnat(), Hnn(), csbt(), csba(), cset(), csea(), CC(), DD(), AA(), BB(), GG(), UMA(), AqTi(), CCt(), _
            DDt(), AAt(), BBt(), zone(), qrs, dt, UCF, n, qnoc(), r, widp(), bppb(), Lons())

subsflag = 1                           'Independent domain budget block.
GoTo Line15
Line48: '==========================
Qtt = qrs * Qtt                        'Volume into aquifer during step.
RSP(n) = (-1) * UCF * RSP(n)           'Volume out.
dS = 0
i = 1
For jm = 1 To m
  For im = 1 To L                      'Change in storage.
```

```
    dS = dS + FuncdVC(Trans(i + r / 2), dx(im), dy(jm), dt, (Hno(jm, im) + UMA(jm, im)), (Hnn(jm, im) + UMA(jm, im)), CT(jm, im), SIT, ATC, AqT(jm,
im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), _
                      bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), SB, UMA(jm, im), UMS(jm, im), _
                      ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im,
3), beds(jm, im, 3), _
                      ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm,
im, 5), THRCNT(jm, im))
    dS = dS + FuncdVatC(Trans(i), dx(im), dy(jm), dt, (Hnoat(jm, im) + UMAT(jm, im)), (Hnnat(jm, im) + UMAT(jm, im)), CT(jm, im), SIT, ATC, AqT(jm,
im), widp(jm, im), _
                        lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im), _
                        AqB(jm, im), SB, UMAT(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), _
                        ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), _
                        beds(jm, im, 4), THRSt(jm, im, 1), THRSt(jm, im, 2), THRSt(jm, im, 3), THRSt(jm, im, 4), THRSt(jm, im, 5), THRCNTt(jm, im),
DEC(jm, im))
    i = i + 1
  Next im
Next jm
dS = qs * dS
BD = Abs(Qb - (dS + RSP(n)))          'Budget difference: volume in vs. change in storage plus volume out.
BLT = BLTF(Qb, dS, RSP(n), ic, Qm(nm), UCF, nc) 'Largest term.
If BLT > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then 'Minimum scale for evaluation.
If BD > 0.0001 * UCF * (Abs(IMPSMX) + Abs(IMPSMN)) / 2 / deno Then
If 100 * BD / BLT > 0.1 Then 'Difference divided by largest term, converted to percentage, compared to acceptance level...1/10 of 1%.
 If msgc <> "Moist Surface! " Then
   NegHeadNM = nm + 38
   NegHead = 5
   If Abs(IMPSMX) + Abs(IMPSMN) > 0.00001 Then
     msgep = "Budget Gap. "
    Else
     msgep = "No Volume Budget. "
   End If
   GoTo Line50
  End If
 End If
 End If
 End If
RSP(n) = RSP(n) / ((-1) * UCF)
Qtt = Qtt / qrs

   If HTII > 0 Then                'Stash head to plot.
    If o = nc Then
         HTS(nm) = -1 * Hnn(HTIJ, HTII)
         HTS2(nm) = -1 * Hnnat(HTIJ, HTII)
    End If
   End If
If nm = nh Then
  If o = nc Then
   For im = 1 To L
    For jm = 1 To m
      Hnnp(jm, im) = -1 * Hnn(jm, im)
      Hnnpat(jm, im) = -1 * Hnnat(jm, im)
      If Hnnpat(jm, im) = -(AqT(jm, im) - UMA(jm, im)) Then
       ATDF = 1
      End If
    Next jm
  Next im
```

```
 End If
End If

n = n + 1                        'TIME STEP PROGRESSION.
o = o + 1
If o = nc + 1 Then
 o = 1
 nm = nm + 1
End If

For im = 1 To L
 For jm = 1 To m
   Hn(jm, im) = Hnn(jm, im)      'UPDATE HEAD.
   Hno(jm, im) = Hn(jm, im)
   Hnat(jm, im) = Hnnat(jm, im)
   Hnoat(jm, im) = Hnat(jm, im)
 Next jm
Next im

Loop                              'Time step loop!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                  'END OF TIME STEP COMPUTATION LOOP (continue output section)
Line50: '=========================

 Sheet10.Unprotect Password:=CPC
 Sheet10.Cells(133, 2).Value = "Head Time Series at Node"
 If HTII > 0 Then
   Sheet10.Cells(134, 7).Value = HTII
   Sheet10.Cells(135, 7).Value = HTIJ
    If NegHead > 1 Then
       hrb = NegHeadNM - 38 + 140 - 1
      Else
       hrb = no + 140
    End If
       Sheet10.Range("c140:" & "c" & hrb).Value = WorksheetFunction.Transpose(HTS())        'Output head time series.
       Sheet10.Range("d140:" & "d" & hrb).Value = WorksheetFunction.Transpose(HTS2())
   Else
   Sheet10.Cells(135, 3).Value = "To plot an output head time series, please specify, on the Time tab, non-zero node indeces for the desired
location.  Thank you!"
 End If
     Sheet10.Cells(72, 2).Value = "Aquifer Head Surface Output"
     Sheet10.Cells(72, 55).Value = "Aquitard Head Surface Output"
If nh > 0 Then
  Sheet10.Cells(73, 7).Value = nh
If nh < no + 1 Then
hrb = 76 + m
hrr = crf(L)
hrr2 = crf2(L)
If msgsep = "2" Then
  Sheet10.Range("bc74").Value = 1
  Else
  Sheet10.Range("bc74").Value = 0
End If
Sheet10.Range("bc73").Value = ATDF
Sheet10.Range("c77:" & hrr & hrb).Value = Hnnp()        'Output head to plot.
Sheet10.Range("bd77:" & hrr2 & hrb).Value = Hnnpat()
Else
```

```
GoTo Line60
End If
Else
Line60: '=========================
Sheet10.Cells(75, 3).Value = "To plot head, please specify, on the Time tab, a period number for plotting that is greater than zero and less than or
equal to the number of simulation periods.  Thank you!"
End If
  If NegHead > 1 Then
  If NegHeadNM - 38 - 1 < nh Then
  hrb = 76 + m
  hrr = crf(L)
  hrr2 = crf2(L)
  Sheet10.Range("c77:" & hrr & hrb).ClearContents
  Sheet10.Range("bd77:" & hrr2 & hrb).ClearContents
  End If
  End If
 Sheet10.Protect Password:=CPC

'Compute Impulse and Response Outputs and Totals.
Call IROT(RSPT, RSPTA, IMPST, n, nm, nn, o, RSPO(), RSPOA(), nc, RSP(), RSPA(), Qm(), IMPSMX, IMPSMN)

Sheet11.Unprotect Password:=CPC

hrb = 38 + no
Sheet11.Cells(6, 22).Value = flaw               'Flag indicates whether oscillation flaw or no.
Sheet11.Range("e29").ClearContents              'Message prep.
   If msgh222(msgim2, msgti, msgep, NegHead) > 1 Then
   Sheet11.Cells(6, 22).Value = msgh222(msgim2, msgti, msgep, NegHead)
   End If
If hrb222(msgim2, msgti, msgep, NegHead, NegHeadNM) > 0 Then
hrb = hrb222(msgim2, msgti, msgep, NegHead, NegHeadNM)
End If
   msgh = msgpb & msgh
   msgtt = msgtt & FuncNegTT(NegHead)
If msgep = "Budget Gap. " Then
Sheet11.Cells(10, 10).Value = epse
Else
Sheet11.Cells(10, 10).ClearContents
End If
Sheet11.Cells(10, 11).Value = fun1011(NegHead, msgres, epse)

If NegHead = 7 Then
hrb = NegHeadNM
End If

If hrb > 38 Then
Sheet11.Range("e39:" & "e" & hrb).Value = WorksheetFunction.Transpose(RSPOA())     'Output response time series, Zone 1, then total.
Sheet11.Range("f39:" & "f" & hrb).Value = WorksheetFunction.Transpose(RSPO())
 If IMPSMX <> 999999999.123457 Then
  Else
  msgh = "Impulse Alternation. "
 End If
End If

Call SOB(St, PD, no, NegHead, hrb, RSPOFM, CPC, RSPO(), _
Qm(), IMPST, IMPSMX, IMPSMN, RSPT)
```

```
Else                                    'Thickness input else.
Sheet11.Unprotect Password:=CPC
   msgh = msghfun(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1))
   If msgh22(TipW(5), TipW(4), TipW(3), TipW(2), TipW(1)) = 2 Then
   Sheet11.Cells(6, 22).Value = 2
   End If
End If                                  'End thickness input if.
   If TipW(3) = 1 Then
   msgtp = "Principal Aquifer Top at or above Initial Piezometric Surface. "
   End If

Sheet11.Protect Password:=CPC

Call Messenger(msgc, msgtp, msgim, msgim2, msgt, msgti, msgbg, msgat, msgtt, msgh, msgres, msg, msgmt, msgtg, msgep, CPC, msgu, msgumat, msgclt,
msgwp, msgsep, "", "")

End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'Subroutine to load initial parameter values.
Sub Load(RSPOFM2 As Integer, SB As String, SIT As String, IMS As String, im As Integer, L As Integer, jm As Integer, m As Integer, msgwp As String, _
         dx() As Double, dy() As Double, CT() As Integer, widp() As Double, widpp() As Double, lenp() As Double, Lons() As Integer, AqT() As Double,
_
         kv() As Double, kh() As Double, kvq() As Double, LK As Integer, Sigma() As Double, Sigs() As Double, AqB() As Double, DEC() As Double, _
         k() As Double, zone() As String, Sy() As Double, Ss() As Double, Hn() As Double, Hno() As Double, Hnp() As Double, Hnat() As Double, _
         Hnoat() As Double, Hnpat() As Double, UMA() As Double, UMAT() As Double, kpp() As Double, bpp() As Double, _
         SHD() As Double, Bed() As Double, msgpb As String, msgh As String, TipW() As Integer, UMS() As Double, WR() As Double, _
         WRAT() As Double, Qd() As Double, Qdmx As Double, Qdmn As Double, Qdmna As Double, headsum2 As Double, count2 As Double, _
         headsum2s As Double, headsum13 As Double, count13 As Double, msgu As String, Lon1 As Integer, Lon2 As Integer, ATC As String)

    Dim i As Integer, sph As Double, ph As Integer

i = 1
For jm = 1 To m
   For im = 1 To L                       'Load parameter values.
    If WorksheetFunction.Proper(Sheet3.Cells(13, 8).Value) = "Uniform" Then  'Load x-axis increment sizes for uniform case.
     dx(im) = Sheet3.Cells(14, 8).Value
    Else
     dx(im) = Sheet3.Cells(25, 3 + im).Value   'Load x-axis increment sizes for discrete case.
    End If
    If WorksheetFunction.Proper(Sheet3.Cells(16, 8).Value) = "Uniform" Then  'Load y-axis increment sizes for uniform case.
     dy(jm) = Sheet3.Cells(17, 8).Value
    Else
     dy(jm) = Sheet3.Cells(26 + jm, 2).Value   'Load y-axis increment sizes for discrete case.
    End If
    If CT(jm, im) = 2 Then
     If dy(jm) > dx(im) Then
      lenp(jm, im) = dy(jm)                    'Load stream lengths (longer of cell dimensions).
      widp(jm, im) = dx(im)
      Lons(jm, im) = 1
      Else
      lenp(jm, im) = dx(im)
```

```
   widp(jm, im) = dy(jm)
   Lons(jm, im) = 2
  End If
 Call Lon12(Lon1, Lon2, ph, jm, im, CT(), L, m)
 If Lon1 <> Lon2 Then
   If Lon1 > Lon2 Then
     Lons(jm, im) = 1
      lenp(jm, im) = dy(jm)
      widp(jm, im) = dx(im)
    Else
     Lons(jm, im) = 2
      lenp(jm, im) = dx(im)
      widp(jm, im) = dy(jm)
   End If
  End If
 End If
If WorksheetFunction.Proper(Sheet4.Cells(8, 5).Value) = "Uniform" Then
  AqT(jm, im) = -1 * Sheet4.Cells(9, 5).Value
 Else
  AqT(jm, im) = -1 * Sheet4.Cells(23 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet6.Cells(8, 24).Value) = "Uniform" Then
  kv(jm, im) = Sheet6.Cells(9, 24).Value
 Else
  kv(jm, im) = Sheet6.Cells(193 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet6.Cells(8, 19).Value) = "Uniform" Then
  kh(jm, im) = Sheet6.Cells(9, 19).Value
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 19).Value) = "Discrete" Then
  kh(jm, im) = Sheet6.Cells(134 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Uniform" Then
  kvq(jm, im) = Sheet6.Cells(9, 11).Value
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Discrete" Then
  kvq(jm, im) = Sheet6.Cells(75 + jm, 2 + im).Value
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Permissive" Then
  kvq(jm, im) = 999999.12321
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Leakance" Then
  kvq(jm, im) = Sheet6.Cells(9, 11).Value
  LK = 1
End If
If WorksheetFunction.Proper(Sheet7.Cells(8, 17).Value) = "Uniform" Then
  Sigma(jm, im) = Sheet7.Cells(9, 17).Value
 Else
  Sigma(jm, im) = Sheet7.Cells(134 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet7.Cells(10, 17).Value) = "Uniform" Then
  Sigs(jm, im) = Sheet7.Cells(11, 17).Value
 Else
  Sigs(jm, im) = Sheet7.Cells(192 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
 AqB(jm, im) = -1 * Sheet4.Cells(12, 5).Value
 Else
 AqB(jm, im) = -1 * Sheet4.Cells(82 + jm, 2 + im).Value
End If
```

```
     If WorksheetFunction.Proper(Sheet5.Cells(8, 5).Value) = "Uniform" Then
       DEC(jm, im) = -1 * Sheet5.Cells(9, 5).Value
       Else
        If CT(jm, im) = 2 Then
         If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
           DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
          Else
           DEC(jm, im) = 0
         End If
         Else
           DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
        End If
      End If
    If WorksheetFunction.Proper(Sheet6.Cells(8, 6).Value) = "Uniform" Then
     k(jm, im) = Sheet6.Cells(9, 6).Value
     Else
     k(jm, im) = Sheet6.Cells(16 + jm, 2 + im).Value
    End If
    If Sheet9.Cells(7, 6).Value = 1 Then
     zone(jm, im) = 1
     Else
     zone(jm, im) = Sheet9.Cells(15 + jm, 2 + im).Value
    End If
    If WorksheetFunction.Proper(Sheet7.Cells(8, 7).Value) = "Uniform" Then
     Sy(jm, im) = Sheet7.Cells(9, 7).Value
     Else
     Sy(jm, im) = Sheet7.Cells(18 + jm, 2 + im).Value
    End If
    If WorksheetFunction.Proper(Sheet7.Cells(10, 7).Value) = "Uniform" Then
     Ss(jm, im) = Sheet7.Cells(11, 7).Value
     Else
     Ss(jm, im) = Sheet7.Cells(76 + jm, 2 + im).Value
    End If
    If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) = "Uniform" Then
     Hn(jm, im) = -1 * Sheet10.Cells(9, 5).Value
     Else
     Hn(jm, im) = -1 * Sheet10.Cells(16 + jm, 2 + im).Value
    End If
    Hno(jm, im) = Hn(jm, im)
    Hnp(jm, im) = Hn(jm, im)
If WorksheetFunction.Proper(Sheet4.Cells(11, 14).Value) <> "None" Then
 If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
    UMA(jm, im) = Sheet4.Cells(12, 14).Value                    'Load Ultimate Matric Suction in Aquifer.
    Else
    UMA(jm, im) = Sheet4.Cells(200 + jm, 2 + im).Value
  End If
 End If
If WorksheetFunction.Proper(Sheet4.Cells(8, 14).Value) <> "None" Then
 If WorksheetFunction.Proper(Sheet4.Cells(8, 5).Value) = "Uniform" Then
    UMAT(jm, im) = Sheet4.Cells(9, 14).Value                    'Load Ultimate Matric Suction in Aquitard.
    Else
    UMAT(jm, im) = Sheet4.Cells(141 + jm, 2 + im).Value
  End If
 End If
 If UMA(jm, im) > UMAT(jm, im) Then
    UMAT(jm, im) = UMA(jm, im)
```

```
  msgu = "Matric Suction Value(s) Altered! "
End If
  If CT(jm, im) = 2 Then                                        'CT=2 if.
    If SB = "Restrictive" Then
     If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
      kpp(jm, im) = Sheet12.Cells(13, 6).Value
      bpp(jm, im) = Sheet12.Cells(14, 6).Value
      Else
      kpp(jm, im) = Sheet12.Cells(27 + jm, 2 + im).Value
      bpp(jm, im) = Sheet12.Cells(87 + jm, 2 + im).Value
     End If
    End If
    If SB <> "Permissive" Then
    If SIT <> "None" Then
    If WorksheetFunction.Proper(Sheet12.Cells(9, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       SHD(jm, im) = -1 * Sheet12.Cells(15, 6).Value
       Else
       SHD(jm, im) = -1 * Sheet12.Cells(147 + jm, 2 + im).Value
      End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
             If SIT = "None" Then
               Bed(jm, im) = SHD(jm, im)
              ElseIf SIT <> "None" Then
              If WorksheetFunction.Proper(Sheet12.Cells(10, 6).Value) = "Specified" Then
               If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
                Bed(jm, im) = -1 * Sheet12.Cells(16, 6).Value
                Else
                Bed(jm, im) = -1 * Sheet12.Cells(207 + jm, 2 + im).Value
               End If
               Else
               Bed(jm, im) = SHD(jm, im)
              End If
            End If
     If Bed(jm, im) > SHD(jm, im) Then
      Bed(jm, im) = SHD(jm, im)
      RSPOFM2 = 2
      If Right(msgh, 32) <> "Bed Level Reset to Stream Head. " Then
        msgh = msgh & "Bed Level Reset to Stream Head. "
      End If
     End If
     If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       widpp(jm, im) = Sheet12.Cells(17, 6).Value
       Else
       widpp(jm, im) = Sheet12.Cells(267 + jm, 2 + im).Value
      End If
      Else
```

```
      widpp(jm, im) = widp(jm, im)
     End If
  If AqB(jm, im) >= Bed(jm, im) - bpp(jm, im) Then
  TipW(1) = 3
  End If
If SB = "Restrictive" Then
If SIT <> "None" Then
If WorksheetFunction.Proper(Sheet12.Cells(7, 14).Value) = "Simple" Then
If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
  UMS(jm, im) = Sheet12.Cells(8, 14).Value                      'Load Ultimate Matric Suction Below Bed.
  Else
  UMS(jm, im) = Sheet12.Cells(327 + jm, 2 + im).Value
End If
End If
 If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
   If UMAT(jm, im) > UMS(jm, im) Then
    UMS(jm, im) = UMAT(jm, im)
    msgu = "Matric Suction Value(s) Altered! "
   End If
   If (Bed(jm, im) - bpp(jm, im)) - (AqT(jm, im) - UMA(jm, im)) <= UMS(jm, im) Then
    UMS(jm, im) = (Bed(jm, im) - bpp(jm, im)) - (AqT(jm, im) - UMA(jm, im))
   End If
   If widpp(jm, im) = widp(jm, im) Then
    UMAT(jm, im) = UMS(jm, im)
   End If
  ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
   If UMA(jm, im) > UMS(jm, im) Then
    UMS(jm, im) = UMA(jm, im)
    msgu = "Matric Suction Value(s) Altered! "
   End If
   If Bed(jm, im) - bpp(jm, im) - AqB(jm, im) <= UMS(jm, im) Then
    UMS(jm, im) = Bed(jm, im) - bpp(jm, im) - AqB(jm, im)
   End If
   If widpp(jm, im) = widp(jm, im) Then
    UMAT(jm, im) = 0
    UMA(jm, im) = UMS(jm, im)
   End If
 End If
End If
End If
          If SIT = "None" Then
            WR(jm, im) = 0
            WRAT(jm, im) = 0
           ElseIf SIT <> "None" Then
           If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
            WRAT(jm, im) = widpp(jm, im) / widp(jm, im)
            WR(jm, im) = 0
            ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
            WR(jm, im) = widpp(jm, im) / widp(jm, im)
            WRAT(jm, im) = WR(jm, im)
           End If
           If WR(jm, im) > 1 Then
             WR(jm, im) = 1
             widpp(jm, im) = widp(jm, im)
             msgwp = "Stream Width Limited to Cell Width. "
           End If
```

```
            If WRAT(jm, im) > 1 Then
              WRAT(jm, im) = 1
              widpp(jm, im) = widp(jm, im)
              msgwp = "Stream Width Limited to Cell Width. "
            End If
           End If
 ElseIf CT(jm, im) <> 2 Then
  WR(jm, im) = 0
  WRAT(jm, im) = 0
End If                              'CT=2.
If Hn(jm, im) > AqT(jm, im) - UMA(jm, im) Then
     Hnat(jm, im) = Hn(jm, im)
    Else
     Hnat(jm, im) = AqT(jm, im) - UMA(jm, im)
End If
Hnoat(jm, im) = Hnat(jm, im)
Hnpat(jm, im) = Hnat(jm, im)
  If CT(jm, im) = 1 Then
   If IMS = "Discrete, Head" Then
     Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
     ElseIf IMS = "Uniform, Head" Then
     Qd(jm, im) = 1
     ElseIf IMS = "Discrete, Volume" Then
     Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
     ElseIf IMS = "Uniform, Volume" Then
     Qd(jm, im) = 1
   End If
   If Qdmx = 0 Then
      Qdmx = Qd(jm, im)
     End If
     If Qdmn = 0 Then
      Qdmn = Qd(jm, im)
     End If
     If Qdmna = 0 Then
      Qdmna = Abs(Qd(jm, im))
   End If
   If Qd(jm, im) > Qdmx Then
      Qdmx = Qd(jm, im)
      End If
      If Qd(jm, im) < Qdmn Then
      Qdmn = Qd(jm, im)
      End If
      If Qd(jm, im) <> 0 Then
      If Abs(Qd(jm, im)) < Qdmna Then
      Qdmna = Abs(Qd(jm, im))
      End If
   End If
  End If
    If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
    sph = Sfunc(2, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), _
                Ss(jm, im), Sy(jm, im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im))
    sph = sph + SfuncAT(CT(jm, im), SIT, WRAT(jm, im), (Hnat(jm, im) + UMAT(jm, im)), _
                AqT(jm, im), Bed(jm, im), Sigs(jm, im), Sigma(jm, im), bpp(jm, im), SB, UMS(jm, im), UMAT(jm, im))
     If CT(jm, im) = 2 Then
     headsum2 = headsum2 + Hn(jm, im) * dx(im) * dy(jm) * sph
     count2 = count2 + dx(im) * dy(jm) * sph
```

```
     If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
       headsum2s = headsum2s + (Hnat(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
       Else
       headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
     End If
     End If
     If CT(jm, im) Mod 2 = 1 Then
     headsum13 = headsum13 + Hn(jm, im) * dx(im) * dy(jm) * sph
     count13 = count13 + dx(im) * dy(jm) * sph
     End If
   Else
     If CT(jm, im) = 2 Then
       If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
       headsum2s = headsum2s + (Hnat(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
       Else
       headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
       End If
     End If
   End If
    i = i + 1
   Next im
  Next jm
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'Preliminary processing of parameters.
Sub Prelim(jm As Integer, m As Integer, im As Integer, L As Integer, _
           CT() As Integer, Bed() As Double, bpp() As Double, _
           AqT() As Double, DEC() As Double, TipW() As Integer, _
           Hn() As Double, Hnat() As Double, AqB() As Double, AqBi() As Double, AqTi() As Double, BEDi() As Double, _
           CTi() As Integer, bppi() As Double, UMAi() As Double, UMATi() As Double, UMSi() As Double, _
           ho() As Double, hoo() As Double, UMA() As Double, UMAT() As Double, UMS() As Double, _
           ATC As String, r As Integer, PermFact() As Double, widp() As Double, widpp() As Double, SB As String, _
           SIT As String, msgsep As String, flaw As Integer, WR() As Double)

Dim i As Integer

msgsep = "2"

i = 1
For jm = 1 To m
   For im = 1 To L
   PermFact(jm, im, 1) = 1
   PermFact(jm, im, 2) = 1
   If CT(jm, im) <> 4 Then
    If DEC(jm, im) > 0 Then
    TipW(1) = 1
    End If
    If Hnat(jm, im) > DEC(jm, im) Then
    TipW(2) = 1
    End If
    If AqT(jm, im) >= Hn(jm, im) Then
    TipW(3) = 1
```

```
    End If
    If AqB(jm, im) >= AqT(jm, im) Then
    TipW(4) = 1
    End If
   End If
    AqBi(i) = AqB(jm, im)               'Fill single index variables.
    AqTi(i) = AqT(jm, im)
    BEDi(i) = Bed(jm, im)
    CTi(i) = CT(jm, im)
    bppi(i) = bpp(jm, im)
    UMAi(i + r / 2) = UMA(jm, im)
    UMATi(i) = UMAT(jm, im)
    UMSi(i) = UMS(jm, im)
    ho(jm, im) = hofunc(Hn(jm, im), AqT(jm, im), AqB(jm, im), UMA(jm, im), ATC, UMS(jm, im), Bed(jm, im), bpp(jm, im), WR(jm, im))
    hoo(jm, im) = ho(jm, im)
    If CT(jm, im) = 2 Then
    If SB = "Restrictive" Then
     If SIT = "Simple" Then
       If Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
         If widpp(jm, im) < widp(jm, im) Then
          msgsep = "1"
          flaw = 4
         End If
       End If 'Bed
     End If  'SIT
    End If 'SB
    End If 'CT
    i = i + 1
   Next im
  Next jm
End Sub
'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************
'IMPULSE DISTRIBUTION BLOCK for Delayed-Yield.
Sub IDB(o As Long, ic As String, Qc() As Double, dH As Double, Qtt As Double, Qm() As Double, Qf As Double, IA As Double, _
        IAP As Double, jm As Integer, im As Integer, m As Integer, L As Integer, i As Integer, CT() As Integer, IMS As String, _
        Qd() As Double, ITS As String, dx() As Double, dy() As Double, Sigma() As Double, ATC As String, SIT As String, WR() As Double, _
        Hn() As Double, AqT() As Double, Bed() As Double, Ss() As Double, Sy() As Double, AqB() As Double, bpp() As Double, SB As String, _
        UMA() As Double, UMS() As Double, GTC As Integer, msgim As String, msgti As String, UCF As Double, qs As Double, Qdmna As Double, _
        msgim2 As String, Hnat() As Double, msgc As String, NegHead As Integer, NegHeadNM As Integer, IGTC As Integer, ObF As Double, _
        Qt As Double, ObFF As Double, dFdu As Double, msgtt As String, subsflag As Integer, DEC() As Double, nm As Long, nc As Long, _
        dt As Double, flaw As Integer, Qb As Double)

Dim kitr As Integer

If o > 1 Then
  If ic = "Instantaneous" Then
    ReDim Qc(1 To m, 1 To L)
    dH = 0
    Qtt = 0
    kitr = 0
    Qb = 0
  End If
```

```
    subsflag = 8
    Exit Sub
End If
If ic = "Steady" Then
   Qb = -1 * UCF * Qm(nm) / nc
   Qm(nm) = Qm(nm) / nc
  Else
   Qb = -1 * UCF * Qm(nm)
End If
'||||
dH = 0
Qf = 0
  IA = 0
  IAP = 0
    For jm = 1 To m                                 'TALLEY RELATIVE IMPULSE MAGNITUDE WEIGHTING.
      For im = 1 To L
        If CT(jm, im) = 1 Then                            'CT if - impulse cells.
         If Right(IMS, 4) = "lume" Then                       'IMS if - Volumetric impulse distribution case.
          IA = IA + Qd(jm, im)
          If Qd(jm, im) > 0 Then                          'Alternate denominator for zero net impulse case, by volume.
            IAP = IAP + Qd(jm, im)
          End If                                          'Denominator end if.
          ElseIf Right(IMS, 4) = "Head" Then                  'IMS ElseIf.
           If ITS = "Deep Percolation" Then
             IA = IA + (Qd(jm, im) * dx(im) * dy(jm) * Sigma(jm, im))
            Else
             IA = IA + (Qd(jm, im) * dx(im) * dy(jm) * Sfunc(1, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), Ss(jm,
im), Sy(jm, im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im)))
            End If
          If Qd(jm, im) > 0 Then                          'Alternate denominator for zero net impulse case.
            If ITS = "Deep Percolation" Then
              IAP = IAP + (Qd(jm, im) * dx(im) * dy(jm) * Sigma(jm, im))
             Else
              IAP = IAP + (Qd(jm, im) * dx(im) * dy(jm) * Sfunc(1, SIT, ATC, WR(jm, im), (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), Bed(jm, im), Ss(jm,
im), Sy(jm, im), AqB(jm, im), bpp(jm, im), SB, UMA(jm, im), UMS(jm, im)))
            End If
          End If                                          'Denominator end if.
        End If                                                'IMS end if.
        End If                                            'CT end if
      Next im
   Next jm
If IMS = "Uniform, Head" Then                   'IMS if.++++++++++++++++++++++++++++++++++
    Qf = UCF * qs * Qm(nm)
    dH = -1 * Qf * Qdmna / IA
     GTC = 1
     GoTo Line4
Line1:   '=========================
ElseIf IMS = "Uniform, Volume" Then             'IMS elseif.++++++++++++++++++++++++++++++
  For jm = 1 To m                                              'Initialize loops, separate to allow prior loops to total denominator, AI.
   For im = 1 To L                                            'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   If CT(jm, im) = 1 Then                                'CT if - only impulse cells.
    Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IA 'Fill impulse volume.
   End If                                               'CT end if
   Next im
  Next jm
ElseIf IMS = "Discrete, Head" Then              'IMS elseif.++++++++++++++++++++++++++++++
```

```
  If msgim = "Mixed Impulse. " Then
     msgti = "Discrete Impulse Distribution by Head Requires Like Signs Throughout Grid. Distribute by Volume Instead. "
     subsflag = 50
     Exit Sub
  End If
    If IA = 0 Then
     msgti = "Discrete Impulse Distribution by Head Requires Non-zero Weighting. "
     subsflag = 50
     Exit Sub
    End If
    If IA > 0 Then                                          'IA If.
         Qf = UCF * qs * Qm(nm)
         dH = -1 * Qf * Qdmna / IA
       ElseIf IA < 0 Then                                   'IA ElseIf.
         Qf = UCF * qs * Qm(nm) * (-1)
         dH = -1 * Qf * Qdmna / IA
    End If                                                  'IA end if.
     GTC = 2
     GoTo Line4
Line2:   '=========================
ElseIf IMS = "Discrete, Volume" Then           'IMS elseif.++++++++++++++++++++++++++++++
For jm = 1 To m                                              'Initialize loops, separate to allow prior loops to total denominator, AI.
   For im = 1 To L                                             'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   If CT(jm, im) = 1 Then                                 'CT if - only impulse cells.
    If IA <> 0 Then                                       'IA If.
    Qc(jm, im) = UCF * qs * Qm(nm) * Abs(Qd(jm, im)) / IA 'Fill impulse volume.
    Else
      If IAP > 0 Then                                      'IAP if.
      msgim2 = "Impulses Net 0! "                'Zero NET impulse case.
      Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IAP
      Else
      msgim2 = "Impulse Tab 0s! "                         'Zero impulse case.
      subsflag = 50
      Exit Sub
      End If                                               'End IAP end if.
    End If                                                 'IA end if.
   End If                                                 'CT end if
   Next im
Next jm
End If                                         'IMS end if.
'||||
  If ic = "Instantaneous" Then                 'ic if.++++++++++++++++++++++++++++++++++
  Qtt = 0
  i = 1
  For jm = 1 To m                                              'Initialize loops, separate to allow prior loops to total denominator, AI.
   For im = 1 To L                                            'CONVERT INSTANTANEOUS IMPULSE TO HEAD.
   If CT(jm, im) = 1 Then                               'CT if - only impulse cells.
         Qtt = Qtt + Qc(jm, im)                         'For Domain Volumetric Budget Later.
If ITS = "Deep Percolation" Then 'ITS if.
           If Hnat(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Sigma(jm, im)) < AqT(jm, im) - UMA(jm, im) Then 'Dewatering.
            GoTo Line3
           Else                                                                    'Dewatering else.
            Hnat(jm, im) = Hnat(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Sigma(jm, im))      'Not dewatered.
         End If                                                                    'Dewatering end if.
         If Hnat(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
            msgc = "Inundated Ground! "
```

```
Line3:   '==========================
             msgti = "Impulse Exceeds Aquitard Capacity. "
             flaw = 3
             NegHead = 6
             NegHeadNM = nm + 38
             subsflag = 50
             Exit Sub
          End If
Else                              'ITS else.
           If Hn(jm, im) >= AqT(jm, im) - UMA(jm, im) Then
            If Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im))) < AqT(jm, im) - UMA(jm, im) Then
'Depressurization.
             Hn(jm, im) = (AqT(jm, im) - UMA(jm, im)) - (Qc(jm, im) - (Hn(jm, im) - (AqT(jm, im) - UMA(jm, im))) * dx(im) * dy(jm) * Ss(jm, im) *
(AqT(jm, im) - AqB(jm, im))) / (dx(im) * dy(jm) * Sy(jm, im))
    If msgti = "" Then
    msgti = "Impulse Distribution Through Confinement Transition. "
    End If
            Else
             Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im))) 'Pressurized.
          End If
          ElseIf Hn(jm, im) < AqT(jm, im) - UMA(jm, im) Then
           If Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Sy(jm, im)) > AqT(jm, im) - UMA(jm, im) Then 'Pressurization.
             Hn(jm, im) = (AqT(jm, im) - UMA(jm, im)) - (Qc(jm, im) - (Hn(jm, im) - (AqT(jm, im) - UMA(jm, im))) * dx(im) * dy(jm) * Sy(jm, im)) /
(dx(im) * dy(jm) * Ss(jm, im) * (AqT(jm, im) - AqB(jm, im)))
    If msgti = "" Then
    msgti = "Impulse Distribution Through Confinement Transition. "
    End If
            Else
             Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Sy(jm, im)) 'Depressurized.
           End If
          End If
End If                          'ITS end if.
       Qc(jm, im) = 0                              'Annul converted flow (instantaneous).
     End If                                        'CT end if.
     i = i + 1
     Next im                                      'Increment to loop.
    Next jm
    ElseIf ic = "Steady" Then                   'ic elseif.++++++++++++++++++++++++++++++
    Qtt = 0
    For jm = 1 To m                                  'CONVERT STEADY IMPULSE TO FLOW.
    For im = 1 To L
     If CT(jm, im) = 1 Then
       Qtt = Qtt + Qc(jm, im)                       'For Domain Volumetric Budget Later.
       Qc(jm, im) = Qc(jm, im) / (dt)               'Convert to volume per time.
     End If
   Next im
   Next jm
  End If                                       'ic end if.
'||||
If ic = "Steady" Then                          'Restore Qm for later use.
    Qm(nm) = Qm(nm) * nc
End If
subsflag = 8
Exit Sub                                    'Direct past solver.++++++++++++++++++++++
'||||>
Line4:          '==========================
```

```
'START NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION OVER GRID BY HEAD]]]]]]]]]]
   kitr = 0                                        'Initialize iteration counter.
   IGTC = 0                                        'Initialize impulse go to code.
   ObF = 1                                         'Initialize objective function.
   Do While Abs(ObF) > 0.0000001        'Solver loop.
   If kitr < 500 Then                             'Iteration limit check
     kitr = kitr + 1                              'Increment iteration counter.
Line5:          '==========================
   Qt = 0                                          'Initialize virtual volume tally.
   '((((>
   i = 1
   For jm = 1 To m                                       'Prepare for calculation loop.
   For im = 1 To L                          'Calculation loop.
   If CT(jm, im) = 1 Then                         'CT if - cell type check.
   Qc(jm, im) = QCF(ATC, ic, ITS, Qd(jm, im), dH, Qdmna, Sigma(jm, im), dx(im), dy(jm), Hn(jm, im), AqT(jm, im), Ss(jm, im), AqB(jm, im), Sy(jm, im),
UMA(jm, im))
     If IA <> 0 Then
     Qt = Qt + Qc(jm, im)                          'For distribution by Newton Solver.
     Else
     If Qd(jm, im) > 0 Then
     Qt = Qt + Qc(jm, im)
     End If
     End If
  End If                                        'CT end if.
   i = i + 1
  Next im
  Next jm                                          'Close calculation loop.
   '((((>
   If IGTC = 6 Then
   IGTC = 0
   GoTo Line6
   End If
   '((((>
   ObF = Qf - Qt                                  'Compute objective function [F(n)].
  If Abs(ObF) <= 0.0000001 Then
  GoTo Line7
  End If
  dH = (1 + 0.00000001) * dH                      'Increment head increment.
   IGTC = 6
   GoTo Line5
Line6:   '==========================
  ObFF = Qf - Qt                                  'Recompute objective function [F(n+1)].
  dFdu = (ObFF - ObF) / (dH - dH / (1 + 0.00000001)) 'Calculate derivative.
  dH = dH / (1 + 0.00000001)                     'Restore head increment.
  dH = dH - ObF / dFdu                           'Compute next estimate for head increment.
Line7:   '==========================
  Else                                                    'Intervene to end iteration if not converging in a reasonable number of cycles.
   NegHead = 4
   msgtt = "Impulse Distribution Iteration Limit Exceeded! "
   NegHeadNM = nm + 38
    subsflag = 50
    Exit Sub
  End If                                                  'Iteration counter end if.
  Loop                                           'Close solver loop.
'END NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION]]]]]]]]]]
If GTC = 1 Then
```

```
 GoTo Line1
ElseIf GTC = 2 Then
 GoTo Line2
End If                                  'END IMPULSE DISTRIBUTION BLOCK.++++++++++
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'Subroutine to load initial parameter values.
Sub LoadC(RSPOFM2 As Integer, SB As String, SIT As String, IMS As String, im As Integer, L As Integer, jm As Integer, m As Integer, msgwp As String,
_
         dx() As Double, dy() As Double, CT() As Integer, widp() As Double, widpp() As Double, lenp() As Double, Lons() As Integer, AqT() As Double,
_
         kv() As Double, kh() As Double, kvq() As Double, LK As Integer, Sigma() As Double, Sigs() As Double, AqB() As Double, DEC() As Double, _
         k() As Double, zone() As String, Sy() As Double, Ss() As Double, Hn() As Double, Hno() As Double, Hnp() As Double, Hnat() As Double, _
         Hnoat() As Double, Hnpat() As Double, UMA() As Double, UMAT() As Double, kpp() As Double, bpp() As Double, kppb() As Double, bppb() As
Double, _
         SHD() As Double, Bed() As Double, msgpb As String, msgh As String, TipW() As Integer, UMS() As Double, _
         Qd() As Double, Qdmx As Double, Qdmn As Double, Qdmna As Double, msgu As String, Lon1 As Integer, Lon2 As Integer, ATC As String)

    Dim i As Integer, ph As Integer

i = 1
For jm = 1 To m
   For im = 1 To L                          'Load parameter values.
    If WorksheetFunction.Proper(Sheet3.Cells(13, 8).Value) = "Uniform" Then  'Load x-axis increment sizes for uniform case.
     dx(im) = Sheet3.Cells(14, 8).Value
    Else
     dx(im) = Sheet3.Cells(25, 3 + im).Value   'Load x-axis increment sizes for discrete case.
    End If
    If WorksheetFunction.Proper(Sheet3.Cells(16, 8).Value) = "Uniform" Then  'Load y-axis increment sizes for uniform case.
     dy(jm) = Sheet3.Cells(17, 8).Value
    Else
     dy(jm) = Sheet3.Cells(26 + jm, 2).Value   'Load y-axis increment sizes for discrete case.
    End If
    If CT(jm, im) = 2 Then
     If dy(jm) > dx(im) Then
      lenp(jm, im) = dy(jm)                     'Load stream lengths (longer of cell dimensions).
      widp(jm, im) = dx(im)
      Lons(jm, im) = 1
      Else
      lenp(jm, im) = dx(im)
      widp(jm, im) = dy(jm)
      Lons(jm, im) = 2
     End If
    Call Lon12(Lon1, Lon2, ph, jm, im, CT(), L, m)
    If Lon1 <> Lon2 Then
      If Lon1 > Lon2 Then
        Lons(jm, im) = 1
         lenp(jm, im) = dy(jm)
         widp(jm, im) = dx(im)
       Else
        Lons(jm, im) = 2
         lenp(jm, im) = dx(im)
```

```
      widp(jm, im) = dy(jm)
   End If
  End If
 End If
If WorksheetFunction.Proper(Sheet4.Cells(8, 5).Value) = "Uniform" Then
  AqT(jm, im) = -1 * Sheet4.Cells(9, 5).Value
 Else
  AqT(jm, im) = -1 * Sheet4.Cells(23 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet6.Cells(8, 24).Value) = "Uniform" Then
  kv(jm, im) = Sheet6.Cells(9, 24).Value
 Else
  kv(jm, im) = Sheet6.Cells(193 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet6.Cells(8, 19).Value) = "Uniform" Then
  kh(jm, im) = Sheet6.Cells(9, 19).Value
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 19).Value) = "Discrete" Then
  kh(jm, im) = Sheet6.Cells(134 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Uniform" Then
  kvq(jm, im) = Sheet6.Cells(9, 11).Value
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Discrete" Then
  kvq(jm, im) = Sheet6.Cells(75 + jm, 2 + im).Value
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Permissive" Then
  kvq(jm, im) = 999999.12321
 ElseIf WorksheetFunction.Proper(Sheet6.Cells(8, 11).Value) = "Leakance" Then
  kvq(jm, im) = Sheet6.Cells(9, 11).Value
  LK = 1
End If
If WorksheetFunction.Proper(Sheet7.Cells(8, 17).Value) = "Uniform" Then
  Sigma(jm, im) = Sheet7.Cells(9, 17).Value
 Else
  Sigma(jm, im) = Sheet7.Cells(134 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet7.Cells(10, 17).Value) = "Uniform" Then
  Sigs(jm, im) = Sheet7.Cells(11, 17).Value
 Else
  Sigs(jm, im) = Sheet7.Cells(192 + jm, 2 + im).Value
End If
If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
 AqB(jm, im) = -1 * Sheet4.Cells(12, 5).Value
 Else
 AqB(jm, im) = -1 * Sheet4.Cells(82 + jm, 2 + im).Value
End If
  If WorksheetFunction.Proper(Sheet5.Cells(8, 5).Value) = "Uniform" Then
    DEC(jm, im) = -1 * Sheet5.Cells(9, 5).Value
    Else
     If CT(jm, im) = 2 Then
      If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
        DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
       Else
        DEC(jm, im) = 0
      End If
      Else
        DEC(jm, im) = -1 * Sheet5.Cells(16 + jm, 2 + im).Value
     End If
```

```
     End If
    If WorksheetFunction.Proper(Sheet6.Cells(8, 6).Value) = "Uniform" Then
     k(jm, im) = Sheet6.Cells(9, 6).Value
     Else
     k(jm, im) = Sheet6.Cells(16 + jm, 2 + im).Value
    End If
    If Sheet9.Cells(7, 6).Value = 1 Then
     zone(jm, im) = 1
     Else
     zone(jm, im) = Sheet9.Cells(15 + jm, 2 + im).Value
    End If
    If WorksheetFunction.Proper(Sheet7.Cells(8, 7).Value) = "Uniform" Then
     Sy(jm, im) = Sheet7.Cells(9, 7).Value
     Else
     Sy(jm, im) = Sheet7.Cells(18 + jm, 2 + im).Value
    End If
    If WorksheetFunction.Proper(Sheet7.Cells(10, 7).Value) = "Uniform" Then
     Ss(jm, im) = Sheet7.Cells(11, 7).Value
     Else
     Ss(jm, im) = Sheet7.Cells(76 + jm, 2 + im).Value
    End If
    If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) = "Uniform" Then
     Hn(jm, im) = -1 * Sheet10.Cells(9, 5).Value
     Else
     Hn(jm, im) = -1 * Sheet10.Cells(16 + jm, 2 + im).Value
    End If
    Hno(jm, im) = Hn(jm, im)
    Hnp(jm, im) = Hn(jm, im)
If WorksheetFunction.Proper(Sheet4.Cells(11, 14).Value) <> "None" Then
 If WorksheetFunction.Proper(Sheet4.Cells(11, 5).Value) = "Uniform" Then
    UMA(jm, im) = Sheet4.Cells(12, 14).Value                    'Load Ultimate Matric Suction in Aquifer.
    Else
    UMA(jm, im) = Sheet4.Cells(200 + jm, 2 + im).Value
  End If
 End If
If WorksheetFunction.Proper(Sheet4.Cells(8, 14).Value) <> "None" Then
 If WorksheetFunction.Proper(Sheet4.Cells(8, 5).Value) = "Uniform" Then
    UMAT(jm, im) = Sheet4.Cells(9, 14).Value                    'Load Ultimate Matric Suction in Aquitard.
    Else
    UMAT(jm, im) = Sheet4.Cells(141 + jm, 2 + im).Value
  End If
 End If
 If UMA(jm, im) > UMAT(jm, im) Then
    UMAT(jm, im) = UMA(jm, im)
    msgu = "Matric Suction Value(s) Altered! "
 End If
    If CT(jm, im) = 2 Then                                      'CT=2 if.
       If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
        kpp(jm, im) = Sheet12.Cells(13, 6).Value
        bpp(jm, im) = Sheet12.Cells(14, 6).Value
        Else
        kpp(jm, im) = Sheet12.Cells(27 + jm, 2 + im).Value
        bpp(jm, im) = Sheet12.Cells(87 + jm, 2 + im).Value
       End If
       If WorksheetFunction.Proper(Sheet12.Cells(13, 14).Value) = "Equivalent" Then
        kppb(jm, im) = kpp(jm, im)
```

```
      bppb(jm, im) = bpp(jm, im)
     Else
     If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
      kppb(jm, im) = Sheet12.Cells(14, 14).Value
      bppb(jm, im) = Sheet12.Cells(15, 14).Value
      Else
      kppb(jm, im) = Sheet12.Cells(387 + jm, 2 + im).Value
      bppb(jm, im) = Sheet12.Cells(447 + jm, 2 + im).Value
     End If
     End If  'Equivalent.
    If SB <> "Permissive" Then
    If SIT <> "None" Then
    If WorksheetFunction.Proper(Sheet12.Cells(9, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       SHD(jm, im) = -1 * Sheet12.Cells(15, 6).Value
       Else
       SHD(jm, im) = -1 * Sheet12.Cells(147 + jm, 2 + im).Value
      End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
      Else
      SHD(jm, im) = Hn(jm, im)
     End If
              If WorksheetFunction.Proper(Sheet12.Cells(10, 6).Value) = "Specified" Then
               If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
                Bed(jm, im) = -1 * Sheet12.Cells(16, 6).Value
                Else
                Bed(jm, im) = -1 * Sheet12.Cells(207 + jm, 2 + im).Value
               End If
               Else
               Bed(jm, im) = SHD(jm, im)
              End If
     If Bed(jm, im) > SHD(jm, im) Then
      Bed(jm, im) = SHD(jm, im)
      RSPOFM2 = 2
      If Right(msgh, 32) <> "Bed Level Reset to Stream Head. " Then
        msgh = msgh & "Bed Level Reset to Stream Head. "
      End If
     End If
     If WorksheetFunction.Proper(Sheet12.Cells(11, 6).Value) = "Specified" Then
      If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
       widpp(jm, im) = Sheet12.Cells(17, 6).Value
       Else
       widpp(jm, im) = Sheet12.Cells(267 + jm, 2 + im).Value
      End If
      Else
      widpp(jm, im) = widp(jm, im)
     End If
If SIT <> "None" Then
If WorksheetFunction.Proper(Sheet12.Cells(7, 14).Value) = "Simple" Then
If WorksheetFunction.Proper(Sheet12.Cells(12, 6).Value) = "Uniform" Then
  UMS(jm, im) = Sheet12.Cells(8, 14).Value                      'Load Ultimate Matric Suction Below Bed.
```

```
   Else
   UMS(jm, im) = Sheet12.Cells(327 + jm, 2 + im).Value
 End If
 End If
  If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
    If UMAT(jm, im) > UMS(jm, im) Then
     UMS(jm, im) = UMAT(jm, im)
     msgu = "Matric Suction Value(s) Altered! "
    End If
    If (Bed(jm, im) - bpp(jm, im)) - (AqT(jm, im) - UMA(jm, im)) <= UMS(jm, im) Then
     UMS(jm, im) = (Bed(jm, im) - bpp(jm, im)) - (AqT(jm, im) - UMA(jm, im))
    End If
    If widpp(jm, im) = widp(jm, im) Then
     UMAT(jm, im) = UMS(jm, im)
    End If
   ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
    If UMA(jm, im) > UMS(jm, im) Then
     UMS(jm, im) = UMA(jm, im)
     msgu = "Matric Suction Value(s) Altered! "
    End If
    If Bed(jm, im) - bpp(jm, im) - AqB(jm, im) <= UMS(jm, im) Then
     UMS(jm, im) = Bed(jm, im) - bpp(jm, im) - AqB(jm, im)
    End If
    If widpp(jm, im) = widp(jm, im) Then
     UMAT(jm, im) = 0
     UMA(jm, im) = UMS(jm, im)
    End If
  End If
 End If
            If widpp(jm, im) > widp(jm, im) Then
              widpp(jm, im) = widp(jm, im)
              msgwp = "Stream Width Limited to Cell Width. "
            End If
 ElseIf CT(jm, im) <> 2 Then
End If                              'CT=2.
If Hn(jm, im) > AqT(jm, im) - UMA(jm, im) Then
     Hnat(jm, im) = Hn(jm, im)
    Else
     Hnat(jm, im) = AqT(jm, im) - UMA(jm, im)
End If
Hnoat(jm, im) = Hnat(jm, im)
Hnpat(jm, im) = Hnat(jm, im)
  If CT(jm, im) = 1 Then
   If IMS = "Discrete, Head" Then
     Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
     ElseIf IMS = "Uniform, Head" Then
     Qd(jm, im) = 1
     ElseIf IMS = "Discrete, Volume" Then
     Qd(jm, im) = Sheet8.Cells(20 + jm, 2 + im).Value
     ElseIf IMS = "Uniform, Volume" Then
     Qd(jm, im) = 1
   End If
   If Qdmx = 0 Then
      Qdmx = Qd(jm, im)
     End If
     If Qdmn = 0 Then
```

```
      Qdmn = Qd(jm, im)
     End If
     If Qdmna = 0 Then
      Qdmna = Abs(Qd(jm, im))
   End If
   If Qd(jm, im) > Qdmx Then
      Qdmx = Qd(jm, im)
      End If
      If Qd(jm, im) < Qdmn Then
      Qdmn = Qd(jm, im)
      End If
      If Qd(jm, im) <> 0 Then
      If Abs(Qd(jm, im)) < Qdmna Then
      Qdmna = Abs(Qd(jm, im))
      End If
   End If
  End If
    i = i + 1
   Next im
  Next jm
End Sub
'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************
'Preliminary processing of parameters.
Sub PrelimC(jm As Integer, m As Integer, im As Integer, L As Integer, _
            CT() As Integer, Bed() As Double, bpp() As Double, _
            AqT() As Double, DEC() As Double, TipW() As Integer, _
            Hn() As Double, Hnat() As Double, AqB() As Double, AqBi() As Double, AqTi() As Double, BEDi() As Double, _
            CTi() As Integer, bppi() As Double, UMAi() As Double, UMATi() As Double, UMSi() As Double, _
            UMA() As Double, UMAT() As Double, UMS() As Double, _
            ATC As String, r As Integer, PermFact() As Double, ovp() As Double, ovw() As Double, beds() As Double, _
            SIT As String, dx() As Double, dy() As Double, headsum2 As Double, count2 As Double, headsum2s As Double, _
            headsum13 As Double, count13 As Double, widp() As Double, widpp() As Double, bppb() As Double, lenp() As Double, _
            Ss() As Double, Sy() As Double, SB As String, SHD() As Double, hx() As Double, Sigs() As Double, _
            Sigma() As Double, msgsep As String, flaw As Integer)

Dim i As Integer, Sp As Double
ReDim hx(1 To r)

msgsep = "2"

i = 1
For jm = 1 To m
   For im = 1 To L
   If WorksheetFunction.Proper(Sheet10.Cells(8, 5).Value) <> "Uniform" Then
    Sp = SfuncC(CT(jm, im), SIT, ATC, (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), widp(jm, im), _
              lenp(jm, im), widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
              SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
              ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
              ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
    Sp = Sp + SfuncCt(CT(jm, im), SIT, "Confined", (Hnat(jm, im) + UMAT(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), _
                    widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im), _
                    SB, UMAT(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
```

```
                ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
  If CT(jm, im) = 2 Then
  headsum2 = headsum2 + Hn(jm, im) * dx(im) * dy(jm) * Sp
  count2 = count2 + dx(im) * dy(jm) * Sp
  If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
    headsum2s = headsum2s + (Hnat(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
    Else
    headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
  End If
  End If
  If CT(jm, im) Mod 2 = 1 Then
  headsum13 = headsum13 + Hn(jm, im) * dx(im) * dy(jm) * Sp
  count13 = count13 + dx(im) * dy(jm) * Sp
  End If
Else
  If CT(jm, im) = 2 Then
    If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
    headsum2s = headsum2s + (Hnat(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
    Else
    headsum2s = headsum2s + (Hn(jm, im) - SHD(jm, im)) * dx(im) * dy(jm)
    End If
  End If
End If
PermFact(jm, im, 1) = 1
PermFact(jm, im, 2) = 1
If CT(jm, im) <> 4 Then
 If DEC(jm, im) > 0 Then
 TipW(1) = 1
 End If
 If Hnat(jm, im) > DEC(jm, im) Then
 TipW(2) = 1
 End If
 If AqT(jm, im) >= Hn(jm, im) Then
 TipW(3) = 1
 End If
 If AqB(jm, im) >= AqT(jm, im) Then
 TipW(4) = 1
 End If
End If
 AqBi(i) = AqB(jm, im)               'Fill single index variables.
 AqTi(i) = AqT(jm, im)
 BEDi(i) = Bed(jm, im)
 CTi(i) = CT(jm, im)
 bppi(i) = bpp(jm, im)
 UMAi(i + r / 2) = UMA(jm, im)
 UMATi(i) = UMAT(jm, im)
 UMSi(i) = UMS(jm, im)
 hx(i) = Hnat(jm, im)
 hx(i + r / 2) = Hn(jm, im)
 If CT(jm, im) = 2 Then
    If Bed(jm, im) <= AqT(jm, im) Then
      If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
       msgsep = "1"
       flaw = 4
      End If
```

```
        ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
         If widpp(jm, im) < widp(jm, im) Then
          msgsep = "1"
          flaw = 4
         End If
        End If 'Bed
    End If 'CT
    i = i + 1
   Next im
  Next jm
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'IMPULSE DISTRIBUTION BLOCK for Delayed-Yield.
Sub IDBC(o As Long, ic As String, Qc() As Double, Qtt As Double, Qm() As Double, Qf As Double, IA As Double, _
        IAP As Double, m As Integer, L As Integer, CT() As Integer, IMS As String, _
        Qd() As Double, ITS As String, dx() As Double, dy() As Double, Sigma() As Double, Sigs() As Double, SIT As String, _
        Hn() As Double, AqT() As Double, Bed() As Double, Ss() As Double, Sy() As Double, AqB() As Double, bpp() As Double, SB As String, _
        UMA() As Double, UMS() As Double, msgim As String, msgti As String, UCF As Double, qs As Double, Qdmna As Double, _
        msgim2 As String, Hnat() As Double, msgc As String, NegHead As Integer, NegHeadNM As Integer, _
        Qt As Double, msgtt As String, subsflag As Integer, DEC() As Double, nm As Long, nc As Long, _
        dt As Double, flaw As Integer, THRS() As Double, THRCNT() As Integer, THRSt() As Double, THRCNTt() As Integer, _
        widp() As Double, ovp() As Double, ovw() As Double, beds() As Double, lenp() As Double, widpp() As Double, _
        bppb() As Double, UMAT() As Double, Qb As Double)

Dim kitr As Integer
Dim jm As Integer, im As Integer, i As Integer, z As Integer
Dim GTC As Integer, IGTC As Integer
Dim Si As Double, sti As Double
Dim dH As Double, ObF As Double, ObFF As Double, dFdu As Double

If o > 1 Then
  If ic = "Instantaneous" Then
    ReDim Qc(1 To m, 1 To L)
    dH = 0
    Qtt = 0
    kitr = 0
    Qb = 0
  End If
    subsflag = 8
    Exit Sub
End If
If ic = "Steady" Then
   Qb = -1 * UCF * Qm(nm) / nc
   Qm(nm) = Qm(nm) / nc
  Else
   Qb = -1 * UCF * Qm(nm)
End If
'||||
dH = 0
Qf = 0
  IA = 0
  IAP = 0
```

```
    For jm = 1 To m                                 'TALLEY RELATIVE IMPULSE MAGNITUDE WEIGHTING.
      For im = 1 To L
        If CT(jm, im) = 1 Then                                  'CT if - impulse cells.
         If Right(IMS, 4) = "lume" Then                             'IMS if - Volumetric impulse distribution case.
          IA = IA + Qd(jm, im)
          If Qd(jm, im) > 0 Then                                'Alternate denominator for zero net impulse case, by volume.
            IAP = IAP + Qd(jm, im)
          End If                                                'Denominator end if.
          ElseIf Right(IMS, 4) = "Head" Then                        'IMS ElseIf.
           If ITS = "Deep Percolation" Then
           sti = SfuncCt(CT(jm, im), SIT, "Confined", (Hnat(jm, im) + UMAT(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), _
                    widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im), _
                    SB, UMAT(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                    ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                    ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
            IA = IA + (Qd(jm, im) * dx(im) * dy(jm) * sti)
           Else
           Si = SfuncC(CT(jm, im), SIT, "Confined", (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), _
                    widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
                    SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                    ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                    ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
            IA = IA + (Qd(jm, im) * dx(im) * dy(jm) * Si)
           End If
          If Qd(jm, im) > 0 Then                                'Alternate denominator for zero net impulse case.
            If ITS = "Deep Percolation" Then
              IAP = IAP + (Qd(jm, im) * dx(im) * dy(jm) * sti)
             Else
              IAP = IAP + (Qd(jm, im) * dx(im) * dy(jm) * Si)
            End If
          End If                                                'Denominator end if.
        End If                                                      'IMS end if.
        End If                                                  'CT end if
      Next im
   Next jm
If IMS = "Uniform, Head" Then                   'IMS if.++++++++++++++++++++++++++++++++++
    Qf = UCF * qs * Qm(nm)
    dH = -1 * Qf * Qdmna / IA
     GTC = 1
     GoTo Line4
Line1:   '=========================
ElseIf IMS = "Uniform, Volume" Then             'IMS elseif.+++++++++++++++++++++++++++++++
  For jm = 1 To m                                               'Initialize loops, separate to allow prior loops to total denominator, AI.
   For im = 1 To L                                              'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   If CT(jm, im) = 1 Then                               'CT if - only impulse cells.
    Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IA 'Fill impulse volume.
   End If                                               'CT end if
   Next im
  Next jm
ElseIf IMS = "Discrete, Head" Then              'IMS elseif.+++++++++++++++++++++++++++++++
  If msgim = "Mixed Impulse. " Then
     msgti = "Discrete Impulse Distribution by Head Requires Like Signs Throughout Grid. Distribute by Volume Instead. "
     subsflag = 50
     Exit Sub
  End If
    If IA = 0 Then
```

```
     msgti = "Discrete Impulse Distribution by Head Requires Non-zero Weighting. "
     subsflag = 50
     Exit Sub
    End If
    If IA > 0 Then                                          'IA If.
         Qf = UCF * qs * Qm(nm)
         dH = -1 * Qf * Qdmna / IA
       ElseIf IA < 0 Then                                   'IA ElseIf.
         Qf = UCF * qs * Qm(nm) * (-1)
         dH = -1 * Qf * Qdmna / IA
    End If                                                  'IA end if.
     GTC = 2
     GoTo Line4
Line2:   '=========================
ElseIf IMS = "Discrete, Volume" Then           'IMS elseif.++++++++++++++++++++++++++++++
For jm = 1 To m                                             'Initialize loops, separate to allow prior loops to total denominator, AI.
   For im = 1 To L                                            'DISTRIBUTE IMPULSE VOLUME THROUGHOUT GRID BY VOLUME.
   If CT(jm, im) = 1 Then                               'CT if - only impulse cells.
    If IA <> 0 Then                                     'IA If.
    Qc(jm, im) = UCF * qs * Qm(nm) * Abs(Qd(jm, im)) / IA 'Fill impulse volume.
    Else
      If IAP > 0 Then                                     'IAP if.
      msgim2 = "Impulses Net 0! "               'Zero NET impulse case.
      Qc(jm, im) = UCF * qs * Qm(nm) * Qd(jm, im) / IAP
      Else
      msgim2 = "Impulse Tab 0s! "                       'Zero impulse case.
      subsflag = 50
      Exit Sub
      End If                                            'End IAP end if.
    End If                                              'IA end if.
   End If                                               'CT end if
   Next im
Next jm
End If                                         'IMS end if.
'||||
  If ic = "Instantaneous" Then                 'ic if.++++++++++++++++++++++++++++++++++
  Qtt = 0
  i = 1
  For jm = 1 To m                                             'Initialize loops, separate to allow prior loops to total denominator, AI.
   For im = 1 To L                                          'CONVERT INSTANTANEOUS IMPULSE TO HEAD.
   If CT(jm, im) = 1 Then                             'CT if - only impulse cells.
         Qtt = Qtt + Qc(jm, im)                       'For Domain Volumetric Budget Later.
If ITS = "Deep Percolation" Then 'ITS if.
          sti = SfuncCt(CT(jm, im), SIT, "Confined", (Hnat(jm, im) + UMAT(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), _
                   widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im), _
                   SB, UMAT(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                   ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                   ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
         If THRCNTt(jm, im) > 1 Then 'Confinement threshold present.
          If Qc(jm, im) >= 0 Then 'H falling.
            For z = 1 To THRCNTt(jm, im)
             If (Hnat(jm, im) + UMAT(jm, im)) >= THRSt(jm, im, z) Then
              If (Hnat(jm, im) + UMAT(jm, im)) - Qc(jm, im) / (dx(im) * dy(jm) * sti) < THRSt(jm, im, z) Then 'Depressurization.
                 Hnat(jm, im) = HnDepres(THRSt(jm, im, 1), THRSt(jm, im, 2), THRSt(jm, im, 3), THRSt(jm, im, 4), THRSt(jm, im, 5), THRCNTt(jm, im),
_
                                  Qc(jm, im), Hnat(jm, im), dx(im), dy(jm), Sigs(jm, im), Sigma(jm, im), AqT(jm, im), AqB(jm, im), _
```

```
                                CT(jm, im), SIT, "Confined", ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, "T", widp(jm, im), lenp(jm, im), widpp(jm, im), _
                                Bed(jm, im), bpp(jm, im), bppb(jm, im), SB, UMAT(jm, im), UMS(jm, im))
                     If msgti = "" Then
                     msgti = "Impulse Distribution Through Confinement Transition. "
                     End If
                   Exit For
                  Else  'Hn...
                    Hnat(jm, im) = Hnat(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * sti) 'Pressurized.
                    Exit For
                 End If  'Hn...
               End If
               If z = THRCNTt(jm, im) Then
                Hnat(jm, im) = Hnat(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * sti) 'Depressurized below lowest threshold.
               End If
              Next z
             ElseIf Qc(jm, im) < 0 Then 'H rising.
              For z = THRCNTt(jm, im) To 1 Step -1
               If (Hnat(jm, im) + UMAT(jm, im)) < THRSt(jm, im, z) Then
                If (Hnat(jm, im) + UMAT(jm, im)) - Qc(jm, im) / (dx(im) * dy(jm) * sti) >= THRSt(jm, im, z) Then 'Pressurization.
                   Hnat(jm, im) = HnPres(THRSt(jm, im, 1), THRSt(jm, im, 2), THRSt(jm, im, 3), THRSt(jm, im, 4), THRSt(jm, im, 5), THRCNTt(jm, im), _
                                Qc(jm, im), Hnat(jm, im), dx(im), dy(jm), Sigs(jm, im), Sigma(jm, im), AqT(jm, im), AqB(jm, im), _
                                CT(jm, im), SIT, "Confined", ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, "T", widp(jm, im), lenp(jm, im), widpp(jm, im), _
                                Bed(jm, im), bpp(jm, im), bppb(jm, im), SB, UMAT(jm, im), UMS(jm, im))
                     If msgti = "" Then
                     msgti = "Impulse Distribution Through Confinement Transition. "
                     End If
                   Exit For
                  Else  'Hn-Qc...
                    Hnat(jm, im) = Hnat(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * sti) 'Depressurized.
                    Exit For
                 End If  'Hn-Qc...
               End If
               If z = 1 Then
                Hnat(jm, im) = Hnat(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * sti) 'Pressurized Above highest threshold.
               End If
              Next z
            End If 'Qc
           ElseIf THRCNTt(jm, im) = 1 Then 'No confinement threshold present.
            Hnat(jm, im) = Hnat(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * sti)
          End If                          'Confinement threshold end if.
            If Hnat(jm, im) < AqT(jm, im) - UMA(jm, im) Then 'Dewatering.
               GoTo Line3
            End If                              'Dewatering end if.
            If Hnat(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
               msgc = "Inundated Ground! "
Line3:   '=========================
               msgti = "Impulse Exceeds Aquitard Capacity. "
               flaw = 3
               NegHead = 6
               NegHeadNM = nm + 38
               subsflag = 50
               Exit Sub
```

```
         End If
Else                            'ITS else.
         Si = SfuncC(CT(jm, im), SIT, "Confined", (Hn(jm, im) + UMA(jm, im)), AqT(jm, im), widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im),
bpp(jm, im), bppb(jm, im), Ss(jm, im), Sy(jm, im), AqB(jm, im), _
                  SB, UMA(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), _
                  ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                  ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), dx(im), dy(jm))
         If THRCNT(jm, im) > 0 Then 'Confinement threshold present.
          If Qc(jm, im) >= 0 Then 'H falling.
            For z = 1 To THRCNT(jm, im)
             If (Hn(jm, im) + UMA(jm, im)) >= THRS(jm, im, z) Then
              If (Hn(jm, im) + UMA(jm, im)) - Qc(jm, im) / (dx(im) * dy(jm) * Si) < THRS(jm, im, z) Then 'Depressurization.
                 Hn(jm, im) = HnDepres(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), _
                                  Qc(jm, im), Hn(jm, im), dx(im), dy(jm), Ss(jm, im), Sy(jm, im), AqT(jm, im), AqB(jm, im), _
                                  CT(jm, im), SIT, "Confined", ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                  beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                  ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, "A", widp(jm, im), lenp(jm, im), widpp(jm, im), _
                                  Bed(jm, im), bpp(jm, im), bppb(jm, im), SB, UMA(jm, im), UMS(jm, im))
                    If msgti = "" Then
                    msgti = "Impulse Distribution Through Confinement Transition. "
                    End If
                Exit For
               Else  'Hn...
                 Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Si) 'Pressurized.
                 Exit For
              End If  'Hn...
             End If
             If z = THRCNT(jm, im) Then
              Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Si) 'Depressurized below lowest threshold.
             End If
            Next z
           ElseIf Qc(jm, im) < 0 Then 'H rising.
            For z = THRCNT(jm, im) To 1 Step -1
             If (Hn(jm, im) + UMA(jm, im)) < THRS(jm, im, z) Then
              If (Hn(jm, im) + UMA(jm, im)) - Qc(jm, im) / (dx(im) * dy(jm) * Si) >= THRS(jm, im, z) Then 'Pressurization.
                 Hn(jm, im) = HnPres(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), _
                                  Qc(jm, im), Hn(jm, im), dx(im), dy(jm), Ss(jm, im), Sy(jm, im), AqT(jm, im), AqB(jm, im), _
                                  CT(jm, im), SIT, "Confined", ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), _
                                  beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), _
                                  ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), z, "A", widp(jm, im), lenp(jm, im), widpp(jm, im), Bed(jm, im), _
                                  bpp(jm, im), bppb(jm, im), SB, UMA(jm, im), UMS(jm, im))
                    If msgti = "" Then
                    msgti = "Impulse Distribution Through Confinement Transition. "
                    End If
                Exit For
               Else  'Hn-Qc...
                 Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Si) 'Depressurized.
                 Exit For
              End If  'Hn-Qc...
             End If
             If z = 1 Then
              Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Si) 'Pressurized Above highest threshold.
             End If
            Next z
          End If 'Qc
        ElseIf THRCNT(jm, im) = 0 Then 'No confinement threshold present.
```

```
          Hn(jm, im) = Hn(jm, im) - Qc(jm, im) / (dx(im) * dy(jm) * Si)
        End If                            'Confinement threshold end if.
End If                           'ITS end if.
        Qc(jm, im) = 0                                'Annul converted flow (instantaneous).
      End If                                          'CT end if.
      i = i + 1
      Next im                                        'Increment to loop.
    Next jm
    ElseIf ic = "Steady" Then                     'ic elseif.++++++++++++++++++++++++++++++
    Qtt = 0
    For jm = 1 To m                                    'CONVERT STEADY IMPULSE TO FLOW.
    For im = 1 To L
      If CT(jm, im) = 1 Then
        Qtt = Qtt + Qc(jm, im)                        'For Domain Volumetric Budget Later.
        Qc(jm, im) = Qc(jm, im) / (dt)                'Convert to volume per time.
      End If
   Next im
   Next jm
  End If                                         'ic end if.
'||||
If ic = "Steady" Then                            'Restore Qm for later use.
    Qm(nm) = Qm(nm) * nc
End If
subsflag = 8
Exit Sub                                     'Direct past solver.++++++++++++++++++++++
'||||>
Line4:          '==========================
'START NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION OVER GRID BY HEAD]]]]]]]]]]
   kitr = 0                                          'Initialize iteration counter.
   IGTC = 0                                          'Initialize impulse go to code.
   ObF = 1                                           'Initialize objective function.
   Do While Abs(ObF) > 0.0000001        'Solver loop.
   If kitr < 500 Then                               'Iteration limit check
     kitr = kitr + 1                                'Increment iteration counter.
Line5:          '==========================
   Qt = 0                                           'Initialize virtual volume tally.
   '((((>
   i = 1
   For jm = 1 To m                                       'Prepare for calculation loop.
   For im = 1 To L                        'Calculation loop.
   If CT(jm, im) = 1 Then                           'CT if - cell type check.
     Qc(jm, im) = QCFC("Confined", ic, ITS, Qd(jm, im), dH, Qdmna, dx(im), dy(jm), Hn(jm, im), Hnat(jm, im), AqT(jm, im), _
                       Ss(jm, im), AqB(jm, im), Sy(jm, im), UMA(jm, im), SIT, widp(jm, im), lenp(jm, im), _
                       widpp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), Sigs(jm, im), Sigma(jm, im), _
                       SB, UMAT(jm, im), UMS(jm, im), ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), _
                       ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3), beds(jm, im, 3), ovp(jm, im, 4), _
                       ovw(jm, im, 4), beds(jm, im, 4), THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), _
                       THRS(jm, im, 4), THRS(jm, im, 5), THRCNT(jm, im), THRSt(jm, im, 1), THRSt(jm, im, 2), _
                       THRSt(jm, im, 3), THRSt(jm, im, 4), THRSt(jm, im, 5), THRCNTt(jm, im), CT(jm, im), DEC(jm, im))
     If IA <> 0 Then
     Qt = Qt + Qc(jm, im)                          'For distribution by Newton Solver.
     Else
     If Qd(jm, im) > 0 Then
     Qt = Qt + Qc(jm, im)
     End If
     End If
```

```
  End If                                      'CT end if.
   i = i + 1
  Next im
  Next jm                                       'Close calculation loop.
   '((((>
   If IGTC = 6 Then
   IGTC = 0
   GoTo Line6
   End If
   '((((>
   ObF = Qf - Qt                               'Compute objective function [F(n)].
  If Abs(ObF) <= 0.0000001 Then
  GoTo Line7
  End If
  dH = (1 + 0.00000001) * dH                    'Increment head increment.
   IGTC = 6
   GoTo Line5
Line6:   '==========================
  ObFF = Qf - Qt                               'Recompute objective function [F(n+1)].
  dFdu = (ObFF - ObF) / (dH - dH / (1 + 0.00000001)) 'Calculate derivative.
  dH = dH / (1 + 0.00000001)                   'Restore head increment.
  dH = dH - ObF / dFdu                         'Compute next estimate for head increment.
Line7:   '==========================
  Else                                                  'Intervene to end iteration if not converging in a reasonable number of cycles.
   NegHead = 4
   msgtt = "Impulse Distribution Iteration Limit Exceeded! "
   NegHeadNM = nm + 38
    subsflag = 50
    Exit Sub
  End If                                                'Iteration counter end if.
  Loop                                         'Close solver loop.
'END NEWTON-RAPHSON SOLVER FOR IMPULSE DISTRIBUTION]]]]]]]]]]
If GTC = 1 Then
 GoTo Line1
ElseIf GTC = 2 Then
 GoTo Line2
End If                                    'END IMPULSE DISTRIBUTION BLOCK.++++++++++
End Sub
'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************
Sub OVPERer(m As Integer, L As Integer, jm As Integer, im As Integer, ovp() As Double, ovw() As Double, Bed() As Double, beds() As Double, widpp() As
Double, _
            Lons() As Integer, widp() As Double, bppb() As Double, dx() As Double, dy() As Double, SIT As String, msgclt As String, NBRS() As
Integer, SEB As String)
For jm = 1 To m                  'Calculates bank overlap from adjacent cells.
 For im = 1 To L
    If jm - 1 > 0 Then
     ovp(jm, im, 1) = OVPfun(widpp(jm, im), Lons(jm, im), widp(jm - 1, im), widpp(jm - 1, im), bppb(jm - 1, im), Lons(jm - 1, im), 2, dx(im), dy(jm),
SIT, NBRS(jm - 1, im, 3), SEB)
       If ovp(jm, im, 1) >= dy(jm) / 2 Then
        msgclt = "Excessive bank overlap! "
        Exit Sub
       End If
```

```
     ovw(jm, im, 1) = OVWfun(widpp(jm, im), Lons(jm, im), widp(jm - 1, im), widpp(jm - 1, im), bppb(jm - 1, im), Lons(jm - 1, im), 2, dx(im), dy(jm),
SIT, NBRS(jm - 1, im, 3), SEB)
     beds(jm, im, 1) = Bed(jm - 1, im)
   End If
   If im - 1 > 0 Then
     ovp(jm, im, 2) = OVPfun(widpp(jm, im), Lons(jm, im), widp(jm, im - 1), widpp(jm, im - 1), bppb(jm, im - 1), Lons(jm, im - 1), 1, dx(im), dy(jm),
SIT, NBRS(jm, im - 1, 4), SEB)
       If ovp(jm, im, 2) >= dx(im) / 2 Then
        msgclt = "Excessive bank overlap! "
        Exit Sub
       End If
     ovw(jm, im, 2) = OVWfun(widpp(jm, im), Lons(jm, im), widp(jm, im - 1), widpp(jm, im - 1), bppb(jm, im - 1), Lons(jm, im - 1), 1, dx(im), dy(jm),
SIT, NBRS(jm, im - 1, 4), SEB)
     beds(jm, im, 2) = Bed(jm, im - 1)
   End If
   If jm + 1 < m + 1 Then
     ovp(jm, im, 3) = OVPfun(widpp(jm, im), Lons(jm, im), widp(jm + 1, im), widpp(jm + 1, im), bppb(jm + 1, im), Lons(jm + 1, im), 2, dx(im), dy(jm),
SIT, NBRS(jm + 1, im, 1), SEB)
       If ovp(jm, im, 3) >= dy(jm) / 2 Then
        msgclt = "Excessive bank overlap! "
        Exit Sub
       End If
     ovw(jm, im, 3) = OVWfun(widpp(jm, im), Lons(jm, im), widp(jm + 1, im), widpp(jm + 1, im), bppb(jm + 1, im), Lons(jm + 1, im), 2, dx(im), dy(jm),
SIT, NBRS(jm + 1, im, 1), SEB)
     beds(jm, im, 3) = Bed(jm + 1, im)
   End If
   If im + 1 < L + 1 Then
     ovp(jm, im, 4) = OVPfun(widpp(jm, im), Lons(jm, im), widp(jm, im + 1), widpp(jm, im + 1), bppb(jm, im + 1), Lons(jm, im + 1), 1, dx(im), dy(jm),
SIT, NBRS(jm, im + 1, 2), SEB)
       If ovp(jm, im, 4) >= dx(im) / 2 Then
        msgclt = "Excessive bank overlap! "
        Exit Sub
       End If
     ovw(jm, im, 4) = OVWfun(widpp(jm, im), Lons(jm, im), widp(jm, im + 1), widpp(jm, im + 1), bppb(jm, im + 1), Lons(jm, im + 1), 1, dx(im), dy(jm),
SIT, NBRS(jm, im + 1, 2), SEB)
     beds(jm, im, 4) = Bed(jm, im + 1)
   End If
Next im
Next jm
End Sub
'*************************************************************************************************************************************************
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
'*************************************************************************************************************************************************
Sub HoHo(hx() As Double, m As Integer, L As Integer, jm As Integer, im As Integer, ovp() As Double, ovw() As Double, beds() As Double, Bed() As
Double, _
         hob() As Double, ho() As Double, widpp() As Double, Lons() As Integer, widp() As Double, bppb() As Double, bpp() As Double, dx() As Double,
_
         dy() As Double, SIT As String, AqT() As Double, AqB() As Double, UMA() As Double, UMS() As Double, ATC As String, CT() As Integer, _
         lenp() As Double, SB As String, DEC() As Double)          'Calculates saturated thickness for each side of the cell, then an overall value.
Dim i As Integer, av As Double
i = 1
For jm = 1 To m
 For im = 1 To L
 av = DEC(jm, im) - AqB(jm, im)
```

```
   If jm - 1 > 0 Then
     hob(jm, im, 1) = hobfunC(hx(i), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm,
im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 1)
     If hob(jm, im, 1) > av Then
       hob(jm, im, 1) = av
     End If
   End If
   If im - 1 > 0 Then
     hob(jm, im, 2) = hobfunC(hx(i), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm,
im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 2)
     If hob(jm, im, 2) > av Then
       hob(jm, im, 2) = av
     End If
   End If
   If jm + 1 < m + 1 Then
     hob(jm, im, 3) = hobfunC(hx(i), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm,
im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 3)
     If hob(jm, im, 3) > av Then
       hob(jm, im, 3) = av
     End If
   End If
   If im + 1 < L + 1 Then
     hob(jm, im, 4) = hobfunC(hx(i), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm,
im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 4)
     If hob(jm, im, 4) > av Then
       hob(jm, im, 4) = av
     End If
   End If
   If CT(jm, im) <> 4 Then
    ho(jm, im) = hofuncC(hx(i), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm, im),
lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB)
     If ho(jm, im) > av Then
       ho(jm, im) = av
     End If
   End If
i = i + 1
Next im
Next jm
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub HoHo3(hx() As Double, m As Integer, L As Integer, jm As Integer, im As Integer, ovp() As Double, ovw() As Double, beds() As Double, Bed() As
Double, _
```

```
        hob() As Double, ho() As Double, widpp() As Double, Lons() As Integer, widp() As Double, bppb() As Double, bpp() As Double, dx() As Double,
_
        dy() As Double, SIT As String, AqT() As Double, AqB() As Double, UMA() As Double, UMS() As Double, ATC As String, CT() As Integer, _
        lenp() As Double, SB As String)            'Calculates saturated thickness for each side of the cell, then an overall value.
Dim i As Integer, r As Integer
r = m * L * 2
i = 1
For jm = 1 To m
 For im = 1 To L
   If jm - 1 > 0 Then
     hob(jm, im, 1) = hobfunC(hx(i + r / 2), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm),
widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 1)
   End If
   If im - 1 > 0 Then
     hob(jm, im, 2) = hobfunC(hx(i + r / 2), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm),
widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 2)
   End If
   If jm + 1 < m + 1 Then
     hob(jm, im, 3) = hobfunC(hx(i + r / 2), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm),
widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 3)
   End If
   If im + 1 < L + 1 Then
     hob(jm, im, 4) = hobfunC(hx(i + r / 2), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm),
widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 4)
   End If
   If CT(jm, im) <> 4 Then
    ho(jm, im) = hofuncC(hx(i + r / 2), AqT(jm, im), AqB(jm, im), UMA(jm, im), UMS(jm, im), ATC, CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm,
im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB)
   End If
i = i + 1
Next im
Next jm
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub HoHoat(i As Integer, r As Integer, hx() As Double, Hn() As Double, m As Integer, L As Integer, jm As Integer, im As Integer, ovp() As Double, _
          ovw() As Double, beds() As Double, hobat() As Double, hoat() As Double, widpp() As Double, Lons() As Integer, widp() As Double, bppb() _
          As Double, dx() As Double, dy() As Double, SIT As String, AqT() As Double, AqB() As Double, UMAT() As Double, UMS() As Double, _
          DEC() As Double, CT() As Integer, lenp() As Double, Bed() As Double, bpp() As Double, SB As String, rpc() As Integer, UMA() As Double)
i = 1                                    'Calculates saturated thickness for each side of aquitard cell, then an overall value.
For jm = 1 To m
 For im = 1 To L
   If jm - 1 > 0 Then
```

```
     hobat(jm, im, 1) = hobatfunC(rpc(jm, im), hx(i), hx(i + r / 2), Hn(jm, im), AqT(jm, im), AqB(jm, im), UMAT(jm, im), UMS(jm, im), "Confined",
CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 1, UMA(jm, im))
   End If
   If im - 1 > 0 Then
     hobat(jm, im, 2) = hobatfunC(rpc(jm, im), hx(i), hx(i + r / 2), Hn(jm, im), AqT(jm, im), AqB(jm, im), UMAT(jm, im), UMS(jm, im), "Confined",
CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 2, UMA(jm, im))
   End If
   If jm + 1 < m + 1 Then
     hobat(jm, im, 3) = hobatfunC(rpc(jm, im), hx(i), hx(i + r / 2), Hn(jm, im), AqT(jm, im), AqB(jm, im), UMAT(jm, im), UMS(jm, im), "Confined",
CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 3, UMA(jm, im))
   End If
   If im + 1 < L + 1 Then
     hobat(jm, im, 4) = hobatfunC(rpc(jm, im), hx(i), hx(i + r / 2), Hn(jm, im), AqT(jm, im), AqB(jm, im), UMAT(jm, im), UMS(jm, im), "Confined",
CT(jm, im), widp(jm, im), dx(im), dy(jm), widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, 4, UMA(jm, im))
   End If
   If CT(jm, im) <> 4 Then
    hoat(jm, im) = hoatfuncC(rpc(jm, im), hx(i), hx(i + r / 2), Hn(jm, im), AqT(jm, im), AqB(jm, im), UMAT(jm, im), UMS(jm, im), "Confined", CT(jm,
im), widp(jm, im), dx(im), dy(jm), widpp(jm, im), lenp(jm, im), Bed(jm, im), bpp(jm, im), bppb(jm, im), _
                ovp(jm, im, 1), ovw(jm, im, 1), beds(jm, im, 1), ovp(jm, im, 2), ovw(jm, im, 2), beds(jm, im, 2), ovp(jm, im, 3), ovw(jm, im, 3),
beds(jm, im, 3), ovp(jm, im, 4), ovw(jm, im, 4), beds(jm, im, 4), SIT, SB, UMA(jm, im))
   End If
  If AqT(jm, im) + hoat(jm, im) > DEC(jm, im) Then
   hoat(jm, im) = DEC(jm, im) - AqT(jm, im)
  End If
i = i + 1
Next im
Next jm
End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub Hoot(m As Integer, L As Integer, jm As Integer, im As Integer, ho() As Double, hoo() As Double)
For jm = 1 To m
 For im = 1 To L
    hoo(jm, im) = ho(jm, im)
Next im
Next jm
End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
'Subroutine to calculate stream response flow from solved heads.
Sub Qnout(RSP() As Double, RSPA() As Double, Qnno() As Double, i As Integer, jm As Integer, im As Integer, m As Integer, L As Integer, CT() As
Integer, _
```

```
          SB As String, Bed() As Double, bpp() As Double, AqT() As Double, hx() As Double, UMS() As Double, kpp() As Double, widpp() As Double, _
          lenp() As Double, SHD() As Double, msgbg As String, SIT As String, WRAT() As Double, Hnnat() As Double, Hnn() As Double, _
          CC() As Double, DD() As Double, AA() As Double, BB() As Double, _
          GG() As Double, UMA() As Double, AqTi() As Double, CCt() As Double, DDt() As Double, AAt() As Double, BBt() As Double, zone() As String, _
          qrs As Double, dt As Double, UCF As Double, n As Long, qnoc() As Double, r As Integer, Lons() As Integer, CTP() As Integer, BedCP() As
Double)
 i = 1
For jm = 1 To m
  For im = 1 To L                    'Solved heads allow explicit calculation of flow through streambed.
    If CTP(jm, im) = 2 Then             'Restrictive bed flow by Darcy's law.
      If SB = "Restrictive" Then
      If SIT <> "None" Then
      If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
       If hx(i) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then          'Check for drawdown below bottom of bed, bed in aquitard.
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnnat(jm, im) - SHD(jm, im)) / bpp(jm, im)  'Full bed flux term.
       ElseIf hx(i) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im) - UMS(jm, im)) / bpp(jm,
im) 'Bed flux limited by gradient to bottom of bed.
       msgbg = "Bed Flux Limited by Desaturation. "
       End If
      ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
       If hx(i + r / 2) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then          'Check for drawdown below bottom of bed, bed below aquitard.
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnn(jm, im) - SHD(jm, im)) / bpp(jm, im)  'Full bed flux term.
       ElseIf hx(i + r / 2) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im) - UMS(jm, im)) / bpp(jm,
im) 'Bed flux limited by gradient to bottom of bed.
       msgbg = "Bed Flux Limited by Desaturation. "
       End If
      End If
      Else 'SIT
      If Bed(jm, im) > AqT(jm, im) Then
       If hx(i) > (Bed(jm, im) - bpp(jm, im)) Then         'Check for drawdown below bottom of bed, bed in aquitard.
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnnat(jm, im) - SHD(jm, im)) / bpp(jm, im)  'Full bed flux term.
       ElseIf hx(i) <= (Bed(jm, im) - bpp(jm, im)) Then
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im)) / bpp(jm, im) 'Bed flux
limited by gradient to bottom of bed.
       msgbg = "Bed Flux Limited by Desaturation. "
       End If
      ElseIf Bed(jm, im) <= AqT(jm, im) Then
       If hx(i + r / 2) > (Bed(jm, im) - bpp(jm, im)) Then          'Check for drawdown below bottom of bed, bed below aquitard.
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnn(jm, im) - SHD(jm, im)) / bpp(jm, im)  'Full bed flux term.
       ElseIf hx(i + r / 2) <= (Bed(jm, im) - bpp(jm, im)) Then
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im)) / bpp(jm, im) 'Bed flux
limited by gradient to bottom of bed.
       msgbg = "Bed Flux Limited by Desaturation. "
       End If
      End If
      End If
      ElseIf SB = "Permissive" Then
       Qnno(jm, im) = 0
       If hx(i) > AqT(jm, im) Then
        If jm - 1 > 0 Then
         If CTP(jm - 1, im) Mod 2 = 1 Then                        'Not class 2 or 4 (remainder of division is 1).
          If qnoc(jm - 1, im, 3) <> 0 Then
            Qnno(jm, im) = Qnno(jm, im) - qnoc(jm - 1, im, 3)      'Cell side seepage into dewatered cell (then down to aquifer).
          End If
```

```
         End If
        End If
        If im - 1 > 0 Then
         If CTP(jm, im - 1) Mod 2 = 1 Then
          If qnoc(jm, im - 1, 4) <> 0 Then
            Qnno(jm, im) = Qnno(jm, im) - qnoc(jm, im - 1, 4)
          End If
         End If
        End If
        If jm + 1 < m + 1 Then
         If CTP(jm + 1, im) Mod 2 = 1 Then
          If qnoc(jm + 1, im, 1) <> 0 Then
            Qnno(jm, im) = Qnno(jm, im) - qnoc(jm + 1, im, 1)
          End If
         End If
        End If
        If im + 1 < L + 1 Then
         If CTP(jm, im + 1) Mod 2 = 1 Then
          If qnoc(jm, im + 1, 2) <> 0 Then
            Qnno(jm, im) = Qnno(jm, im) - qnoc(jm, im + 1, 2)
          End If
         End If
        End If
       End If
        If BedCP(jm, im) <= AqT(jm, im) Then
        If jm - 1 > 0 Then              'Permissive bed flow is cell flow, by Darcy's law, no change in storage.
         If CTP(jm - 1, im) Mod 2 = 1 Then
          Qnno(jm, im) = Qnno(jm, im) + CC(jm - 1, im) * (Hnn(jm - 1, im) - Hnn(jm, im))
         End If
        End If
        If im - 1 > 0 Then
         If CTP(jm, im - 1) Mod 2 = 1 Then
          Qnno(jm, im) = Qnno(jm, im) + DD(jm, im - 1) * (Hnn(jm, im - 1) - Hnn(jm, im))
         End If
        End If
        If jm + 1 < m + 1 Then
         If CTP(jm + 1, im) Mod 2 = 1 Then
          Qnno(jm, im) = Qnno(jm, im) + AA(jm + 1, im) * (Hnn(jm + 1, im) - Hnn(jm, im))
         End If
        End If
        If im + 1 < L + 1 Then
         If CTP(jm, im + 1) Mod 2 = 1 Then
          Qnno(jm, im) = Qnno(jm, im) + BB(jm, im + 1) * (Hnn(jm, im + 1) - Hnn(jm, im))
         End If
        End If
        ElseIf BedCP(jm, im) > AqT(jm, im) Then
         Qnno(jm, im) = Qnno(jm, im) + GG(jm, im) * (Hnn(jm, im) - Hnnat(jm, im))
           If Hnn(jm, im) <= AqTi(i) - UMA(jm, im) Then                                          'Aquitard flux limited (aquifer head drawn down
below top, b adjusted in coefficient block).
            If Hnnat(jm, im) > AqTi(i) Then
             Qnno(jm, im) = Qnno(jm, im) - GG(jm, im) * (Hnn(jm, im) - (AqT(jm, im) - UMA(jm, im)))
            End If
           End If
        End If                      'bed end if.
          If jm - 1 > 0 Then
           If CTP(jm - 1, im) Mod 2 = 1 Then
```

```
            Qnno(jm, im) = Qnno(jm, im) + CCt(jm - 1, im) * (Hnnat(jm - 1, im) - Hnnat(jm, im)) 'bed or sides above aquitard.
           End If
          End If
          If im - 1 > 0 Then
           If CTP(jm, im - 1) Mod 2 = 1 Then
            Qnno(jm, im) = Qnno(jm, im) + DDt(jm, im - 1) * (Hnnat(jm, im - 1) - Hnnat(jm, im))
           End If
          End If
          If jm + 1 < m + 1 Then
           If CTP(jm + 1, im) Mod 2 = 1 Then
            Qnno(jm, im) = Qnno(jm, im) + AAt(jm + 1, im) * (Hnnat(jm + 1, im) - Hnnat(jm, im))
           End If
          End If
          If im + 1 < L + 1 Then
           If CTP(jm, im + 1) Mod 2 = 1 Then
            Qnno(jm, im) = Qnno(jm, im) + BBt(jm, im + 1) * (Hnnat(jm, im + 1) - Hnnat(jm, im))
           End If
          End If
       End If                                    'SB end if.
      Qnno(jm, im) = qrs * dt * Qnno(jm, im) / UCF  'Flow given correct sign, multiplied by dt to convert to units of volume, converted to acre-feet
if applicable.
     RSP(n) = RSP(n) + Qnno(jm, im) 'Aggregate.
     If zone(jm, im) = 1 Then       'Differentiate.
      RSPA(n) = RSPA(n) + Qnno(jm, im) '.....................................................................................
     End If
    End If                                     'CT end if.
     i = i + 1
   Next im
Next jm
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'Subroutine to calculate secondary output for delayed yield aquifers.
Sub SOB(St As String, PD As String, no As Long, NegHead As Integer, hrb As Variant, RSPOFM As Integer, CPC As Variant, RSPO() As Double, _
Qm() As Double, IMPST As Double, IMPSMX As Double, IMPSMN As Double, RSPT As Double)

Dim SO As String, n As Integer, tt As Integer, SOV() As Variant, CUMRn() As Double, _
CUMI As Double, CUMR As Double, IFT As Double, FCR As Double                                    'Dimension initial selection.

SO = WorksheetFunction.Proper(Sheet11.Cells(19, 5).Value) 'Load secondary output type.

ReDim SOV(1 To no)
ReDim CUMRn(0 To no)

If IMPSMX <> 999999999.123457 Then
If IMPST <> 0 Then                    'SECONDARY OUTPUT BLOCK
If SO = "Truncated Response" Then    'Physical residual distributed evenly for single impulse scenario.
 n = 1
 Do While n < no + 1
  SOV(n) = RSPO(n) + (IMPST - RSPT) / no
  n = n + 1
 Loop
End If
```

```
If SO = "Cumulative Ratio" Then      'Ratio of cumulative response to impulse through each time step.
 CUMI = 0
 CUMR = 0
 n = 1
 Do While n < no + 1
  CUMI = CUMI + Qm(n)
  CUMR = CUMR + RSPO(n)
  If CUMI <> 0 Then
  SOV(n) = CUMR / CUMI
  Else
  SOV(n) = "-"
  End If
  n = n + 1
 Loop
End If
If SO = "Response Ratio" Then        'Ratio of response for time step to impulse total.
 n = 1
 Do While n < no + 1
  SOV(n) = RSPO(n) / IMPST
  n = n + 1
 Loop
End If
If SO = "Period Ratio" Then        'Ratio of response for time step to impulse for time step.
 n = 1
 Do While n < no + 1
  If Qm(n) <> 0 Then
  SOV(n) = RSPO(n) / Qm(n)
  Else
  SOV(n) = "-"
  End If
  n = n + 1
 Loop
End If
If SO = "Impact Factor" Then         'Fraction of annual response on symmetrical rolling basis.
 If St = "Annual Pattern" Then
 If NegHead = 1 Then
 IFT = 0
 CUMRn(0) = 0
 n = 1
 Do While n < no + 1
  CUMRn(n) = CUMRn(n - 1) + RSPO(n)
  n = n + 1
 Loop
 If PD = "Days" Then
  tt = 365 * WorksheetFunction.RoundDown(no / 365, 0) - ((2 * 365) - 1)
  n = 1
  Do While n < no + 1
   If n < 365 + 1 Then
    SOV(n) = RSPO(tt + n - 1) / (CUMRn(tt + n - 1 + (365 - 1) / 2) - CUMRn(tt + n - 1 - 1 - (365 - 1) / 2))
    IFT = IFT + SOV(n)
   Else
    SOV(n) = "-"
   End If
   n = n + 1
  Loop
  For n = 1 To 365
```

```
   SOV(n) = SOV(n) / IFT
  Next n
 Else
  tt = 12 * WorksheetFunction.RoundDown(no / 12, 0) - ((2 * 12) - 1)
  n = 1
  Do While n < no + 1
   If n < 12 + 1 Then
    SOV(n) = RSPO(tt + n - 1) / (((CUMRn(tt + n - 1 + 5) - CUMRn(tt + n - 1 - 7)) + (CUMRn(tt + n - 1 + 6) - CUMRn(tt + n - 1 - 6))) / 2)
    IFT = IFT + SOV(n)
   Else
    SOV(n) = "-"
   End If
   n = n + 1
  Loop
  For n = 1 To 12
   SOV(n) = SOV(n) / IFT
  Next n
 End If                         'PD end if.
 End If                         'NegHead end if.
 End If                         'ST end if.
End If                          'SO end if.
FCR = RSPT / IMPST
End If                          'IMPST end if.
End If                          'IMPSMX end if.

Sheet11.Unprotect Password:=CPC

If hrb > 38 Then
 If IMPSMX <> 999999999.123457 Then
   If hrb = 38 + no Then
    If RSPOFM = 1 Then
      Sheet11.Range("g39:" & "g" & hrb).Value = WorksheetFunction.Transpose(SOV())
      Sheet11.Cells(29, 5).Value = FCR                                          'Output final computed cumulative ratio of response to impulse.
    End If
   End If
 End If
End If

Sheet11.Protect Password:=CPC

End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub Messenger(msgc As String, msgtp As String, msgim As String, msgim2 As String, msgt As String, msgti As String, msgbg As String, _
msgat As String, msgtt As String, msgh As String, msgres As String, msg As String, msgmt As String, msgtg As String, msgep As String, CPC As Variant,
_
msgu As String, msgumat As String, msgclt As String, msgwp As String, msgsep As String, msgctp As String, msgrza As String)
Dim i As Integer

If Left(msgtt, 3) = "Dew" Then
msgtg = ""
End If
```

```
If msgtt <> "" Then                        'Final message prep.
 msgres = ""                                  'If iteration limit exceeded, residual order not reported.
  If msg = "None!" Then
    msg = msgtt
    Else
    msg = msgtt & msg
  End If
End If
If msgc <> "" Then
  If msg = "None!" Then
    msg = msgc
    Else
    msg = msg & msgc
  End If
End If
If msgt <> "" Then
  If msg = "None!" Then
    msg = msgt
    Else
    msg = msg & msgt
  End If
End If
If msgat <> "" Then
  If msg = "None!" Then
    msg = msgat
    Else
    msg = msg & msgat
  End If
End If
If msgh <> "" Then
  If msg = "None!" Then
    msg = msgh
    Else
    msg = msgh & msg
  End If
End If
If msgim <> "" Then
  If msg = "None!" Then
    msg = msgim
    Else
    msg = msg & msgim
  End If
End If
If msgep <> "" Then
  If msg = "None!" Then
    msg = msgep
    Else
    msg = msg & msgep
  End If
End If

Sheet11.Unprotect Password:=CPC
If msgsep = "1" Then
 Sheet11.Cells(10, 20) = 1
 msgsep = "Reconfigure Incision-Divided Stream Cell(s)! "
 Else
```

```
 msgsep = ""
End If
If Len(msg & msgim2 & msgbg & msgti & msgmt & msgtg & msgu & msgumat & msgclt & msgwp & msgsep & msgres & msgctp & msgrza & msgtt) > 18 Then
Sheet11.Cells(30, 5).Value = "Posted at J80."
Sheet11.Range("j80:j158").ClearContents
Sheet11.Cells(80, 10).Value = "Calculation Notes"
i = 0
If msgc <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgc  'Output messages.
End If
If msgtp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtp
End If
If msgim <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgim
End If
If msgim2 <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgim2
End If
If msgat <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgat
End If
If msgt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgt
End If
If msgti <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgti
End If
If msgtg <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtg
End If
If msgbg <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgbg
End If
If msgmt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgmt
End If
If msgtt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtt
End If
If msgh <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgh
End If
If msgep <> "" Then
```

```
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgep
End If
If msgu <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgu
End If
If msgumat <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgumat
End If
If msgclt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgclt
End If
If msgwp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgwp
End If
If msgsep <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgsep
End If
If msgres <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgres
End If
If msgctp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgctp
End If
If msgrza <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgrza
End If
Else
Sheet11.Range("j80:j158").ClearContents
Sheet11.Cells(30, 5).Value = msg & msgim2 & msgbg & msgti & msgmt & msgtt & msgtg & msgu & msgumat & msgclt & msgwp & msgsep & msgres & msgctp &
msgrza
End If
Sheet11.Protect Password:=CPC

End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub MessengerA(msg As String, msgim2 As String, msgbg As String, msgti As String, msgmt As String, msgtg As String, msgu As String, msgumat As
String, _
msgclt As String, msgwp As String, msgc As String, msgtp As String, msgh As String, msgim As String, msgt As String, CPC As Variant, _
msgtt As String, msgep As String, msgres)
Dim i As Integer

If msgtt <> "" Then                  'Final message prep.
 msgres = ""                              'If iterations exceeded or cell dewatered, etc. residual order not reported.
```

```
  If msg = "None!" Then
    msg = msgtt
    Else
    msg = msgtt & msg
  End If
End If
If msgc <> "" Then
  If msg = "None!" Then
    msg = msgc
    Else
    msg = msg & msgc
  End If
End If
If msgt <> "" Then
  If msg = "None!" Then
    msg = msgt
    Else
    msg = msg & msgt
  End If
End If
If msgh <> "" Then
  If msg = "None!" Then
    msg = msgh
    Else
    msg = msgh & msg
  End If
End If
If msgim <> "" Then
  If msg = "None!" Then
    msg = msgim
    Else
    msg = msg & msgim
  End If
End If
If msgep <> "" Then
  If msg = "None!" Then
    msg = msgep
    Else
    msg = msg & msgep
  End If
End If

Sheet11.Unprotect Password:=CPC

If Len(msg & msgim2 & msgbg & msgti & msgmt & msgtg & msgu & msgumat & msgclt & msgwp & msgres & msgtt) > 18 Then
Sheet11.Cells(30, 5).Value = "Posted at J80."
Sheet11.Range("j80:j158").ClearContents
Sheet11.Cells(80, 10).Value = "Calculation Notes"
i = 0
If msgc <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgc  'Output message.
End If
If msgtp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtp
```

```
End If
If msgh <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgh
End If
If msgim <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgim
End If
If msgim2 <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgim2
End If
If msgt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgt
End If
If msgti <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgti
End If
If msgtg <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtg
End If
If msgbg <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgbg
End If
If msgmt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgmt
End If
If msgtt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgtt
End If
If msgep <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgep
End If
If msgu <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgu
End If
If msgumat <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgumat
End If
If msgclt <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgclt
End If
If msgwp <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgwp
```

```
End If
If msgres <> "" Then
i = i + 1
Sheet11.Cells(80 + i, 10).Value = msgres
End If
Else
Sheet11.Range("j80:j158").ClearContents
Sheet11.Cells(30, 5).Value = msg & msgim2 & msgbg & msgti & msgtg & msgmt & msgtt & msgu & msgumat & msgclt & msgwp & msgres
End If

Sheet11.Protect Password:=CPC

End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub TSC()                                                'Subroutine to set Time Series Chart ranges to match input scenario.
Dim NSP As Variant, re As Variant, xr As Variant, ir As Variant       'Number of simulation periods, range extent of plot series, x range, input
range.
Dim u1r As Variant, ur As Variant, lsp As Variant, tsp As Variant     'Output range, x (time) label spacing, x (time) gridline spacing.
Dim OFP As Variant, ONP As Variant, OPG As Variant                    'First period charted, number of periods charted, gridline spacing.
Dim rb As Variant, Sp As Variant                                      'Range beginning of plot series, spacing of gridlines.
Dim IT As String, SN As String, SD As String, PD As String, MD As String  'Dimension impulse type, site name, scenario description, period
denomination, magnitude denomination.
Dim CPC As Variant
CPC = "*******"

Sheets("Time").Select
IT = WorksheetFunction.Proper(Sheet3.Cells(7, 8).Value)                    'Load impulse type.
NSP = Sheet11.Cells(13, 5).Value                            'Loads number of simulation periods.
OFP = Sheet11.Cells(20, 5).Value                            'Loads first period plotted.
ONP = Sheet11.Cells(22, 5).Value                           'Loads number of periods plotted.
OPG = Sheet11.Cells(21, 5).Value                           'Loads plot gridline spacing.
SN = Sheet11.Cells(7, 5).Value                            'Loads site name.
SD = Sheet11.Cells(8, 5).Value                            'Loads scenario description.
PD = Sheet11.Cells(38, 3).Value                            'Loads period denomination.
MD = Sheet11.Cells(38, 4).Value                            'Loads magnitude denomination.
 rb = 38 + OFP                                              'Calculates plot range beginning.
 re = rb + ONP - 1                                          'Calculates range extent of plot series per specified number of periods.
xr = "=Time!R" & rb & "C2:R" & re & "C2"                   'Concatenates x (time) plot series range.
ir = "=Time!R" & rb & "C4:R" & re & "C4"                   'Concatenates input plot series range.
u1r = "=Time!R" & rb & "C5:R" & re & "C5"                   'Concatenates output plot series range, zone 1.
ur = "=Time!R" & rb & "C6:R" & re & "C6"                   'Concatenates output plot series range, total.
lsp = OPG                                                     'Override time gridline label.
tsp = OPG                                                     'Override time gridline spacing.
Sheet11.Unprotect Password:=CPC                                 'Prepare for chart update.
ActiveSheet.Shapes.Range(Array("TextBox 1")).Select
 Selection.ShapeRange(1).TextFrame2.TextRange.Characters.Text = SN
 ActiveSheet.Shapes.Range(Array("TextBox 5")).Select
 Selection.ShapeRange(1).TextFrame2.TextRange.Characters.Text = SD
ActiveSheet.ChartObjects("Chart 2").Activate                    'Activates plot.
    ActiveChart.Axes(xlValue).AxisTitle.Select
    ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Magnitude " & MD  'Vertical axis title.
    ActiveChart.Axes(xlCategory).AxisTitle.Select
```

```
    ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Period"   ' " & PD  'Horizontal axis title.
ActiveChart.SeriesCollection(1).XValues = xr              'Sets input series x (time) plot series range.
ActiveChart.SeriesCollection(1).Values = ir               'Sets input series y (input) plot series range.
ActiveChart.SeriesCollection(2).XValues = xr              'Sets output series x (time) plot series range.
ActiveChart.SeriesCollection(2).Values = ur               'Sets output series y (output) plot series range, total.
ActiveChart.SeriesCollection(3).XValues = xr              'Sets output series x (time) plot series range.
ActiveChart.SeriesCollection(3).Values = u1r               'Sets output series y (output) plot series range, zone 1.
ActiveChart.Axes(xlValue).MinimumScaleIsAuto = True
 If Sheet11.Cells(10, 13).Value < 0 Then
  If Abs(Sheet11.Cells(10, 13).Value) < 0.00001 Then
   ActiveChart.Axes(xlValue).MinimumScale = 0
  End If
 End If
ActiveChart.SeriesCollection(1).Select
    With Selection.Format.Fill
                If IT = "Deep Percolation" Then
                  .ForeColor.RGB = RGB(0, 176, 80)         'green
                 ElseIf IT = "Well Injection" Then
                  .ForeColor.RGB = RGB(255, 192, 0)         'orange
                 Else
                  .ForeColor.RGB = RGB(153, 51, 255)        'purple
                End If
      End With
ActiveChart.SeriesCollection(2).Select
    With Selection.Format.Fill
                If IT = "Deep Percolation" Then
                  .ForeColor.RGB = RGB(0, 0, 255)           'blue
                 ElseIf IT = "Well Injection" Then
                  .ForeColor.RGB = RGB(0, 0, 255)           'blue
                 Else
                  .ForeColor.RGB = RGB(51, 204, 204)         'teal
                End If
      End With
With ActiveChart.Axes(xlCategory)                          'Selects x axis category.
 .TickLabelSpacing = lsp                                    'Sets label spacing.
 .TickMarkSpacing = tsp                                     'Sets gridline spacing.
End With                                                   'End with.
Sheet11.Protect Password:=CPC                               'Restore sheet.

End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub HTSC()                                                 'Subroutine to set Output Head Time Series Chart ranges and titles to match scenario.
Dim NSP As Integer, rb As Integer, re As Integer                'Number of simulation periods, range beginning of plot series, range extent of plot
series.
Dim xr As Variant, hor As Variant, hor2 As Variant              'X range, head output range.
Dim ni As Integer, nj As Integer, serp As Integer               'Node indeces, series past.
Dim hu As String, pds As String, AqType As String               'Head units, period denomination.
Dim CPC As Variant
CPC = "********"
Dim selh As Range

NSP = Sheet11.Cells(13, 5).Value                           'Loads number of simulation periods.
```

```
AqType = Sheet3.Cells(9, 8).Value

Sheets("Head").Select
Set selh = Selection                                    'Loads initial selection on tab.

ni = Sheet10.Cells(134, 7).Value                        'Loads i index.
nj = Sheet10.Cells(135, 7).Value                        'Loads j index.
pds = Sheet10.Cells(139, 2).Value                       'Loads period denomination.
hu = Sheet10.Cells(139, 3).Value                        'Loads head units.
serp = Sheet10.Cells(140, 7).Value                      'Loads past series count.
 rb = 140                                               'Calculates plot range beginning.
 re = rb + NSP                                          'Calculates range extent of plot series per specified number of periods.
xr = "=Head!R" & rb & "C2:R" & re & "C2"                'Concatenates x (time) plot series range.
hor = "=Head!R" & rb & "C3:R" & re & "C3"               'Concatenates head output plot series range.
    If AqType = "Delayed" Then
      hor2 = "=Head!R" & rb & "C4:R" & re & "C4"
    End If
Sheet10.Unprotect Password:=CPC                         'Prepare for chart update.
ActiveSheet.ChartObjects("Chart 11").Activate           'Activates plot.

    ActiveChart.Axes(xlCategory).Select
    ActiveChart.Axes(xlCategory).MinimumScale = 0
    ActiveChart.Axes(xlCategory).MaximumScale = NSP     'Sets horizontal axis extent.
    ActiveChart.Axes(xlCategory).MajorUnitIsAuto = True
    If ActiveChart.Axes(xlCategory).MajorUnit < 1 Then
      ActiveChart.Axes(xlCategory).MajorUnit = 1
    End If

    ActiveChart.SeriesCollection(1).Select
    ActiveChart.SeriesCollection(1).Name = "=""Aquifer"""
    ActiveChart.SeriesCollection(1).XValues = xr        'Sets x (time) range of data to plot.
    ActiveChart.SeriesCollection(1).Values = hor        'Sets y (head output) range of data to plot.

    If AqType = "Delayed" Then
     If serp = 2 Then
      ActiveChart.SeriesCollection(2).XValues = xr        'Sets x (time) range of data to plot.
      ActiveChart.SeriesCollection(2).Values = hor2       'Sets y (head output) range of data to plot.
     Else
      ActiveChart.SeriesCollection.NewSeries
      ActiveChart.SeriesCollection(2).Select
      ActiveChart.SeriesCollection(2).Name = "=""Aquitard"""
      ActiveChart.SeriesCollection(2).XValues = xr
      ActiveChart.SeriesCollection(2).Values = hor2
      With Selection
        .MarkerStyle = 2
        .MarkerSize = 5
        .Border.Color = RGB(128, 128, 128)
        .Border.Weight = xlMedium
        .MarkerForegroundColor = RGB(255, 255, 255)
        .MarkerBackgroundColor = RGB(167, 167, 167)
      End With
      ActiveChart.SetElement (msoElementLegendBottom)
      ActiveChart.Legend.Select
      Selection.Left = 170.482
      Selection.Width = 325.284
     End If
```

```
    Sheet10.Cells(140, 7).Value = 2
    Else
     If serp = 2 Then
      ActiveChart.Legend.Delete
      ActiveChart.SeriesCollection(2).Delete
     End If
    Sheet10.Cells(140, 7).Value = 1
    End If

    ActiveChart.ChartTitle.Text = "Head at Node " & ni & ", " & nj  'Customizes chart title.

    ActiveChart.Axes(xlValue).AxisTitle.Select
    ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Text = "Magnitude " & hu  'Vertical axis title.

    ActiveChart.Axes(xlCategory).AxisTitle.Select
    ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "Period"   '& pds  'Horizontal axis title.

selh.Select                     'Reassigns activity to the initial selection in the Head sheet.
Sheet10.Protect Password:=CPC

End Sub
'*************************************************************************************************************************************************
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
Function crf(L)        'Function to index columns for surface chart of differential head.
Dim cr As String
If L = 1 Then
 cr = "C"
 ElseIf L = 2 Then
 cr = "D"
 ElseIf L = 3 Then
 cr = "E"
 ElseIf L = 4 Then
 cr = "F"
 ElseIf L = 5 Then
 cr = "G"
 ElseIf L = 6 Then
 cr = "H"
 ElseIf L = 7 Then
 cr = "I"
 ElseIf L = 8 Then
 cr = "J"
 ElseIf L = 9 Then
 cr = "K"
 ElseIf L = 10 Then
 cr = "L"
 ElseIf L = 11 Then
 cr = "M"
 ElseIf L = 12 Then
 cr = "N"
 ElseIf L = 13 Then
 cr = "O"
 ElseIf L = 14 Then
 cr = "P"
 ElseIf L = 15 Then
```

```
cr = "Q"
ElseIf L = 16 Then
cr = "R"
ElseIf L = 17 Then
cr = "S"
ElseIf L = 18 Then
cr = "T"
ElseIf L = 19 Then
cr = "U"
ElseIf L = 20 Then
cr = "V"
ElseIf L = 21 Then
cr = "W"
ElseIf L = 22 Then
cr = "X"
ElseIf L = 23 Then
cr = "Y"
ElseIf L = 24 Then
cr = "Z"
ElseIf L = 25 Then
cr = "AA"
ElseIf L = 26 Then
cr = "AB"
ElseIf L = 27 Then
cr = "AC"
ElseIf L = 28 Then
cr = "AD"
ElseIf L = 29 Then
cr = "AE"
ElseIf L = 30 Then
cr = "AF"
ElseIf L = 31 Then
cr = "AG"
ElseIf L = 32 Then
cr = "AH"
ElseIf L = 33 Then
cr = "AI"
ElseIf L = 34 Then
cr = "AJ"
ElseIf L = 35 Then
cr = "AK"
ElseIf L = 36 Then
cr = "AL"
ElseIf L = 37 Then
cr = "AM"
ElseIf L = 38 Then
cr = "AN"
ElseIf L = 39 Then
cr = "AO"
ElseIf L = 40 Then
cr = "AP"
ElseIf L = 41 Then
cr = "AQ"
ElseIf L = 42 Then
cr = "AR"
ElseIf L = 43 Then
```

```
 cr = "AS"
 ElseIf L = 44 Then
 cr = "AT"
 ElseIf L = 45 Then
 cr = "AU"
 ElseIf L = 46 Then
 cr = "AV"
 ElseIf L = 47 Then
 cr = "AW"
 ElseIf L = 48 Then
 cr = "AX"
 ElseIf L = 49 Then
 cr = "AY"
 ElseIf L = 50 Then
 cr = "AZ"
End If
crf = cr
End Function
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
Function crf2(L)        'Function to index columns for surface chart of differential head.
Dim cr As String
If L = 1 Then
 cr = "BD"
 ElseIf L = 2 Then
 cr = "BE"
 ElseIf L = 3 Then
 cr = "BF"
 ElseIf L = 4 Then
 cr = "BG"
 ElseIf L = 5 Then
 cr = "BH"
 ElseIf L = 6 Then
 cr = "BI"
 ElseIf L = 7 Then
 cr = "BJ"
 ElseIf L = 8 Then
 cr = "BK"
 ElseIf L = 9 Then
 cr = "BL"
 ElseIf L = 10 Then
 cr = "BM"
 ElseIf L = 11 Then
 cr = "BN"
 ElseIf L = 12 Then
 cr = "BO"
 ElseIf L = 13 Then
 cr = "BP"
 ElseIf L = 14 Then
 cr = "BQ"
 ElseIf L = 15 Then
 cr = "BR"
 ElseIf L = 16 Then
 cr = "BS"
 ElseIf L = 17 Then
```

```
cr = "BT"
ElseIf L = 18 Then
cr = "BU"
ElseIf L = 19 Then
cr = "BV"
ElseIf L = 20 Then
cr = "BW"
ElseIf L = 21 Then
cr = "BX"
ElseIf L = 22 Then
cr = "BY"
ElseIf L = 23 Then
cr = "BZ"
ElseIf L = 24 Then
cr = "CA"
ElseIf L = 25 Then
cr = "CB"
ElseIf L = 26 Then
cr = "CC"
ElseIf L = 27 Then
cr = "CD"
ElseIf L = 28 Then
cr = "CE"
ElseIf L = 29 Then
cr = "CF"
ElseIf L = 30 Then
cr = "CG"
ElseIf L = 31 Then
cr = "CH"
ElseIf L = 32 Then
cr = "CI"
ElseIf L = 33 Then
cr = "CJ"
ElseIf L = 34 Then
cr = "CK"
ElseIf L = 35 Then
cr = "CL"
ElseIf L = 36 Then
cr = "CM"
ElseIf L = 37 Then
cr = "CN"
ElseIf L = 38 Then
cr = "CO"
ElseIf L = 39 Then
cr = "CP"
ElseIf L = 40 Then
cr = "CQ"
ElseIf L = 41 Then
cr = "CR"
ElseIf L = 42 Then
cr = "CS"
ElseIf L = 43 Then
cr = "CT"
ElseIf L = 44 Then
cr = "CU"
ElseIf L = 45 Then
```

```
 cr = "CV"
 ElseIf L = 46 Then
 cr = "CW"
 ElseIf L = 47 Then
 cr = "CX"
 ElseIf L = 48 Then
 cr = "CY"
 ElseIf L = 49 Then
 cr = "CZ"
 ElseIf L = 50 Then
 cr = "DA"
End If
crf2 = cr
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function khbave(AvTp, k1c, h1c, k2c, h2c, dx1c, dx2c)
Dim khba As Double
  If AvTp = "Simple Arithmetic" Then
  khba = (k1c * h1c + k2c * h2c) / 2
  ElseIf AvTp = "Weighted Arithmetic" Then
  khba = (k1c * h1c * dx1c + k2c * h2c * dx2c) / (dx1c + dx2c)       'Distance weighted average kh.
  Else                                                    'AvTp = "Weighted Harmonic" by default
    khba = (dx1c + dx2c) / ((dx1c / (k1c * h1c)) + (dx2c / (k2c * h2c)))
  End If
khbave = khba
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function khbar(SIT, AvTp, k1, h1, dx1, WR1, hx1, UMS1, Bed1, bpp1, AqB1, k2, h2, dx2, WR2, hx2, UMS2, Bed2, bpp2, AqB2, dyy, UMA1, UMA2, ATC, _
               AqT1, AqT2, Lon1, Lon2, CT1, CT2, side, SB) 'Function to calculate specified transmission property intercell average in aquifer.
Dim khb As Double, dxb As Double, dx1c As Double, dx2c As Double, k1c As Double, k2c As Double, h1c As Double, h2c As Double
Dim Incr1 As Double, Incr2 As Double, Thk1 As Double, Thk2 As Double, Bedby As Double
Dim Topper As Double, Topper2 As Double, Topper1 As Double, Topper1u As Double
dx1c = dx1
dx2c = dx2
k1c = k1
k2c = k2
h1c = h1
h2c = h2
If k1c * h1c * dx1c * k2c * h2c * dx2c = 0 Then
khb = 0
Else
dxb = (dx1c + dx2c) / 2
khb = khbave(AvTp, k1c, h1c, k2c, h2c, dx1c, dx2c)
khb = khb * dyy / dxb
If SB = "Permissive" Then
 If SIT = "Complete" Then
  If WR1 > 0 Then
   If CT2 <> 2 Then
          If Bed1 < AqB2 Then
            Bedby = AqB2
           ElseIf Bed1 > AqT2 Then
            Bedby = AqT2
```

```
       Else
        Bedby = Bed1
       End If
         If hx2 + UMA2 > AqT2 Then
           Topper2 = AqT2
          Else
           Topper2 = hx2 + UMA2
         End If
         If hx1 > AqT2 Then
           Topper1 = AqT2
          Else
           Topper1 = hx1
         End If
         If hx1 + UMA1 > AqT2 Then
           Topper1u = AqT2
          Else
           Topper1u = hx1 + UMA1
         End If
         If hx2 + UMA2 > hx1 Then
           Topper = Topper2
          ElseIf hx2 + UMA2 <= hx1 Then
           Topper = Topper1
         End If
If side <> Lon1 Then
  If hx2 + UMA2 > Bed1 Then
    h2c = hofunc(hx2, AqT2, AqB2, UMA2, ATC, 0, Bedby, 0, WR1)
    khb = khbave(AvTp, k1c, h1c, k2c, h2c, dx1c, dx2c)
    khb = khb * dyy / dxb
    khb = khb + k2c * (Topper - Bedby) * WR1 * dyy / (dx2c / 2)
   ElseIf hx2 + UMA2 <= Bed1 Then
    khb = khb + k2c * (Topper - Bedby) * WR1 * dyy / (dx2c / 2)
  End If
 ElseIf side = Lon1 Then
    h1c = WR1 * (Bedby - AqB1)
      If Bedby >= AqB2 Then
        h1c = h1c + (1 - WR1) * (Bedby - AqB1)
       Else
        If hx1 + UMA1 > AqB2 Then
          If AqB2 > AqB1 Then
            h1c = h1c + (1 - WR1) * (AqB2 - AqB1)
           Else
            h1c = h1c + (1 - WR1) * (Topper1u - AqB1)
          End If
         Else
          h1c = h1c + (1 - WR1) * (Topper1u - AqB1)
        End If
      End If
  If hx2 + UMA2 > Bed1 Then
    h2c = Bedby - AqB2
  End If
    If h2c > 0 Then
      If h1c > 0 Then
       khb = khbave(AvTp, k1c, h1c, k2c, h2c, dx1c, dx2c)
       khb = khb * dyy / dxb
       Else
        khb = 0
```

```
          End If
         Else
          khb = 0
        End If
         If WR1 = 1 Then
           khb = khb + k2c * (Topper - Bedby) * dyy / (dx2c / 2)
          Else
           If (Topper1 - Bedby) > 0 Then
            If (Topper2 - Bedby) > 0 Then
              khb = khb + khbave(AvTp, k1c, (Topper1 - Bedby), k2c, (Topper2 - Bedby), ((1 - WR1) * dx1c), dx2c) * dyy / ((((1 - WR1) * dx1c) + dx2c)
/ 2)
            End If
           End If
         End If
    End If
   End If
  End If
 End If
End If
End If
khbar = khb
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function khbarat(SIT, SB, AvTp, k1, h1, dx1, Lon1, WRAT1, Bed1, bpp1, AqT1, hxat1, UMS1, _
                 k2, h2, dx2, Lon2, WRAT2, Bed2, bpp2, AqT2, hxat2, UMS2, side, dyy, UMAT1, UMAT2, _
                 CT1, CT2, DEC1, DEC2, rpc, AqB1, AqB2, hx1, hx2, Hn1, Hn2, UMA2) 'Function to calculate specified transmission property intercell
average in aquitard.
Dim khb As Double, dx1c As Double, dx2c As Double, dxb As Double, k1c As Double, k2c As Double, h1c As Double, h2c As Double
Dim Incr1 As Double, Incr2 As Double, BedbyT As Double, TopperT As Double, Topper2T As Double, Topper1T As Double, Topper1uT As Double
dx1c = dx1
dx2c = dx2
k1c = k1
k2c = k2
h1c = h1
h2c = h2
If k1c * h1c * k2c * h2c = 0 Then
khb = 0
Else
dxb = (dx1c + dx2c) / 2
khb = khbave(AvTp, k1c, h1c, k2c, h2c, dx1c, dx2c)
khb = khb * dyy / dxb
If SB = "Permissive" Then
 If SIT = "Complete" Then
  If CT1 = 2 Then
   If CT2 <> 2 Then
          If Bed1 < AqT2 Then
            BedbyT = AqT2
           ElseIf Bed1 > DEC2 Then
            BedbyT = DEC2
           Else
            BedbyT = Bed1
          End If
            If hxat2 + UMAT2 > DEC2 Then
              Topper2T = DEC2
```

```
         Else
          Topper2T = hxat2 + UMAT2
        End If
          Topper1T = hxat1
        If hxat1 + UMAT1 > DEC1 Then
          Topper1uT = DEC1
         Else
          Topper1uT = hxat1 + UMAT1
        End If
        If hxat2 + UMAT2 > hxat1 Then
          TopperT = Topper2T
         ElseIf hxat2 + UMAT2 <= hxat1 Then
          TopperT = Topper1T
        End If
If side <> Lon1 Then
  If hxat2 + UMAT2 > Bed1 Then
    h2c = hoatfunc(rpc, hxat2, AqT2, AqB2, hx2, UMAT2, Hn2, UMA2, 0, BedbyT, 0, WRAT1)
    khb = khbave(AvTp, k1c, h1c, k2c, h2c, dx1c, dx2c)
    khb = khb * dyy / dxb
    khb = khb + k2c * (TopperT - BedbyT) * WRAT1 * dyy / (dx2c / 2)
   ElseIf hx2 + UMAT2 <= Bed1 Then
    khb = khb + k2c * (TopperT - BedbyT) * WRAT1 * dyy / (dx2c / 2)
  End If
 ElseIf side = Lon1 Then
    If Bed1 > AqT1 Then
       h1c = WRAT1 * (Bed1 - AqT1)
      Else
       h1c = 0
    End If
      If Bed1 >= AqT2 Then
        h1c = h1c + (1 - WRAT1) * (Bed1 - AqT1)
       Else
        If hxat1 + UMAT1 > AqT2 Then
          If AqT2 > AqT1 Then
            h1c = h1c + (1 - WRAT1) * (AqT2 - AqT1)
           Else
            h1c = h1c + (1 - WRAT1) * (Topper1uT - AqT1)
          End If
         Else
          h1c = h1c + (1 - WRAT1) * (Topper1uT - AqT1)
        End If
      End If
  If hxat2 + UMAT2 > Bed1 Then
    h2c = BedbyT - AqT2
  End If
    If h2c > 0 Then
      If h1c > 0 Then
        khb = khbave(AvTp, k1c, h1c, k2c, h2c, dx1c, dx2c)
        khb = khb * dyy / dxb
       Else
        khb = 0
      End If
     Else
      khb = 0
    End If
     If WRAT1 = 1 Then
```

```
          khb = khb + k2c * (TopperT - BedbyT) * dyy / (dx2c / 2)
         Else
          If (Topper1T - BedbyT) > 0 Then
           If (Topper2T - BedbyT) > 0 Then
            khb = khb + khbave(AvTp, k1c, (Topper1T - BedbyT), k2c, (Topper2T - BedbyT), ((1 - WRAT1) * dx1c), dx2c) * dyy / ((((1 - WRAT1) * dx1c)
+ dx2c) / 2)
           End If
          End If
        End If
    End If
   End If
  End If
 End If
End If
End If
khbarat = khb
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function hofunc(Hf, AqT, AqB, UMA, ATC, UMS, Bed, bpp, WR) 'Function to calculate applicable saturated thickness.
Dim hof As Double, Hff As Double
 Hff = Hf
 If ATC = "Unconfined" Then
  hof = (1 - WR) * (Hff - AqB + UMA)
  If Bed - bpp < Hff + UMS Then
    hof = hof + WR * (Bed - bpp - AqB)
   Else
    hof = hof + WR * (Hff - AqB + UMS)
  End If
 Else 'Confined.
  If Hff >= AqT - UMA Then
  hof = (1 - WR) * (AqT - AqB)
  Else
  hof = (1 - WR) * (Hff - AqB + UMA)
  End If
  If Bed - bpp < AqT Then
   If Bed - bpp < Hff + UMS Then
     hof = hof + WR * (Bed - bpp - AqB)
    Else
     hof = hof + WR * (Hff - AqB + UMS)
   End If
   Else 'bed-bpp > AqT.
   If Bed - bpp < Hff + UMS Then
     hof = hof + WR * (AqT - AqB)
    Else
     If AqT < Hff + UMS Then
       hof = hof + WR * (AqT - AqB)
      Else
       hof = hof + WR * (Hff - AqB + UMS)
    End If
   End If
  End If
 End If
hofunc = hof
End Function
```

```
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function hoatfunc(rpc, hxat, AqT, AqB, hx, UMAT, Hn, UMA, UMS, Bed, bpp, WRAT) 'Function to calculate applicable saturated thickness.
Dim hoatf As Double, hxs As Double, Htf As Double, hs As Double
 Htf = hxat
 If Htf <= AqT - UMA Then
   If hx > AqT - UMA Then
    Htf = hx
   End If
 End If
     If Htf > AqT - UMA Then
        hoatf = (1 - WRAT) * (Htf + UMAT - AqT)
        If WRAT > 0 Then
        If Bed - bpp > AqT Then
         If Bed - bpp < Htf + UMS Then
           hoatf = hoatf + WRAT * (Bed - bpp - AqT)
          Else
           hoatf = hoatf + WRAT * (Htf + UMS - AqT)
         End If
        End If
        End If
       Else
        hoatf = 0
     End If
 hoatfunc = hoatf
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function Sfunc(CT, SIT, ATC, WR, H, AqT, Bed, Ss, Sy, AqB, bpp, SB, UMA, UMS) 'Function to calculate applicable storage coefficient when no
transition.
Dim sf As Double, Incr As Double
If CT = 4 Then
sf = 1                                  '1 affords stability without affecting result.
Else
If CT = 2 Then
If SIT <> "None" Then
      If UMS > UMA Then
         Incr = UMS - UMA
      End If
 If ATC = "Unconfined" Then
  If H >= Bed - bpp - Incr Then
   sf = (1 - WR) * Sy + WR * Ss * (Bed - bpp - AqB)
  Else
   sf = Sy
  End If
 ElseIf ATC = "Confined" Then
  If Bed - bpp <= AqT Then
   If H >= AqT Then
   sf = (1 - WR) * Ss * (AqT - AqB) + WR * Ss * (Bed - bpp - AqB)
   ElseIf H >= Bed - bpp - Incr Then
   sf = (1 - WR) * Sy + WR * Ss * (Bed - bpp - AqB)
   Else
   sf = Sy
   End If
```

```
  ElseIf Bed - bpp > AqT Then
   If H >= AqT Then
   sf = Ss * (AqT - AqB)
   ElseIf H >= Bed - bpp - Incr Then
   sf = (1 - WR) * Sy + WR * Ss * (AqT - AqB)
   Else
   sf = Sy
   End If
  End If
 End If
ElseIf SIT = "None" Then
If ATC = "Unconfined" Then
   sf = Sy
 ElseIf ATC = "Confined" Then
  If Bed <= AqT Then
   If H >= AqT Then
   sf = Ss * (AqT - AqB)
   Else
   sf = Sy
   End If
  ElseIf Bed > AqT Then
   If H >= AqT Then
   sf = Ss * (AqT - AqB)
   Else
   sf = Sy
   End If
  End If
End If
End If
ElseIf CT <> 2 Then
 If ATC = "Unconfined" Then
  sf = Sy
 Else
  If H >= AqT Then
  sf = Ss * (AqT - AqB)
  Else
  sf = Sy
  End If
 End If
End If
End If
Sfunc = sf
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function SfuncAT(CT, SIT, WRAT, hx, AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) 'Function to calculate weighted storage coefficient for aquitard when
no transition.
Dim sfat As Double, Incr As Double
If CT = 4 Then
sfat = 1                                  '1 affords stability without affecting result.
ElseIf CT = 2 Then
  If SIT <> "None" Then
  If UMS > UMAT Then
   Incr = UMS - UMAT
  End If
```

```
  If Bed - bpp <= AqT Then
   sfat = (1 - WRAT) * Sigma
  ElseIf Bed - bpp > AqT Then
   If hx > Bed - bpp - Incr Then    '''=
    sfat = (1 - WRAT) * Sigma + WRAT * Sigs * (Bed - bpp - AqT)
   Else
    sfat = Sigma
   End If
  End If
  Else
  sfat = Sigma
  End If
ElseIf CT <> 2 Then
  sfat = Sigma
End If
SfuncAT = sfat
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function SfuncNB(CT, ATC, H, AqT, AqB, Ss, Sy, UMA) 'Function to calculate applicable storage coefficient not beneath streambed when no transition.
Dim sf As Double
 If ATC = "Unconfined" Then
   sf = Sy
  Else
   If H >= AqT Then
    sf = Ss * (AqT - AqB)
    Else
    sf = Sy
   End If
 End If
SfuncNB = sf
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function SfuncB(SIT, ATC, H, AqT, AqB, Bed, Ss, Sy, bpp, SB, UMS, UMA, tttbx, Thick) 'Function to calculate applicable storage coefficient beneath
stream bed when no transition.
Dim sf As Double, Incr As Double
If UMS > UMA Then
  Incr = UMS - UMA
End If
If SIT <> "None" Then
 If ATC = "Unconfined" Then
  If H + Incr >= tttbx Then
   sf = Ss * Thick
  Else
   sf = Sy
  End If
 ElseIf ATC = "Confined" Then
   If H + Incr >= tttbx Then
     sf = Ss * Thick
    Else
     sf = Sy
   End If
  End If
```

```
ElseIf SIT = "None" Then
If ATC = "Unconfined" Then
   sf = Sy
 ElseIf ATC = "Confined" Then
   If H + Incr > AqT Then
   sf = Ss * (AqT - AqB)
   Else
   sf = Sy
   End If
  End If
End If
SfuncB = sf
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function SfuncATB(SIT, hx, AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) 'Function to calculate storage coefficient beneath bed for aquitard when no
transition.
Dim sfatb As Double, Incr As Double
If UMS > UMAT Then
 Incr = UMS - UMAT
End If
If SIT <> "None" Then
  If Bed - bpp <= AqT Then
   sfatb = 0
  ElseIf Bed - bpp > AqT Then
   If hx > Bed - bpp - Incr Then
    sfatb = Sigs * (Bed - bpp - AqT)
   Else
    sfatb = Sigma
   End If
  End If
ElseIf SIT = "None" Then
   sfatb = Sigma
End If
SfuncATB = sfatb
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
  Function FuncE(AA, BB, CC, DD, Trans, TransB, CT, dx, dy, dt, WR, ATC, AqT, AqB, Ss, Sy, SIT, Bed, bpp, hx, SB, Hn, UMA, UMS)
  Dim EEx As Double, Thick As Double, tttbx As Double
  EEx = AA + BB + CC + DD                                                              'Defined to facilitate coding.
    If Trans + TransB = 0 Then
     EEx = EEx + Sfunc(CT, SIT, ATC, WR, (hx + UMA), AqT, Bed, Ss, Sy, AqB, bpp, SB, UMA, UMS) * dx * dy / dt
      Else
      If WR < 1 Then
      If Trans = 0 Then
       EEx = EEx + (1 - WR) * SfuncNB(CT, ATC, (hx + UMA), AqT, AqB, Ss, Sy, UMA) * dx * dy / dt
       ElseIf Trans = 1 Then             'Depressurization.
       EEx = EEx + (1 - WR) * (Sy) * dx * dy / dt
       ElseIf Trans = 2 Then             'Pressurization.
       EEx = EEx + (1 - WR) * (Ss * (AqT - AqB)) * dx * dy / dt
      End If
      End If
      If WR > 0 Then
```

```
       tttbx = TTB(SIT, ATC, AqT, Bed, bpp, UMS, UMA)
       Thick = fThick(SIT, ATC, Bed, bpp, AqT, AqB)
      If TransB = 0 Then
        EEx = EEx + WR * SfuncB(SIT, ATC, (Hn + UMA), AqT, AqB, Bed, Ss, Sy, bpp, SB, UMS, UMA, tttbx, Thick) * dx * dy / dt
      ElseIf TransB = 1 Then              'Bed Depressurization.
          EEx = EEx + WR * (Sy) * dx * dy / dt
      ElseIf TransB = 2 Then              'Bed Pressurization
        Thick = fThick(SIT, ATC, Bed, bpp, AqT, AqB)
        EEx = EEx + WR * Ss * Thick * dx * dy / dt
      End If
      End If
    End If
    FuncE = EEx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
  Function FuncET(AAt, BBt, CCt, DDt, Trans, TransB, CT, dx, dy, dt, WRAT, ATC, AqT, Sigs, Sigma, SIT, Bed, bpp, hxat, SB, Hnatt, UMS, UMAT) 'For
aquitard cells.
  Dim ETx As Double, ThickT As Double
  ETx = AAt + BBt + CCt + DDt                                                              'Defined to facilitate coding.
    If Trans + TransB = 0 Then
     ETx = ETx + SfuncAT(CT, SIT, WRAT, (hxat + UMAT), AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) * dx * dy / dt
      Else
      If WRAT < 1 Then
      If Trans = 0 Then
       ETx = ETx + (1 - WRAT) * Sigma * dx * dy / dt
       ElseIf Trans = 1 Then             'Depressurization.
       ETx = ETx + (1 - WRAT) * Sigma * dx * dy / dt
       ElseIf Trans = 2 Then            'Pressurization.
       ETx = ETx + (1 - WRAT) * Sigma * dx * dy / dt
      End If
      End If
      If WRAT > 0 Then
       If Bed - bpp > AqT Then
         ThickT = Bed - bpp - AqT
         If TransB = 0 Then
          ETx = ETx + WRAT * SfuncATB(SIT, (Hnatt + UMAT), AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) * dx * dy / dt
         ElseIf TransB = 1 Then               'Bed Depressurization.
           ETx = ETx + WRAT * Sigma * dx * dy / dt
         ElseIf TransB = 2 Then               'Just rewet.
           ETx = ETx + WRAT * Sigma * dx * dy / dt
         ElseIf TransB = 3 Then               'Depressurize and dewater.
           ETx = ETx + WRAT * Sigma * dx * dy / dt
         ElseIf TransB = 4 Then               'Rewet and pressurize.
           ETx = ETx + WRAT * Sigs * ThickT * dx * dy / dt
         ElseIf TransB = 5 Then               'Just dewater.
           ETx = ETx + WRAT * Sigma * dx * dy / dt
         ElseIf TransB = 6 Then               'Bed Pressurization.
           ETx = ETx + WRAT * Sigs * ThickT * dx * dy / dt
         End If
       ElseIf Bed - bpp <= AqT Then
       End If
      End If
    End If
    FuncET = ETx
```

```
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function FuncB(Qc, Trans, TransB, CT, dx, dy, dt, WR, ATC, Hn, AqT, AqB, Ss, Sy, SIT, Bed, bpp, SB, UMA, UMS)
  Dim bx As Double, tttbx As Double, Thick As Double
  bx = Qc
    If Trans + TransB = 0 Then
      bx = bx - Hn * Sfunc(CT, SIT, ATC, WR, (Hn + UMA), AqT, Bed, Ss, Sy, AqB, bpp, SB, UMA, UMS) * dx * dy / dt 'Terms from right side of equation.
     Else
      If WR < 1 Then
      If Trans = 0 Then
        bx = bx - (1 - WR) * Hn * SfuncNB(CT, ATC, (Hn + UMA), AqT, AqB, Ss, Sy, UMA) * dx * dy / dt
       ElseIf Trans = 1 Then              'Depressurization.
        bx = bx + (1 - WR) * ((AqT) * (Ss * (AqT - AqB) - Sy) + Sy * UMA - (Hn + UMA) * Ss * (AqT - AqB)) * dx * dy / dt
       ElseIf Trans = 2 Then              'Pressurization.
        bx = bx + (1 - WR) * ((AqT) * (Sy - Ss * (AqT - AqB)) + Ss * (AqT - AqB) * UMA - (Hn + UMA) * Sy) * dx * dy / dt
      End If
      End If
      If WR > 0 Then
       tttbx = TTB(SIT, ATC, AqT, Bed, bpp, UMS, UMA)
       Thick = fThick(SIT, ATC, Bed, bpp, AqT, AqB)
      If TransB = 0 Then
        bx = bx - WR * Hn * SfuncB(SIT, ATC, (Hn + UMA), AqT, AqB, Bed, Ss, Sy, bpp, SB, UMS, UMA, tttbx, Thick) * dx * dy / dt
       ElseIf TransB = 1 Then             'Bed Depressurization.
        bx = bx + WR * (tttbx * (Ss * Thick - Sy) + Sy * UMS - (Hn + UMS) * Ss * Thick) * dx * dy / dt
       ElseIf TransB = 2 Then             'Bed Pressurization
       bx = bx + WR * (tttbx * (Sy - (Ss * Thick)) + Ss * Thick * UMS - (Hn + UMS) * Sy) * dx * dy / dt
      End If
      End If
    End If
    FuncB = bx
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function FuncBB(Qc, Trans, TransB, CT, dx, dy, dt, WR, ATC, Hn, AqT, AqB, Ss, Sy, SIT, Bed, bpp, SB, hx, ITS, UMA, UMS) 'For cells beneath aquitard.
  Dim bx As Double, tttbx As Double, Thick As Double
  If ITS = "Deep Percolation" Then
   Else
   bx = Qc
  End If
    If Trans + TransB = 0 Then
      bx = bx - Hn * Sfunc(CT, SIT, ATC, WR, (Hn + UMA), AqT, Bed, Ss, Sy, AqB, bpp, SB, UMA, UMS) * dx * dy / dt 'Terms from right side of equation.
     Else
      If WR < 1 Then
      If Trans = 0 Then
        bx = bx - (1 - WR) * Hn * SfuncNB(CT, ATC, (Hn + UMA), AqT, AqB, Ss, Sy, UMA) * dx * dy / dt
       ElseIf Trans = 1 Then              'Depressurization.
        bx = bx + (1 - WR) * ((AqT) * (Ss * (AqT - AqB) - Sy) + Sy * UMA - (Hn + UMA) * Ss * (AqT - AqB)) * dx * dy / dt
       ElseIf Trans = 2 Then              'Pressurization.
        bx = bx + (1 - WR) * ((AqT) * (Sy - Ss * (AqT - AqB)) + Ss * (AqT - AqB) * UMA - (Hn + UMA) * Sy) * dx * dy / dt
      End If
      End If
      If WR > 0 Then
       tttbx = TTB(SIT, ATC, AqT, Bed, bpp, UMS, UMA)
```

```
       Thick = fThick(SIT, ATC, Bed, bpp, AqT, AqB)
      If TransB = 0 Then
        bx = bx - WR * Hn * SfuncB(SIT, ATC, (Hn + UMA), AqT, AqB, Bed, Ss, Sy, bpp, SB, UMS, UMA, tttbx, Thick) * dx * dy / dt
       ElseIf TransB = 1 Then            'Bed Depressurization.
        bx = bx + WR * (tttbx * (Ss * Thick - Sy) + Sy * UMS - (Hn + UMS) * Ss * Thick) * dx * dy / dt
       ElseIf TransB = 2 Then          'Bed Pressurization
       bx = bx + WR * (tttbx * (Sy - (Ss * Thick)) + Ss * Thick * UMS - (Hn + UMS) * Sy) * dx * dy / dt
      End If
      End If
    End If
    FuncBB = bx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function FuncBT(Qc, Trans, TransB, CT, dx, dy, dt, WRAT, ATC, Hnat, AqT, AqB, Sigs, Sigma, SIT, Bed, bpp, SB, hx, ITS, UMAT, UMS, hoat, UMA) 'For
Aquitard Cells
  Dim bt As Double, ThickT As Double
  If ITS = "Deep Percolation" Then
   bt = Qc
   If bt > 0 Then
    If hoat = 0 Then
     bt = 9.99901110999
     GoTo LineBtX
    End If
   End If
   Else
  End If
     If Trans + TransB = 0 Then
      bt = bt - Hnat * SfuncAT(CT, SIT, WRAT, (Hnat + UMAT), AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) * dx * dy / dt 'Terms from right side of
Equation.
     Else
      If WRAT < 1 Then
      If Trans = 0 Then
        bt = bt - (1 - WRAT) * Hnat * Sigma * dx * dy / dt
       ElseIf Trans = 1 Then             'Depressurization.
        bt = bt - (1 - WRAT) * Hnat * Sigma * dx * dy / dt
       ElseIf Trans = 2 Then             'Pressurization.
        bt = bt - (1 - WRAT) * Hnat * Sigma * dx * dy / dt
      End If
      End If
      If WRAT > 0 Then
       If Bed - bpp > AqT Then
          ThickT = Bed - bpp - AqT
         If TransB = 0 Then
           bt = bt - WRAT * Hnat * SfuncATB(SIT, Hnat + UMAT, AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) * dx * dy / dt
         ElseIf TransB = 1 Then                  'Bed Depressurization.
           bt = bt + WRAT * ((Bed - bpp) * (Sigs * ThickT - Sigma) + Sigma * UMS - (Hnat + UMS) * Sigs * ThickT) * dx * dy / dt
         ElseIf TransB = 2 Then                  'Just rewet.
           bt = bt - WRAT * Hnat * Sigma * dx * dy / dt
         ElseIf TransB = 3 Then                  'Depressurize and dewater.
           bt = bt + WRAT * ((Bed - bpp) * (Sigs * ThickT - Sigma) + Sigma * UMS - (Hnat + UMS) * Sigs * ThickT) * dx * dy / dt
         ElseIf TransB = 4 Then                  'Rewet and pressurize.
           bt = bt + WRAT * ((Bed - bpp) * (Sigma - Sigs * ThickT) + Sigs * ThickT * UMS - (Hnat + UMS) * Sigma) * dx * dy / dt
         ElseIf TransB = 5 Then                  'Just dewater.
           bt = bt - WRAT * Hnat * Sigma * dx * dy / dt
```

```
        ElseIf TransB = 6 Then                'Bed Pressurization.
          bt = bt + WRAT * ((Bed - bpp) * (Sigma - Sigs * ThickT) + Sigs * ThickT * UMS - (Hnat + UMS) * Sigma) * dx * dy / dt
        End If
       ElseIf Bed - bpp <= AqT Then
       End If
      End If
    End If
LineBtX:
    FuncBT = bt
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function khbarC(AvTp, k1, h1, dx1, k2, h2, dx2, dyy) 'Function to calculate specified transmission property average.
Dim khb As Double, dxb As Double
If k1 * h1 * k2 * h2 = 0 Then
khb = 0
Else
dxb = (dx1 + dx2) / 2
If AvTp = "Simple Arithmetic" Then
khb = (k1 * h1 + k2 * h2) / 2
ElseIf AvTp = "Weighted Arithmetic" Then
khb = (k1 * h1 * dx1 + k2 * h2 * dx2) / (2 * dxb)        'Distance weighted average kh.
Else                                                     'AvTp = "Weighted Harmonic" by default
  khb = (2 * dxb) / ((dx1 / (k1 * h1)) + (dx2 / (k2 * h2))) 'Harmonic.
End If
khb = khb * dyy / dxb
End If
khbarC = khb
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function hofuncC(H, AqT, AqB, UMA, UMS, ATC, CT, widp, dx, dy, widpp, lenp, Bed, bpp, bppb, _
               ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, SIT, SB) 'Function to calculate applicable saturated
thickness, for horizontal flow.
Dim BEDpp As Double, Vt As Double, Vm As Double, Vn As Double, hof As Double, Beds1 As Double, Beds2 As Double, hs As Double, hss As Double
Dim V1 As Double, V2 As Double, V3 As Double, V4 As Double, V5 As Double, V6 As Double, V7 As Double, V8 As Double
Dim Beds3 As Double, Beds4 As Double, Hf As Double
 Hf = H
If CT = 2 Then
   If ATC = "Unconfined" Then
     BEDpp = Bed - bpp
     hs = Hf + UMA
    Else 'Confined.
      If Bed - bpp <= AqT Then
        BEDpp = Bed - bpp
       Else
        BEDpp = AqT
      End If
          If Hf + UMA <= AqT Then
             hs = Hf + UMA
           Else
             hs = AqT
          End If
    End If 'ATC.
```

```
 Vt = dx * dy * (hs - AqB)
 If ATC = "Unconfined" Then
          If BEDpp > hs Then
           If Hf + UMS > Bed - bpp Then
     Vt = Vt + widpp * lenp * (BEDpp - hs)
            Else
     Vt = Vt + widpp * lenp * (Hf + UMS - hs)
           End If
          End If
  Else 'Confined.
         If hs < AqT Then
          If BEDpp > hs Then
           If Hf + UMS > Bed - bpp Then
     Vt = Vt + widpp * lenp * (BEDpp - hs)
            Else
     Vt = Vt + widpp * lenp * (Hf + UMS - hs)
           End If
          End If
        End If
 End If 'ATC.
      If BEDpp < hs Then
        Vm = widpp * lenp * (hs - BEDpp)
      End If
      If widpp + 2 * bppb <= widp Then
        If Bed < hs Then
          Vm = Vm + 2 * bppb * lenp * (hs - Bed)
        End If
       ElseIf widpp + 2 * bppb > widp Then
        If Bed < hs Then
          Vm = Vm + (widp - widpp) * lenp * (hs - Bed)
        End If
      End If
ElseIf CT <> 2 Then
 If ATC = "Unconfined" Then
    hs = Hf + UMA
 Else 'Confined.
  If Hf >= AqT - UMA Then
    hs = AqT
   Else 'Hf.
    hs = Hf + UMA
  End If
 End If 'ATC.
 Vt = dx * dy * (hs - AqB)
 If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
  If Bed1 > AqT Then
    Beds1 = AqT
   Else
    Beds1 = Bed1
  End If
  If Bed2 > AqT Then
   Beds2 = AqT
   Else
    Beds2 = Bed2
  End If
  If Bed3 > AqT Then
   Beds3 = AqT
```

```
    Else
     Beds3 = Bed3
   End If
   If Bed4 > AqT Then
    Beds4 = AqT
    Else
     Beds4 = Bed4
   End If
    Vm = Vminus(ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy, hs)
   Else
    Vm = 0
  End If
End If 'CT.
Vn = Vt - Vm
hof = Vn / (dx * dy)
hofuncC = hof
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function hobfunC(H, AqT, AqB, UMA, UMS, ATC, CT, widp, dx, dy, widpp, lenp, Bed, bpp, bppb, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, SIT, SB, side4) 'Function to calculate applicable saturated
thickness, for horizontal flow, from a particular side to the center of the cell.
Dim BEDpp As Double, Vt As Double, Vm As Double, Vn As Double, hof As Double, Beds1 As Double, Beds2 As Double, hs As Double, hss As Double
Dim V1 As Double, V2 As Double, V3 As Double, V4 As Double, V5 As Double, V6 As Double, V7 As Double, V8 As Double
Dim Beds3 As Double, Beds4 As Double, Hf As Double
 Hf = H
If CT = 2 Then
   If ATC = "Unconfined" Then
     BEDpp = Bed - bpp
     hs = Hf + UMA
    Else 'Confined.
      If Bed - bpp <= AqT Then
        BEDpp = Bed - bpp
       Else
        BEDpp = AqT
      End If
          If Hf + UMA <= AqT Then
             hs = Hf + UMA
           Else
             hs = AqT
          End If
    End If 'ATC.
  Vt = dx * dy * (hs - AqB)
  If ATC = "Unconfined" Then
           If BEDpp > hs Then
            If Hf + UMS > Bed - bpp Then
      Vt = Vt + widpp * lenp * (BEDpp - hs)
             Else
      Vt = Vt + widpp * lenp * (Hf + UMS - hs)
            End If
           End If
   Else 'Confined.
          If hs < AqT Then
           If BEDpp > hs Then
            If Hf + UMS > Bed - bpp Then
```

```
      Vt = Vt + widpp * lenp * (BEDpp - hs)
             Else
      Vt = Vt + widpp * lenp * (Hf + UMS - hs)
            End If
           End If
          End If
  End If 'ATC.
       If BEDpp < hs Then
         Vm = widpp * lenp * (hs - BEDpp)
       End If
       If widpp + 2 * bppb <= widp Then
         If Bed < hs Then
           Vm = Vm + 2 * bppb * lenp * (hs - Bed)
         End If
        ElseIf widpp + 2 * bppb > widp Then
         If Bed < hs Then
           Vm = Vm + (widp - widpp) * lenp * (hs - Bed)
         End If
       End If
 ElseIf CT <> 2 Then
  If ATC = "Unconfined" Then
     hs = Hf + UMA
  Else 'Confined.
   If Hf >= AqT - UMA Then
     hs = AqT
    Else 'Hf.
     hs = Hf + UMA
   End If
  End If 'ATC.
  Vt = dx * dy * (hs - AqB)
  If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
   If Bed1 > AqT Then
     Beds1 = AqT
    Else
     Beds1 = Bed1
   End If
   If Bed2 > AqT Then
    Beds2 = AqT
    Else
     Beds2 = Bed2
   End If
   If Bed3 > AqT Then
    Beds3 = AqT
    Else
     Beds3 = Bed3
   End If
   If Bed4 > AqT Then
    Beds4 = AqT
    Else
     Beds4 = Bed4
   End If
    Vm = 2 * VminusB(ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy, hs, side4)
   Else
    Vm = 0
  End If
End If 'CT.
```

```
Vn = Vt - Vm
hof = Vn / (dx * dy)
hobfunC = hof
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function hobatfunC(rpc, hxat, hx, Hn, AqT, AqB, UMAT, UMS, ATC, CT, widp, dx, dy, widpp, lenp, Bed, bpp, bppb, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, SIT, SB, side4, UMA) 'Function to calculate applicable
aquitard saturated thickness, for horizontal flow, from a particular side to the center of the cell.
Dim BEDpp As Double, Vt As Double, Vm As Double, Vn As Double, hof As Double, bed1t As Double, _
    bed2t As Double, bed3t As Double, bed4t As Double, hs As Double, Hf As Double, hxs As Double, _
    bedd As Double
  bedd = Bed
 Hf = hxat
 If Hf <= AqT - UMA Then
   If hx > AqT - UMA Then
    Hf = hx
   End If
 End If
If Hf > AqT - UMA Then
hs = Hf + UMAT
If CT = 2 Then
     BEDpp = bedd - bpp
  If BEDpp < AqT Then
   BEDpp = AqT
  End If
  If bedd < AqT Then
   bedd = AqT
  End If
  Vt = dx * dy * (hs - (AqT))
           If BEDpp > hs Then
            If Hf + UMS > bedd - bpp Then
              Vt = Vt + widpp * lenp * (BEDpp - hs)
               Else
              Vt = Vt + widpp * lenp * (Hf + UMS - hs)
            End If
           End If
       If BEDpp < hs Then
         Vm = widpp * lenp * (hs - BEDpp)
       End If
       If widpp + 2 * bppb <= widp Then
         If bedd < hs Then
           Vm = Vm + 2 * bppb * lenp * (hs - bedd)
         End If
        ElseIf widpp + 2 * bppb > widp Then
         If bedd < hs Then
           Vm = Vm + (widp - widpp) * lenp * (hs - bedd)
         End If
       End If
 ElseIf CT <> 2 Then
  Vt = dx * dy * (hs - (AqT))
  If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
   If Bed1 < AqT Then
     bed1t = AqT
    Else
```

```
     bed1t = Bed1
   End If
   If Bed2 < AqT Then
    bed2t = AqT
    Else
     bed2t = Bed2
   End If
   If Bed3 < AqT Then
    bed3t = AqT
    Else
     bed3t = Bed3
   End If
   If Bed4 < AqT Then
    bed4t = AqT
    Else
     bed4t = Bed4
   End If
    Vm = 2 * VminusB(ovp1, ovw1, bed1t, ovp2, ovw2, bed2t, ovp3, ovw3, bed3t, ovp4, ovw4, bed4t, dx, dy, hs, side4)
   Else
    Vm = 0
  End If
End If 'CT.
Vn = Vt - Vm
hof = Vn / (dx * dy)
End If
hobatfunC = hof
End Function
'*****************************************************************************************************************************************

'*****************************************************************************************************************************************
Function hoatfuncC(rpc, hxat, hx, Hn, AqT, AqB, UMAT, UMS, ATC, CT, widp, dx, dy, widpp, lenp, Bed, bpp, bppb, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, SIT, SB, UMA) 'Function to calculate applicable saturated
aquitard thickness.
Dim BEDpp As Double, Vt As Double, Vm As Double, Vn As Double, hof As Double, bed1t As Double, _
    bed2t As Double, bed3t As Double, bed4t As Double, hs As Double, Hf As Double, hxs As Double, bedd As Double
  bedd = Bed
 Hf = hxat
 If Hf <= AqT - UMA Then
   If hx > AqT - UMA Then
    Hf = hx
   End If
 End If
If Hf > AqT - UMA Then
hs = Hf + UMAT
If CT = 2 Then
     BEDpp = bedd - bpp
  If BEDpp < AqT Then
   BEDpp = AqT
  End If
  If bedd < AqT Then
   bedd = AqT
  End If
  Vt = dx * dy * (hs - (AqT))
           If BEDpp > hs Then
            If Hf + UMS > bedd - bpp Then
              Vt = Vt + widpp * lenp * (BEDpp - hs)
```

```
              Else
              Vt = Vt + widpp * lenp * (Hf + UMS - hs)
            End If
           End If
       If BEDpp < hs Then
         Vm = widpp * lenp * (hs - BEDpp)
       End If
       If widpp + 2 * bppb <= widp Then
         If bedd < hs Then
           Vm = Vm + 2 * bppb * lenp * (hs - bedd)
         End If
        ElseIf widpp + 2 * bppb > widp Then
         If bedd < hs Then
           Vm = Vm + (widp - widpp) * lenp * (hs - bedd)
         End If
       End If
 ElseIf CT <> 2 Then
  Vt = dx * dy * (hs - (AqT))
  If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
   If Bed1 < AqT Then
     bed1t = AqT
    Else
     bed1t = Bed1
   End If
   If Bed2 < AqT Then
    bed2t = AqT
    Else
     bed2t = Bed2
   End If
   If Bed3 < AqT Then
    bed3t = AqT
    Else
     bed3t = Bed3
   End If
   If Bed4 < AqT Then
    bed4t = AqT
    Else
     bed4t = Bed4
   End If
    Vm = Vminus(ovp1, ovw1, bed1t, ovp2, ovw2, bed2t, ovp3, ovw3, bed3t, ovp4, ovw4, bed4t, dx, dy, hs)
   Else
    Vm = 0
  End If
End If 'CT.
Vn = Vt - Vm
hof = Vn / (dx * dy)
End If
hoatfuncC = hof
End Function
'*****************************************************************************************************************************************

'*****************************************************************************************************************************************
Function SfuncC(CT, SIT, ATC, H, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
              ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy) 'Function to calculate applicable storage coefficient
when no transition.
Dim sf As Double, hs As Double, Sa As Double
```

```
Dim ACSs As Double, ACn As Double, AtS As Double, BEDpp As Double, bedd As Double, hss As Double
Dim Incr As Double
bedd = Bed
If CT = 4 Then
sf = 1                                  '1 affords stability without affecting result.
Else
If CT = 2 Then
  If UMS > UMA Then
   Incr = UMS - UMA
  End If
   If ATC = "Unconfined" Then
     BEDpp = Bed - bpp
     hs = H
     Sa = Sy
    Else 'Confined.
      If Bed - bpp <= AqT Then
        BEDpp = Bed - bpp
       Else
        BEDpp = AqT
      End If
          If H < AqT Then
             hs = H
             Sa = Sy
           Else
             hs = AqT
             Sa = Ss * (AqT - AqB)
          End If
      If bedd > AqT Then
        bedd = AqT
      End If
    End If 'ATC.
        If BEDpp > hs Then
           If H >= Bed - bpp - Incr Then
               ACSs = widpp * lenp * (BEDpp - AqB) * Ss
               ACn = widpp * lenp
           End If
         ElseIf BEDpp <= hs Then
           ACSs = widpp * lenp * (BEDpp - AqB) * Ss
           ACn = widpp * lenp
        End If
        If widpp + 2 * bppb <= widp Then
         If bedd <= hs Then
             ACSs = ACSs + 2 * bppb * lenp * (bedd - AqB) * Ss
             ACn = ACn + 2 * bppb * lenp
           ElseIf bedd > hs Then
             If H > Bed Then
               ACSs = ACSs + 2 * bppb * lenp * (bedd - AqB) * Ss
               ACn = ACn + 2 * bppb * lenp
             End If
         End If
        ElseIf widpp + 2 * bppb > widp Then
         If bedd <= hs Then
            ACSs = ACSs + (widp - widpp) * lenp * (bedd - AqB) * Ss
            ACn = ACn + (widp - widpp) * lenp
           ElseIf bedd > hs Then
            If H > Bed Then
```

```
            ACSs = ACSs + (widp - widpp) * lenp * (bedd - AqB) * Ss
            ACn = ACn + (widp - widpp) * lenp
           End If
         End If
       End If
ElseIf CT <> 2 Then
  If ATC = "Unconfined" Then
     hs = H
     Sa = Sy
  Else 'Confined.
   If H >= AqT Then
     hs = AqT
     Sa = Ss * (AqT - AqB)
    Else 'H.
     hs = H
     Sa = Sy
   End If
  End If 'ATC.
  If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
    ACSs = AConfSs(ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy, hs, AqB, Ss)
    ACn = AConf(ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy, hs, AqB)
  Else
    ACSs = 0
    ACn = 0
  End If
End If 'CT 2.
If hs > AqB Then
 AtS = ACSs + (dx * dy - ACn) * Sa
 sf = AtS / (dx * dy)
End If
End If 'CT 4.
SfuncC = sf
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function SfuncCt(CT, SIT, ATC, Hnat, AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
               ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy) 'Function to calculate applicable aquitard storage
coefficient when no transition.
Dim sf As Double, Hats As Double, Sa As Double, bedd As Double
Dim ACSs As Double, ACn As Double, AtS As Double, BEDpp As Double
Dim Bedd1 As Double, Bedd2 As Double, Bedd3 As Double, Bedd4 As Double, Aminus As Double, hss As Double
Dim Incr As Double
bedd = Bed
If CT = 4 Then
sf = 1                                  '1 affords stability without affecting result.
Else
If CT = 2 Then
      If UMS > UMAT Then
         Incr = UMS - UMAT
      End If
      If Bed - bpp > AqT Then
        BEDpp = Bed - bpp
       Else
        BEDpp = AqT
      End If
```

```
        Hats = Hnat
        Sa = Sigma
      If bedd < AqT Then
        bedd = AqT
      End If
        If BEDpp > Hats Then
           If Hats > Bed - bpp - Incr Then
               If Bed - bpp > AqT Then
                 ACSs = widpp * lenp * (BEDpp - AqT) * Sigs
                 ACn = widpp * lenp
                 Aminus = 0
               End If
           End If
         ElseIf BEDpp <= Hats Then
           ACSs = widpp * lenp * (BEDpp - AqT) * Sigs
           If BEDpp <= AqT Then
             ACn = 0
             Aminus = widpp * lenp
            Else
             ACn = widpp * lenp
             Aminus = 0
           End If
        End If
        If widpp + 2 * bppb <= widp Then
         If bedd <= Hats Then
           ACSs = ACSs + 2 * bppb * lenp * (bedd - AqT) * Sigs
           If bedd <= AqT Then
             ACn = ACn + 0
             Aminus = Aminus + 2 * bppb * lenp
            Else
             ACn = ACn + 2 * bppb * lenp
             Aminus = Aminus + 0
           End If
         End If
        ElseIf widpp + 2 * bppb > widp Then
         If bedd <= Hats Then
           ACSs = ACSs + (widp - widpp) * lenp * (bedd - AqT) * Sigs
           If bedd <= AqT Then
             ACn = ACn + 0
             Aminus = Aminus + (widp - widpp) * lenp
            Else
             ACn = ACn + (widp - widpp) * lenp
             Aminus = Aminus + 0
           End If
         End If
       End If
ElseIf CT <> 2 Then
     Sa = Sigma
     Hats = Hnat
  If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
      Bedd1 = Bed1
      Bedd2 = Bed2
      Bedd3 = Bed3
      Bedd4 = Bed4
      If Bedd1 < AqT Then
        Bedd1 = AqT
```

```
      End If
      If Bedd2 < AqT Then
        Bedd2 = AqT
      End If
      If Bedd3 < AqT Then
        Bedd3 = AqT
      End If
      If Bedd4 < AqT Then
        Bedd4 = AqT
      End If
    ACSs = AConfSs(ovp1, ovw1, Bedd1, ovp2, ovw2, Bedd2, ovp3, ovw3, Bedd3, ovp4, ovw4, Bedd4, dx, dy, Hats, AqT, Sigs)
    ACn = AConf(ovp1, ovw1, Bedd1, ovp2, ovw2, Bedd2, ovp3, ovw3, Bedd3, ovp4, ovw4, Bedd4, dx, dy, Hats, AqT)
    Aminus = Aminusf(ovp1, ovw1, Bedd1, ovp2, ovw2, Bedd2, ovp3, ovw3, Bedd3, ovp4, ovw4, Bedd4, dx, dy, AqT)
  End If
End If 'CT 2.
  AtS = ACSs + (dx * dy - ACn - Aminus) * Sa
  sf = AtS / (dx * dy)
End If 'CT 4.
SfuncCt = sf
End Function
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
Function FuncEC(AA, BB, CC, DD, Trans, dx, dy, dt, Hn, hx, CT, SIT, ATC, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT)
Dim EEx As Double, Sef As Double, tens As Integer, ones As Integer, THRS(1 To 5) As Double
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
  EEx = AA + BB + CC + DD                                                              'Defined to facilitate coding.
    If Trans = 0 Then
      Sef = SfuncC(CT, SIT, ATC, (Hn + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
      EEx = EEx + Sef * dx * dy / dt
     ElseIf Trans > 200 Then 'Pressurization.
      ones = (Trans - 200) Mod 10
      Sef = SfuncC(CT, SIT, ATC, THRS(ones), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
      EEx = EEx + Sef * dx * dy / dt
     ElseIf Trans > 100 Then 'Depressurization.
      Sef = SfuncC(CT, SIT, ATC, (hx + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
      EEx = EEx + Sef * dx * dy / dt
    End If
    FuncEC = EEx
End Function
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
  Function FuncETc(AAt, BBt, CCt, DDt, Trans, dx, dy, dt, Hnat, hxat, CT, SIT, ATC, AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, AqB, SB,
UMAT, UMS, _
                ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, THRSt1, THRSt2, THRSt3, THRSt4, THRSt5, THRCNTt, DEC)
'For aquitard cells.
Dim EEx As Double, Sef As Double, tens As Integer, ones As Integer, THRSt(1 To 5) As Double
```

```
THRSt(1) = THRSt1
THRSt(2) = THRSt2
THRSt(3) = THRSt3
THRSt(4) = THRSt4
THRSt(5) = THRSt5
  EEx = AAt + BBt + CCt + DDt                                                    'Defined to facilitate coding.
    If Trans = 0 Then
      Sef = SfuncCt(CT, SIT, ATC, (Hnat + UMAT), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                  SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                  ovp4, ovw4, Beds4, dx, dy)
      EEx = EEx + Sef * dx * dy / dt
     ElseIf Trans > 200 Then 'Pressurization.
      ones = (Trans - 200) Mod 10
      Sef = SfuncCt(CT, SIT, ATC, THRSt(ones), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                  SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                  ovp4, ovw4, Beds4, dx, dy)
      EEx = EEx + Sef * dx * dy / dt
     ElseIf Trans > 100 Then 'Depressurization.
      Sef = SfuncCt(CT, SIT, ATC, (hxat + UMAT), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                  SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                  ovp4, ovw4, Beds4, dx, dy)
      EEx = EEx + Sef * dx * dy / dt
    End If
    FuncETc = EEx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function FuncBC(Qc, Trans, dx, dy, dt, Hn, hx, CT, SIT, ATC, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT)
  Dim bx As Double, Sfb As Double, tens As Integer, ones As Integer, Agg As Double, THRS(1 To 5) As Double
  Dim i As Integer, z As Integer
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
  bx = Qc
    If Trans = 0 Then
      Sfb = SfuncC(CT, SIT, ATC, (Hn + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
      bx = bx - Hn * Sfb * dx * dy / dt                                          'Terms from right side of equation.
     ElseIf Trans > 200 Then 'Pressurization.
        tens = (Trans - 200) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 200) Mod 10
        Sfb = SfuncC(CT, SIT, ATC, (Hn + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = (THRS(tens) - (Hn + UMA)) * Sfb
        For i = tens To ones + 1 Step -1
            Sfb = SfuncC(CT, SIT, ATC, THRS(i), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                      ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRS(i - 1) - THRS(i)) * Sfb
        Next i
        Sfb = SfuncC(CT, SIT, ATC, THRS(ones), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (UMA - THRS(ones)) * Sfb
```

```
      bx = bx + Agg * dx * dy / dt
     ElseIf Trans > 100 Then 'Depressurization.
        tens = (Trans - 100) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 100) Mod 10
        Sfb = SfuncC(CT, SIT, ATC, THRS(tens), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = (THRS(tens) - (Hn + UMA)) * Sfb
        For i = tens To ones - 1
            Sfb = SfuncC(CT, SIT, ATC, THRS(i + 1), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                      ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRS(i + 1) - THRS(i)) * Sfb
        Next i
        Sfb = SfuncC(CT, SIT, ATC, (hx + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (UMA - THRS(ones)) * Sfb
      bx = bx + Agg * dx * dy / dt
   End If  'Trans.
  FuncBC = bx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function FuncBBC(Qc, Trans, dx, dy, dt, Hn, hx, CT, SIT, ATC, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT, ITS)
  Dim bx As Double, Sfb As Double, tens As Integer, ones As Integer, Agg As Double, THRS(1 To 5) As Double
  Dim i As Integer, z As Integer
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
  If ITS = "Deep Percolation" Then
   Else
   bx = Qc
  End If
    If Trans = 0 Then
      Sfb = SfuncC(CT, SIT, ATC, (Hn + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
      bx = bx - Hn * Sfb * dx * dy / dt                                                    'Terms from right side of equation.
     ElseIf Trans > 200 Then 'Pressurization.
        tens = (Trans - 200) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 200) Mod 10
        Sfb = SfuncC(CT, SIT, ATC, (Hn + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = (THRS(tens) - (Hn + UMA)) * Sfb
        For i = tens To ones + 1 Step -1
            Sfb = SfuncC(CT, SIT, ATC, THRS(i), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                      ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRS(i - 1) - THRS(i)) * Sfb
        Next i
        Sfb = SfuncC(CT, SIT, ATC, THRS(ones), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (UMA - THRS(ones)) * Sfb
      bx = bx + Agg * dx * dy / dt
     ElseIf Trans > 100 Then 'Depressurization.
        tens = (Trans - 100) \ 10 'Integer division result with this (reverse) operator.
```

```
        ones = (Trans - 100) Mod 10
        Sfb = SfuncC(CT, SIT, ATC, THRS(tens), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = (THRS(tens) - (Hn + UMA)) * Sfb
        For i = tens To ones - 1
            Sfb = SfuncC(CT, SIT, ATC, THRS(i + 1), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                     ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRS(i + 1) - THRS(i)) * Sfb
        Next i
        Sfb = SfuncC(CT, SIT, ATC, (hx + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (UMA - THRS(ones)) * Sfb
       bx = bx + Agg * dx * dy / dt
    End If  'Trans.
  FuncBBC = bx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function FuncBTc(Qc, Trans, dx, dy, dt, Hnat, hxat, CT, SIT, ATC, AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, AqB, SB, UMAT, UMS, _
              ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, THRSt1, THRSt2, THRSt3, THRSt4, THRSt5, THRCNTt, ITS,
hoat, DEC)
  Dim bx As Double, Sfb As Double, tens As Integer, ones As Integer, Agg As Double, THRSt(1 To 5) As Double
  Dim i As Integer, z As Integer
THRSt(1) = THRSt1
THRSt(2) = THRSt2
THRSt(3) = THRSt3
THRSt(4) = THRSt4
THRSt(5) = THRSt5
  If ITS = "Deep Percolation" Then
   bx = Qc
   If bx > 0 Then
    If hoat = 0 Then
     bx = 9.99901110999
     GoTo LineBtX
    End If
   End If
   Else
  End If
    If Trans = 0 Then
      Sfb = SfuncCt(CT, SIT, ATC, (Hnat + UMAT), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                  SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                  ovp4, ovw4, Beds4, dx, dy)
      bx = bx - Hnat * Sfb * dx * dy / dt                                                    'Terms from right side of equation.
     ElseIf Trans > 200 Then 'Pressurization.
        tens = (Trans - 200) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 200) Mod 10
        Sfb = SfuncCt(CT, SIT, ATC, (Hnat + UMAT), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                    SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                    ovp4, ovw4, Beds4, dx, dy)
        Agg = (THRSt(tens) - (Hnat + UMAT)) * Sfb
        For i = tens To ones + 1 Step -1
            Sfb = SfuncCt(CT, SIT, ATC, THRSt(i), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                    SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                    ovp4, ovw4, Beds4, dx, dy)
            Agg = Agg + (THRSt(i - 1) - THRSt(i)) * Sfb
```

```
        Next i
        Sfb = SfuncCt(CT, SIT, ATC, THRSt(ones), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                    SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                    ovp4, ovw4, Beds4, dx, dy)
        Agg = Agg + (UMAT - THRSt(ones)) * Sfb
       bx = bx + Agg * dx * dy / dt
     ElseIf Trans > 100 Then 'Depressurization.
        tens = (Trans - 100) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 100) Mod 10
        Sfb = SfuncCt(CT, SIT, ATC, THRSt(tens), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                    SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                    ovp4, ovw4, Beds4, dx, dy)
        Agg = (THRSt(tens) - (Hnat + UMAT)) * Sfb
        For i = tens To ones - 1
            Sfb = SfuncCt(CT, SIT, ATC, THRSt(i + 1), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                    SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                    ovp4, ovw4, Beds4, dx, dy)
            Agg = Agg + (THRSt(i + 1) - THRSt(i)) * Sfb
        Next i
        Sfb = SfuncCt(CT, SIT, ATC, (hxat + UMAT), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                    SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                    ovp4, ovw4, Beds4, dx, dy)
        Agg = Agg + (UMAT - THRSt(ones)) * Sfb
       bx = bx + Agg * dx * dy / dt
    End If  'Trans.
LineBtX:
    FuncBTc = bx
End Function
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
Function TTB(SIT, ATC, AqT, Bed, bpp, UMS, UMA) 'Function to give transition threshold beneath stream bed (neglects elastic storage in bed material).
Dim ttbt As Double, Incr As Double
If SIT <> "None" Then
 If ATC = "Unconfined" Then
    ttbt = Bed - bpp
 ElseIf ATC = "Confined" Then
  If Bed - bpp < AqT Then     '''=
    ttbt = Bed - bpp
    Else
    ttbt = AqT
  End If
End If
Else 'SIT
 If ATC = "Unconfined" Then
    ttbt = 999
 ElseIf ATC = "Confined" Then
    ttbt = AqT
End If
End If 'SIT
TTB = ttbt
End Function
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
```

```
Function FuncdV(Trans, TransB, WR, ATC, Hnq, Hnnq, AqT, AqB, Ss, Sy, SIT, Bed, CT, bpp, SB, dx, dy, UMA, UMS) 'Change in Cell Storage Volume During
Time Step.
  Dim dV As Double, tttbx As Double, Thick As Double, Incr As Double
     If Trans + TransB = 0 Then  'Whole cell.
       dV = Sfunc(CT, SIT, ATC, WR, Hnnq, AqT, Bed, Ss, Sy, AqB, bpp, SB, UMA, UMS) * (Hnnq - Hnq) * dx * dy
     Else
      If WR < 1 Then
      If Trans = 0 Then  'Not under bed.
        dV = (1 - WR) * SfuncNB(CT, ATC, Hnnq, AqT, AqB, Ss, Sy, UMA) * (Hnnq - Hnq) * dx * dy
       ElseIf Trans = 1 Then             'Depressurization.
        dV = (1 - WR) * (-Ss * (AqT - AqB) * (Hnq - (AqT)) - Sy * ((AqT) - Hnnq)) * dx * dy
       ElseIf Trans = 2 Then             'Pressurization.
        dV = (1 - WR) * (Sy * ((AqT) - Hnq) + Ss * (AqT - AqB) * (Hnnq - (AqT))) * dx * dy
      End If
      End If
      If WR > 0 Then
       tttbx = TTB(SIT, ATC, AqT, Bed, bpp, UMS, UMA)
       Thick = fThick(SIT, ATC, Bed, bpp, AqT, AqB)
       If UMS > UMA Then
        Incr = UMS - UMA
       End If
      If TransB = 0 Then  'B for bed.
        dV = dV + WR * SfuncB(SIT, ATC, Hnnq, AqT, AqB, Bed, Ss, Sy, bpp, SB, UMS, UMA, tttbx, Thick) * (Hnnq - Hnq) * dx * dy
      ElseIf TransB = 1 Then            'Bed Depressurization.
        dV = dV + WR * (-Ss * Thick * ((Hnq + Incr) - tttbx) - Sy * (tttbx - (Hnnq + Incr))) * dx * dy
       ElseIf TransB = 2 Then           'Bed Pressurization
        dV = dV + WR * (Sy * (tttbx - (Hnq + Incr)) + Ss * Thick * ((Hnnq + Incr) - tttbx)) * dx * dy
      End If
      End If
    End If
    FuncdV = dV
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function FuncdVat(Trans, TransB, WRAT, ATC, Hnatt, Hnnatt, AqT, Sigs, Sigma, SIT, Bed, CT, bpp, SB, dx, dy, UMAT, UMS, UMA) 'Change in Cell Storage
Volume During Time Step.
  Dim dVat As Double, ThickT As Double, Incr As Double
     If Trans + TransB = 0 Then  'Whole cell.
       dVat = SfuncAT(CT, SIT, WRAT, Hnnatt, AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) * (Hnnatt - Hnatt) * dx * dy
     Else
      If WRAT < 1 Then
      If Trans = 0 Then  'Not under bed.
        dVat = (1 - WRAT) * Sigma * (Hnnatt - Hnatt) * dx * dy
       ElseIf Trans = 1 Then             'Depressurization.
        dVat = (1 - WRAT) * Sigma * (Hnnatt - Hnatt) * dx * dy
       ElseIf Trans = 2 Then             'Pressurization.
        dVat = (1 - WRAT) * Sigma * (Hnnatt - Hnatt) * dx * dy
      End If
      End If
      If WRAT > 0 Then
       If Bed - bpp > AqT Then
         ThickT = Bed - bpp - AqT
         If UMS > UMAT Then
          Incr = UMS - UMAT
         End If
```

```
        If TransB = 0 Then
          dVat = dVat + WRAT * SfuncATB(SIT, Hnnatt, AqT, Bed, Sigs, Sigma, bpp, SB, UMS, UMAT) * (Hnnatt - Hnatt) * dx * dy
        ElseIf TransB = 1 Then                'Bed Depressurization.
          dVat = dVat + WRAT * (-Sigs * ThickT * ((Hnatt + Incr) - (Bed - bpp)) - Sigma * ((Bed - bpp) - (Hnnatt + Incr))) * dx * dy
        ElseIf TransB = 2 Then                'Just rewet.
          dVat = dVat + WRAT * Sigma * ((Hnnatt + Incr) - (Hnatt + Incr)) * dx * dy
        ElseIf TransB = 3 Then                'Depressurize and dewater.
          dVat = dVat + WRAT * (-Sigs * ThickT * ((Hnatt + Incr) - (Bed - bpp)) - Sigma * ((Bed - bpp) - (Hnnatt + Incr))) * dx * dy
        ElseIf TransB = 4 Then                'Rewet and pressurize.
          dVat = dVat + WRAT * (Sigma * ((Bed - bpp) - (Hnatt + Incr)) + Sigs * ThickT * ((Hnnatt + Incr) - (Bed - bpp))) * dx * dy
        ElseIf TransB = 5 Then                'Just dewater.
          dVat = dVat + WRAT * Sigma * ((Hnnatt + Incr) - (Hnatt + Incr)) * dx * dy
        ElseIf TransB = 6 Then                'Bed Pressurization.
          dVat = dVat + WRAT * (Sigma * ((Bed - bpp) - (Hnatt + Incr)) + Sigs * ThickT * ((Hnnatt + Incr) - (Bed - bpp))) * dx * dy
        End If
      ElseIf Bed - bpp <= AqT Then
      End If
    End If
    End If
    FuncdVat = dVat
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function FuncdVC(Trans, dx, dy, dt, Hnq, hxq, CT, SIT, ATC, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
              ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT) 'Change in Cell
Storage Volume During Time Step.
Dim dV As Double, Sdv As Double, tens As Integer, ones As Integer, Agg As Double, THRS(1 To 5) As Double
Dim i As Integer, z As Integer
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
    If Trans = 0 Then
      Sdv = SfuncC(CT, SIT, ATC, Hnq, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
      dV = Sdv * (hxq - Hnq) * dx * dy
     ElseIf Trans > 200 Then 'Pressurization.
        tens = (Trans - 200) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 200) Mod 10
        Sdv = SfuncC(CT, SIT, ATC, Hnq, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = (THRS(tens) - Hnq) * Sdv
        For i = tens To ones + 1 Step -1
            Sdv = SfuncC(CT, SIT, ATC, THRS(i), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                      ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRS(i - 1) - THRS(i)) * Sdv
        Next i
        Sdv = SfuncC(CT, SIT, ATC, THRS(ones), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (hxq - THRS(ones)) * Sdv
      dV = Agg * dx * dy
     ElseIf Trans > 100 Then 'Depressurization.
        tens = (Trans - 100) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 100) Mod 10
```

```
        Sdv = SfuncC(CT, SIT, ATC, THRS(tens), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = (THRS(tens) - Hnq) * Sdv
        For i = tens To ones - 1
            Sdv = SfuncC(CT, SIT, ATC, THRS(i + 1), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                      ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRS(i + 1) - THRS(i)) * Sdv
        Next i
        Sdv = SfuncC(CT, SIT, ATC, hxq, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                  ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (hxq - THRS(ones)) * Sdv
      dV = Agg * dx * dy
   End If  'Trans.
  FuncdVC = dV
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function FuncdVatC(Trans, dx, dy, dt, Hnatt, hxatt, CT, SIT, ATC, AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, AqB, SB, UMAT, UMS, _
               ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, THRSt1, THRSt2, THRSt3, THRSt4, THRSt5, THRCNTt, DEC) 'Change
in aquitard Cell Storage Volume During Time Step.
Dim dV As Double, Sdv As Double, tens As Integer, ones As Integer, Agg As Double, THRSt(1 To 5) As Double
Dim i As Integer, z As Integer
THRSt(1) = THRSt1
THRSt(2) = THRSt2
THRSt(3) = THRSt3
THRSt(4) = THRSt4
THRSt(5) = THRSt5
    If Trans = 0 Then
       If hxatt > AqT Then
           Sdv = SfuncCt(CT, SIT, ATC, Hnatt, AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
                     ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
          dV = Sdv * (hxatt - Hnatt) * dx * dy
       Else
          dV = 0
       End If
     ElseIf Trans > 200 Then 'Pressurization.
        tens = (Trans - 200) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 200) Mod 10
        Sdv = SfuncCt(CT, SIT, ATC, Hnatt, AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = (THRSt(tens) - Hnatt) * Sdv
        For i = tens To ones + 1 Step -1
            Sdv = SfuncCt(CT, SIT, ATC, THRSt(i), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRSt(i - 1) - THRSt(i)) * Sdv
        Next i
        Sdv = SfuncCt(CT, SIT, ATC, THRSt(ones), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (hxatt - THRSt(ones)) * Sdv
      dV = Agg * dx * dy
     ElseIf Trans > 100 Then 'Depressurization.
        tens = (Trans - 100) \ 10 'Integer division result with this (reverse) operator.
        ones = (Trans - 100) Mod 10
        Sdv = SfuncCt(CT, SIT, ATC, THRSt(tens), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
```

```
        Agg = (THRSt(tens) - Hnatt) * Sdv
        For i = tens To ones - 1
            Sdv = SfuncCt(CT, SIT, ATC, THRSt(i + 1), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
            Agg = Agg + (THRSt(i + 1) - THRSt(i)) * Sdv
        Next i
        Sdv = SfuncCt(CT, SIT, ATC, hxatt, AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, SB, UMAT, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
        Agg = Agg + (hxatt - THRSt(ones)) * Sdv
      dV = Agg * dx * dy
   End If  'Trans.
  FuncdVatC = dV
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function BLTF(Qtt, dS, RSP, ic, Qm, UCF, nc)
Dim BLT As Double
BLT = 0
If ic = "Steady" Then
 If Abs(Qtt) > BLT Then
  BLT = Abs(Qtt)
 End If
 Else
 If Abs(Qm * UCF / nc) > BLT Then
  BLT = Abs(Qm * UCF / nc)
 End If
End If
If Abs(dS) > BLT Then
 BLT = Abs(dS)
End If
If Abs(RSP) > BLT Then
 BLT = Abs(RSP)
End If
BLTF = BLT
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub Transition(i As Integer, r As Integer, Trans() As Integer, TransB() As Integer, _
               ATC As String, Hni() As Double, AqTi() As Double, hx() As Double, _
               UMAi() As Double, CTi() As Integer, SIT As String, BEDi() As Double, _
               bppi() As Double, UMSi() As Double, Hniat() As Double, UMATi() As Double, _
               SB As String, CTP() As Integer, L As Integer, m As Integer, BedCP() As Double)
Dim tttb As Double
Dim im As Integer, jm As Integer

i = 1
For jm = 1 To m
For im = 1 To L
Trans(i) = 0
TransB(i) = 0
Trans(i + r / 2) = 0
TransB(i + r / 2) = 0
```

```
 If CTi(i) <> 4 Then
    Trans(i) = Transit(ATC, Hniat(i), AqTi(i), hx(i), hx(i + r / 2), Hni(i + r / 2), UMAi(i + r / 2))
    Trans(i + r / 2) = Transi(ATC, Hni(i + r / 2), AqTi(i), hx(i + r / 2), UMAi(i + r / 2))
     If SB = "Permissive" Then
     If CTP(jm, im) = 2 Then
          Trans(i) = 0
          If BedCP(jm, im) <= AqTi(i) Then
           Trans(i + r / 2) = 0
          End If
     End If
     End If
 End If

    If CTi(i) = 2 Then
     If SB <> "Permissive" Then
     TransB(i) = TransBt(Hniat(i) + UMSi(i), BEDi(i), bppi(i), hx(i) + UMSi(i), AqTi(i), UMAi(i + r / 2), SIT, UMSi(i), hx(i + r / 2) + UMSi(i))
     tttb = TTB(SIT, ATC, AqTi(i), BEDi(i), bppi(i), UMSi(i), UMAi(i + r / 2))
     If BEDi(i) - bppi(i) > AqTi(i) Then
        TransB(i + r / 2) = TransBi(Hni(i + r / 2) + UMAi(i + r / 2), tttb, hx(i + r / 2) + UMAi(i + r / 2))
       Else
        TransB(i + r / 2) = TransBi(Hni(i + r / 2) + UMSi(i), tttb, hx(i + r / 2) + UMSi(i))
     End If
     End If
    End If

If TransB(i) = 3 Then
  Trans(i) = 1
 ElseIf TransB(i) = 5 Then
  Trans(i) = 1
End If

i = i + 1
Next im
Next jm
End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub TransitionC(r As Integer, Trans() As Integer, Hniat() As Double, Hni() As Double, _
               hx() As Double, THRSt() As Double, THRCNTt() As Integer, THRS() As Double, THRCNT() As Integer, _
               m As Integer, L As Integer, AqT() As Double, UMA() As Double, UMAT() As Double, CTi() As Integer)
Dim i As Integer, jm As Integer, im As Integer
i = 1
For jm = 1 To m
For im = 1 To L
Trans(i) = 0
Trans(i + r / 2) = 0
 If CTi(i) <> 4 Then
  Trans(i) = TransTHRSt(THRSt(jm, im, 1), THRSt(jm, im, 2), THRSt(jm, im, 3), THRSt(jm, im, 4), THRSt(jm, im, 5), _
                        THRCNTt(jm, im), (Hniat(i) + UMAT(jm, im)), (hx(i) + UMAT(jm, im)), (Hni(i + r / 2) + UMAT(jm, im)), _
                        (hx(i + r / 2) + UMAT(jm, im)), AqT(jm, im))  'UMAT for all four on purpose.
  Trans(i + r / 2) = TransTHRS(THRS(jm, im, 1), THRS(jm, im, 2), THRS(jm, im, 3), THRS(jm, im, 4), THRS(jm, im, 5), _
                               THRCNT(jm, im), (Hni(i + r / 2) + UMA(jm, im)), (hx(i + r / 2) + UMA(jm, im)))
 End If
```

```
 i = i + 1
Next im
Next jm
End Sub
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function TransTHRS(THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT, Hnf, hxf)  'Function checks for confinement transition.
Dim i As Integer, z As Integer, THRS(1 To 5) As Double, transf As Double
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
If THRCNT > 0 Then 'Confinement threshold present.
If hxf < Hnf Then      'H falling.
             For z = 1 To THRCNT
              If Hnf >= THRS(z) Then
                   transf = 0 'Pressurized above some threshold.
                   For i = z To THRCNT
                     If hxf < THRS(i) Then 'Depressurization accross some threshold(s).
                        transf = 1 * 100 + z * 10 + i
                     End If  'hxf.
                   Next i
                   Exit For
              End If
              If z = THRCNT Then
                transf = 0 'Depressurized below lowest threshold.
              End If
             Next z
 ElseIf hxf = Hnf Then 'H stable.
  transf = 0
 ElseIf hxf > Hnf Then 'H rising.
             For z = THRCNT To 1 Step -1
              If Hnf < THRS(z) Then
                   transf = 0 'Depressurized below some threshold.
                   For i = z To 1 Step -1
                     If hxf >= THRS(i) Then 'Pressurization accross some threshold(s).
                        transf = 2 * 100 + z * 10 + i
                     End If  'hxf.
                   Next i
                   Exit For
              End If
              If z = 1 Then
                transf = 0 'Pressurized above highest threshold.
              End If
             Next z
End If  'H rise or fall.
Else
transf = 0
End If  'THRCNT.
TransTHRS = transf
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
```

```
Function TransTHRSt(THRSt1, THRSt2, THRSt3, THRSt4, THRSt5, THRCNTt, Hnatf, hxat, Hnf, hxf, AqT) 'Function checks for confinement transition.
Dim i As Integer, z As Integer, THRSt(1 To 5) As Double, transft As Double
Dim hxatt As Double
hxatt = hxat
THRSt(1) = THRSt1
THRSt(2) = THRSt2
THRSt(3) = THRSt3
THRSt(4) = THRSt4
THRSt(5) = THRSt5
If hxatt < Hnatf Then      'H falling.
             For z = 1 To THRCNTt
              If Hnatf > THRSt(z) Then
                   transft = 0 'Pressurized above some threshold.
                   For i = z To THRCNTt
                     If hxatt <= THRSt(i) Then 'Depressurization accross some threshold(s).
                        If hxf <= THRSt(i) Then
                           transft = 1 * 100 + z * 10 + i
                        End If
                     End If  'hxf.
                   Next i
                   Exit For
              End If
              If z = THRCNTt Then
                transft = 0 'Depressurized below lowest threshold.
              End If
             Next z
 ElseIf hxatt = Hnatf Then 'H stable.
  If hxf > hxatt Then
      hxatt = hxf
      GoTo Line55
    Else
      transft = 0
  End If
 ElseIf hxatt > Hnatf Then 'H rising.
Line55:
             For z = THRCNTt To 1 Step -1
              If Hnatf <= THRSt(z) Then
                   transft = 0 'Depressurized below some threshold.
                   For i = z To 1 Step -1
                     If hxatt > THRSt(i) Then 'Pressurization accross some threshold(s).
                        transft = 2 * 100 + z * 10 + i
                     End If  'hxf.
                   Next i
                   Exit For
              End If
              If z = 1 Then
                transft = 0 'Pressurized above highest threshold.
              End If
             Next z
End If  'H rise or fall.
TransTHRSt = transft
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function Transi(ATC, Hnit, AqTi, hxt, UMA)
```

```
 Dim Transx As Integer
   If ATC = "Confined" Then
     If Hnit >= AqTi - UMA Then
      If hxt < AqTi - UMA Then
       Transx = 1                    'Depressurization code.
       End If
      ElseIf Hnit < AqTi - UMA Then
      If hxt >= AqTi - UMA Then
       Transx = 2                    'Pressurization code.
      End If
     End If
    End If
 Transi = Transx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function TransBi(Hnit, tttb, hxt)
  Dim TransBx As Integer
     If Hnit >= tttb Then
      If hxt < tttb Then
       TransBx = 1
      End If
      ElseIf Hnit < tttb Then
      If hxt >= tttb Then
       TransBx = 2
      End If
     End If
     TransBi = TransBx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function Transit(ATC, Hniatt, AqTi, hxt, hxq, Hnq, UMA)
 Dim Transx As Integer
     If Hniatt > AqTi - UMA Then
       If hxt <= AqTi - UMA Then
        If hxq <= AqTi - UMA Then
         Transx = 1                    'Depressurization code.
        End If
       End If
      ElseIf Hniatt <= AqTi - UMA Then
        If hxt > AqTi - UMA Then
          Transx = 2                    'Pressurization code.
         ElseIf hxq > AqTi - UMA Then
          Transx = 2
        End If
     End If
 Transit = Transx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function TransBt(Hniatbt, bedbt, bppbt, hxbt, AqTbt, UMAbt, SIT, UMSbt, habt)
  Dim TransBx As Integer, hxbtt As Double
    hxbtt = hxbt
```

```
    If bedbt - bppbt <= AqTbt Then
     TransBx = 0
     Else
     If hxbtt < Hniatbt Then                                          'Falling aquitard head.
      If SIT <> "None" Then
        If Hniatbt > bedbt - bppbt Then
         If hxbtt <= bedbt - bppbt Then
          If hxbtt > AqTbt - UMAbt + UMSbt Then
            TransBx = 1 'Just depressurizes beneath bed.
           Else
            If habt <= AqTbt - UMAbt + UMSbt Then
             TransBx = 3 'Depressurizes and dewaters beneath bed.
            End If
          End If
         End If 'hxbtt.
        Else 'Hniatbt.
        If hxbtt <= AqTbt - UMAbt + UMSbt Then
         If habt <= AqTbt - UMAbt + UMSbt Then
          TransBx = 5    'Just dewaters beneath bed.
         End If
        End If
       End If 'Hniatbt.
       Else
        If hxbtt <= AqTbt - UMAbt + UMSbt Then
         If habt <= AqTbt - UMAbt + UMSbt Then
          TransBx = 5
         End If
        End If
      End If 'SIT.
      ElseIf hxbtt > Hniatbt Then                                      'Rising aquitard head.
Line56: '=======================
      If SIT <> "None" Then
        If Hniatbt = AqTbt - UMAbt + UMSbt Then
         If hxbtt > bedbt - bppbt Then
            TransBx = 4 'Rewets and pressurizes beneath bed.
           Else
            TransBx = 2 'Just rewets beneath bed.
          End If
          ElseIf Hniatbt > AqTbt - UMAbt + UMSbt Then
           If Hniatbt <= bedbt - bppbt Then
            If hxbtt > bedbt - bppbt Then
             TransBx = 6 'Just pressurizes beneath bed.
            End If
           End If
         End If 'Hniatbt.
        Else
        If Hniatbt = AqTbt - UMAbt + UMSbt Then
         TransBx = 2
        End If
       End If 'SIT.
      ElseIf hxbtt = Hniatbt Then                                     'Static aquitard head.
       If habt > hxbtt Then
        hxbtt = habt                                                   'Aquifer holds up or rewets aquitard.
        GoTo Line56
       End If
     End If 'hxbtt.
```

```
    End If ' bedbt.
   TransBt = TransBx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function fkvtb(SIT, kv, Bed, bpp, AqT, kvq, LK, dx, dy, CT, widpp, lenp, widp, SB) 'Representative vertical conductivity for limiting condition.
Dim ATP As Double, hoatg As Double
Dim TopT As Double, Vt As Double, Atr As Double
Dim BEDpp As Double, sitter As String
Dim kva As Double
Dim awidpp As Double
Dim awidp As Double

If CT <> 2 Then
awidpp = 0
awidp = 1
Else
awidpp = widpp
awidp = widp
End If

BEDpp = Bed - bpp

kva = kv

sitter = SIT

If LK <> 1 Then

  If sitter <> "None" Then
   If BEDpp <= AqT Then
    kva = kva * dx * dy * (1 - awidpp / awidp)
    Else
    kva = kva * dx * dy
   End If
   Else 'sitter
    kva = kva * dx * dy
  End If

ElseIf LK = 1 Then
  If sitter <> "None" Then
   If BEDpp <= AqT Then
    kva = kvq * dx * dy * (1 - awidpp / awidp)
    Else
    kva = kvq * dx * dy
   End If
   Else 'sitter
    kva = kvq * dx * dy
  End If
End If

fkvtb = kva

End Function
'*****************************************************************************************************************************************************
```

```
'*************************************************************************************************************************************************
Function FGG(rpc, kv, dx, dy, CT, SB, Bed, bpp, AqT, widpp, widp, lenp, AqB, kvq, AvTp, hx, LK, SIT, hxat, Hn, UMA, UMS, GCT, GCA, UMAT, vfhd, vfch)
'Vertical flow coefficient calculation for flux between layers.
Dim GGG As Double, ATP As Double, AQP As Double, TFP As Double, kvqp As Double, kvb As Double, hog As Double, hoatg As Double
Dim TopT As Double, BottomA As Double, Vt As Double, Va As Double, Vam As Double, Atr As Double, Aaq As Double
Dim kva As Double, kvqa As Double, hxs As Double, BEDpp As Double, sitter As String, hoatgb As Double, ATPb As Double
Dim awidpp As Double
Dim awidp As Double
Dim UMATa As Double

If vfch = "Full" Then
 UMATa = UMAT
Else
 UMATa = 0
End If

If CT <> 2 Then
awidpp = 0
awidp = 1
Else
awidpp = widpp
awidp = widp
End If

Dim hxatp As Double

hxatp = hxat

kva = kv
kvqa = kvq

      BEDpp = Bed - bpp

      If hx = 797979 Then
        hog = (AqT - AqB) * GCA / GCT
       ElseIf hx >= AqT - UMA Then
        hog = AqT - AqB
        If hxatp <= AqT - UMA Then
         hxatp = hx
        End If
       ElseIf hx < AqT - UMA Then
        hog = 0
      End If 'hx AqT.

    If CT = 2 Then
      sitter = SIT
    If sitter = "None" Then
        If hxatp > AqT - UMA Then
          hoatg = hxatp + UMATa - AqT
          hoatgb = hoatg
        End If 'hxatp.
    Else 'sitter
        If BEDpp > AqT Then
          If hxatp >= BEDpp - UMS Then
            hoatgb = BEDpp - AqT
```

```
           ElseIf hxatp <= BEDpp - UMS Then
             If hxatp > AqT - UMA Then
                hoatgb = hxatp + UMATa - AqT
               Else
                hoatgb = 0
             End If
           End If
          ElseIf BEDpp <= AqT Then
           hoatgb = 0
         End If
         If hxatp > AqT - UMA Then
           hoatg = hxatp + UMATa - AqT
         End If 'hxatp.
     End If 'sitter
      ElseIf CT <> 2 Then
         If hxatp > AqT - UMA Then
           hoatg = hxatp + UMATa - AqT
         End If 'hxatp.
     End If

If LK <> 1 Then

AQP = 0
If kvqa = 999999.12321 Then  'Permissive vertical conductivity for aquifer (no head loss).
kvqp = kva
 If hoatg <= 0 Then
   ATP = 0
   ATPb = 0
  Else
   ATP = hoatg
   ATPb = hoatgb
 End If
Else
kvqp = kvqa
If hoatg <= 0 Then     'Vertical head loss in aquifer, when saturated.
  ATP = 0
  ATPb = 0
  Else
   AQP = hog / 2
   If vfhd = "All" Then
      ATP = hoatg
      ATPb = hoatgb
     Else
      ATP = hoatg / 2
      ATPb = hoatgb / 2
   End If
  End If
End If

TFP = AQP + ATP

Vam = 0
BottomA = AqT - AQP
Va = dx * dy * AQP
If CT = 2 Then
If sitter <> "None" Then
```

```
If BEDpp < AqT Then
 If BEDpp < BottomA Then
   Vam = awidpp * lenp * (AqT - BottomA)
  Else
   Vam = awidpp * lenp * (AqT - (BEDpp))
 End If
End If
End If 'sitter
End If 'CT
If AQP > 0 Then
Aaq = (Va - Vam) / AQP
 If Aaq <= 0 Then
  Aaq = 0
  AQP = 0
 End If
Else
Aaq = 0
End If

GGG = 0
If ATP > 0 Then
If AQP <= 0 Then
  GGG = (1 - awidpp / awidp) * kva * dx * dy / ATP
 If ATPb > 0 Then
  GGG = GGG + (awidpp / awidp) * kva * dx * dy / ATPb
 End If
 Else 'AQP.
  If awidpp < awidp Then
   If BEDpp <= AqT Then
     GGG = khbave(AvTp, kva, (1 - awidpp / awidp) * dx * dy, kvqp, Aaq, ATP, AQP) / (ATP + AQP)
    Else
     GGG = khbave(AvTp, kva, (1 - awidpp / awidp) * dx * dy, kvqp, (1 - awidpp / awidp) * dx * dy, ATP, AQP) / (ATP + AQP)
     If ATPb > 0 Then
     GGG = GGG + khbave(AvTp, kva, (awidpp / awidp) * dx * dy, kvqp, (awidpp / awidp) * dx * dy, ATPb, AQP) / (ATPb + AQP)
     End If
   End If
   Else 'widpp.
   If BEDpp > AqT Then
     If ATPb > 0 Then
     GGG = khbave(AvTp, kva, dx * dy, kvqp, dx * dy, ATPb, AQP) / (ATPb + AQP)
     End If
   End If
  End If
End If
End If

ElseIf LK = 1 Then

 If hoatg > 0 Then
  If sitter <> "None" Then
   If BEDpp <= AqT Then
    GGG = kvqa * dx * dy * (1 - awidpp / awidp)
    Else
    GGG = kvqa * dx * dy
   End If
   Else 'sitter
```

```
    GGG = kvqa * dx * dy
  End If
 ElseIf hoatg <= 0 Then
      GGG = 0
 End If

End If

FGG = GGG

End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function fkvtbc(SIT, kv, dx, dy, CT, Bed, bpp, AqT, kvq, LK, _
             widpp, widp, lenp, bppb, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, _
             ovp3, ovw3, Beds3, ovp4, ovw4, Beds4)                          'Representative vertical conductivity for limiting condition.
Dim ATP As Double, hoatg As Double
Dim TopT As Double, Vt As Double, Atr As Double
Dim kva As Double, BEDpp As Double
Dim Bedd1 As Double, Bedd2 As Double, Bedd3 As Double, Bedd4 As Double
Dim Aminus As Double
Dim awidpp As Double
Dim awidp As Double

If CT <> 2 Then
 If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
   Bedd1 = Beds1
   Bedd2 = Beds2
   Bedd3 = Beds3
   Bedd4 = Beds4
    If Bedd1 < AqT Then
      Bedd1 = AqT
    End If
    If Bedd2 < AqT Then
      Bedd2 = AqT
    End If
    If Bedd3 < AqT Then
      Bedd3 = AqT
    End If
    If Bedd4 < AqT Then
      Bedd4 = AqT
    End If
  Aminus = Aminusf(ovp1, ovw1, Bedd1, ovp2, ovw2, Bedd2, ovp3, ovw3, Bedd3, ovp4, ovw4, Bedd4, dx, dy, AqT)
  awidpp = Aminus / (dx * dy)
  If awidpp <= 0 Then
   awidpp = 0
  End If
  awidp = 1
 Else
  awidpp = 0
  awidp = 1
 End If
Else
awidpp = widpp
awidp = widp
```

```
End If

BEDpp = Bed - bpp

kva = kv

If LK <> 1 Then
   If BEDpp <= AqT Then
    If Bed <= AqT Then
     If awidpp + 2 * bppb >= awidp Then
       kva = 0
      Else
       kva = kva * dx * dy * (1 - (awidpp + 2 * bppb) / awidp)
     End If
     Else
       kva = kva * dx * dy * (1 - awidpp / awidp)
    End If
    Else
    If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
      kva = kva * dx * dy * (1 - awidpp / awidp)
     Else
      kva = kva * dx * dy
    End If
   End If
ElseIf LK = 1 Then
   If BEDpp <= AqT Then
    If Bed <= AqT Then
     If awidpp + 2 * bppb >= awidp Then
       kva = 0
      Else
       kva = kvq * dx * dy * (1 - (awidpp + 2 * bppb) / awidp)
     End If
     Else
       kva = kvq * dx * dy * (1 - awidpp / awidp)
    End If
    Else
    kva = kvq * dx * dy
   End If
End If

fkvtbc = kva

End Function
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
Function FGGc(rpc, kv, dx, dy, CT, SB, Bed, bpp, AqT, AqB, kvq, AvTp, hx, LK, SIT, hxat, Hn, UMA, UMS, GCT, GCA, _
             widpp, widp, lenp, bppb, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, UMAT, vfhd, vfch) 'Vertical flow
coefficient calculation for flux between layers.
Dim GGG As Double, ATP As Double, AQP As Double, TFP As Double, kvqp As Double, kvb As Double, hog As Double, hoatg As Double
Dim TopT As Double, BottomA As Double, Vt As Double, Va As Double, Vam As Double, Atr As Double, Aaq As Double
Dim kva As Double, kvqa As Double, hxs As Double, BEDpp As Double, ATPb As Double, hoatgb As Double, ATPk As Double, hoatgk As Double

Dim hxatp As Double
Dim awidpp As Double
Dim awidp As Double
```

```
Dim UMATa As Double

If vfch = "Full" Then
 UMATa = UMAT
Else
 UMATa = 0
End If

If CT <> 2 Then
awidpp = 0
awidp = 1
Else
awidpp = widpp
awidp = widp
End If
hxatp = hxat

kva = kv
kvqa = kvq

      BEDpp = Bed - bpp

      If hx = 797979 Then
        hog = (AqT - AqB) * GCA / GCT
       ElseIf hx >= AqT - UMA Then
        hog = AqT - AqB
        If hxatp <= AqT - UMA Then
         hxatp = hx
        End If
       ElseIf hx < AqT - UMA Then
        hog = 0
      End If 'hx AqT.

    If CT = 2 Then
       If awidpp < awidp Then
        If Bed > AqT Then
          If hxatp > Bed - UMAT Then
            hoatgk = Bed - AqT
           ElseIf hxatp <= Bed - UMAT Then
             If hxatp > AqT - UMA Then
               hoatgk = hxatp + UMATa - AqT
             End If
          End If
         ElseIf Bed <= AqT Then
          hoatgk = 0
        End If
       End If 'widpp
        If BEDpp > AqT Then
          If hxatp > BEDpp - UMS Then
            hoatgb = BEDpp - AqT
           ElseIf hxatp <= BEDpp - UMS Then
             If hxatp > AqT - UMA Then
               hoatgb = hxatp + UMATa - AqT
             End If
          End If
         ElseIf BEDpp <= AqT Then
```

```
          hoatgb = 0
        End If
        If hxatp > AqT - UMA Then
          hoatg = hxatp + UMATa - AqT
        End If 'hxatp.
     ElseIf CT <> 2 Then
        If hxatp > AqT - UMA Then
          hoatg = hxatp + UMATa - AqT
        End If 'hxatp.
    End If

If LK <> 1 Then

AQP = 0
If kvqa = 999999.12321 Then  'Permissive vertical conductivity for aquifer (no head loss).
kvqp = kva
 If hoatg <= 0 Then
  ATP = 0
  ATPb = 0
  ATPk = 0
  Else
  ATP = hoatg
  ATPb = hoatgb
  ATPk = hoatgk
 End If
Else
kvqp = kvqa
If hoatg <= 0 Then     'Vertical head loss in aquifer, when saturated.
   ATP = 0
   ATPb = 0
   ATPk = 0
  Else
   AQP = hog / 2
   If vfhd = "All" Then
      ATP = hoatg
      ATPb = hoatgb
      ATPk = hoatgk
     Else
      ATP = hoatg / 2
      ATPb = hoatgb / 2
      ATPk = hoatgk / 2
   End If
  End If
End If

TFP = AQP + ATP

Vam = 0
BottomA = AqT - AQP
Va = dx * dy * AQP
If CT = 2 Then
 If BEDpp < AqT Then
 If BEDpp < BottomA Then
   Vam = awidpp * lenp * (AqT - BottomA)
  Else
   Vam = awidpp * lenp * (AqT - (BEDpp))
```

```
 End If
 End If
 If Bed < AqT Then
 If Bed < BottomA Then
 If awidpp + 2 * bppb >= awidp Then
   If awidpp < awidp Then
    If Bed < BottomA Then
      Vam = Vam + lenp * (awidp - awidpp) * (AqT - BottomA)
     Else
      Vam = Vam + lenp * (awidp - awidpp) * (AqT - (Bed))
    End If
   End If
  Else
    If Bed < BottomA Then
      Vam = Vam + lenp * (2 * bppb) * (AqT - BottomA)
     Else
      Vam = Vam + lenp * (2 * bppb) * (AqT - (Bed))
    End If
 End If
 End If
 End If
If AQP > 0 Then
Aaq = (Va - Vam) / AQP
 If Aaq <= 0 Then
  Aaq = 0
  AQP = 0
 End If
Else
Aaq = 0
AQP = 0
End If
ElseIf CT <> 2 Then
End If 'CT

GGG = 0
If ATP > 0 Then
If AQP <= 0 Then
  If CT = 2 Then
   If BEDpp <= AqT Then
     If Bed <= AqT Then
      If awidpp + 2 * bppb < awidp Then
       GGG = (1 - (awidpp + 2 * bppb) / awidp) * kva * dx * dy / ATP
      End If
      Else
      If awidpp + 2 * bppb < awidp Then
        GGG = (1 - (awidpp + 2 * bppb) / awidp) * kva * dx * dy / ATP
        If ATPk > 0 Then
        GGG = GGG + ((2 * bppb) / awidp) * kva * dx * dy / ATPk
        End If
       ElseIf awidpp < widp Then
        If ATPk > 0 Then
        GGG = ((awidp - awidpp) / awidp) * kva * dx * dy / ATPk
        End If
      End If
     End If
    Else 'BEDpp.
```

```
      If awidpp + 2 * bppb < awidp Then
        GGG = (1 - (awidpp + 2 * bppb) / awidp) * kva * dx * dy / ATP
        If ATPk > 0 Then
        GGG = GGG + ((2 * bppb) / awidp) * kva * dx * dy / ATPk
        End If
        If ATPb > 0 Then
        GGG = GGG + (awidpp / awidp) * kva * dx * dy / ATPb
        End If
       ElseIf awidpp < awidp Then
        If ATPk > 0 Then
        GGG = ((awidp - awidpp) / awidp) * kva * dx * dy / ATPk
        End If
        If ATPb > 0 Then
        GGG = GGG + (awidpp / awidp) * kva * dx * dy / ATPb
        End If
       Else
        If ATPb > 0 Then
        GGG = kva * dx * dy / ATPb
        End If
      End If
   End If 'BEDpp.
   Else 'CT.
    If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
       GGG = GGG13U(kva, ATP, AqT, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy)
      Else
       GGG = kva * dx * dy / ATP
    End If
  End If 'CT.
 Else 'AQP.
  If CT = 2 Then
   If BEDpp <= AqT Then
     If Bed <= AqT Then
      If awidpp + 2 * bppb < awidp Then
       GGG = khbave(AvTp, kva, (1 - (awidpp + 2 * bppb) / awidp) * dx * dy, kvqp, Aaq, ATP, AQP) / (ATP + AQP)
      End If
      Else 'Bed.
      If awidpp + 2 * bppb < awidp Then
        GGG = khbave(AvTp, kva, (1 - (awidpp + 2 * bppb) / awidp) * dx * dy, kvqp, (1 - (awidpp + 2 * bppb) / awidp) * dx * dy, ATP, AQP) / (ATP +
AQP)
        If ATPk > 0 Then
        GGG = GGG + khbave(AvTp, kva, (2 * bppb / awidp) * dx * dy, kvqp, Aaq - (1 - (awidpp + 2 * bppb) / awidp) * dx * dy, ATPk, AQP) / (ATPk +
AQP)
        End If
       ElseIf awidpp < awidp Then
        If ATPk > 0 Then
        GGG = khbave(AvTp, kva, (1 - awidpp / awidp) * dx * dy, kvqp, Aaq, ATPk, AQP) / (ATPk + AQP)
        End If
      End If
     End If
    Else 'BEDpp.
      If awidpp + 2 * bppb < awidp Then
        GGG = khbave(AvTp, kva, (1 - (awidpp + 2 * bppb) / awidp) * dx * dy, kvqp, (1 - (awidpp + 2 * bppb) / awidp) * dx * dy, ATP, AQP) / (ATP +
AQP)
        If ATPk > 0 Then
        GGG = GGG + khbave(AvTp, kva, ((2 * bppb) / awidp) * dx * dy, kvqp, ((2 * bppb) / awidp) * dx * dy, ATPk, AQP) / (ATPk + AQP)
        End If
```

```
        If ATPb > 0 Then
        GGG = GGG + khbave(AvTp, kva, (awidpp / awidp) * dx * dy, kvqp, (awidpp / awidp) * dx * dy, ATPb, AQP) / (ATPb + AQP)
        End If
       ElseIf awidpp < awidp Then
        If ATPk > 0 Then
        GGG = khbave(AvTp, kva, (1 - awidpp / awidp) * dx * dy, kvqp, (1 - awidpp / awidp) * dx * dy, ATPk, AQP) / (ATPk + AQP)
        End If
        If ATPb > 0 Then
        GGG = GGG + khbave(AvTp, kva, (awidpp / awidp) * dx * dy, kvqp, (awidpp / awidp) * dx * dy, ATPb, AQP) / (ATPb + AQP)
        End If
       Else
        If ATPb > 0 Then
        GGG = khbave(AvTp, kva, dx * dy, kvqp, dx * dy, ATPb, AQP) / (ATPb + AQP)
        End If
      End If
   End If 'BEDpp.
   Else 'CT.
    If ovp1 + ovp2 + ovp3 + ovp4 > 0 Then
       GGG = GGG13F(AvTp, kva, kvqp, ATP, AQP, AqT, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy, vfhd)
      Else
       GGG = khbave(AvTp, kva, dx * dy, kvqp, dx * dy, ATP, AQP) / (ATP + AQP)
    End If
  End If
End If
End If

ElseIf LK = 1 Then

 If hoatg > 0 Then
   If BEDpp <= AqT Then
    If Bed <= AqT Then
     If awidpp + 2 * bppb >= awidp Then
       GGG = 0
      Else
       GGG = kvqa * dx * dy * (1 - (awidpp + 2 * bppb) / awidp)
     End If
     Else
       GGG = kvqa * dx * dy * (1 - awidpp / awidp)
    End If
    Else
    GGG = kvqa * dx * dy
   End If
  Else 'hoatg
    GGG = 0
 End If
End If

FGGc = GGG

End Function
'*****************************************************************************************************************************************

'*****************************************************************************************************************************************
Function QCF(ATC, ic, ITS, Qd, dH, Qdmna, Sigma, dx, dy, Hn, AqT, Ss, AqB, Sy, UMA) 'For Newton Solver.
Dim QCPx As Double
   If ATC = "Confined" Then                    'ATC if - confined.
```

```
          If ic = "Steady" Then               'ic if.
            If ITS = "Deep Percolation" Then  'ITS if.
              QCPx = -(Qd * dH / Qdmna) * Sigma * dx * dy
            Else                              'ITS else.
              If Hn >= AqT - UMA Then
                QCPx = -(Qd * dH / Qdmna) * Ss * (AqT - AqB) * dx * dy
              ElseIf Hn < AqT - UMA Then
                QCPx = -(Qd * dH / Qdmna) * Sy * dx * dy
              End If
            End If                             'ITS end if.
          Else                                 'ic else - instantaneous.
          If ITS = "Deep Percolation" Then      'ITS if again.
            QCPx = -(Qd * dH / Qdmna) * Sigma * dx * dy
          Else                                  'ITS else.
          If Hn >= AqT - UMA Then
           If Hn + (Qd * dH / Qdmna) < AqT - UMA Then 'Depressurization.
            QCPx = ((Hn - (AqT - UMA)) * Ss * (AqT - AqB) + ((AqT - UMA) - (Hn + (Qd * dH / Qdmna))) * Sy) * dx * dy
           Else                                   'Pressurized.
            QCPx = -(Qd * dH / Qdmna) * Ss * (AqT - AqB) * dx * dy
           End If
          ElseIf Hn < AqT - UMA Then
           If Hn + (Qd * dH / Qdmna) > AqT - UMA Then 'Pressurization.
            QCPx = ((Hn - (AqT - UMA)) * Sy + ((AqT - UMA) - (Hn + (Qd * dH / Qdmna))) * Ss * (AqT - AqB)) * dx * dy
           Else                                    'Depressurized.
            QCPx = -(Qd * dH / Qdmna) * Sy * dx * dy
           End If
          End If
          End If                              'ic end if.
          End If                              'ITS end if.
         Else                                'ATC else - unconfined.
            QCPx = -(Qd * dH / Qdmna) * Sy * dx * dy
     End If
   QCF = QCPx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function QCFC(ATC, ic, ITS, Qd, dH, Qdmna, dx, dy, Hn, Hnat, AqT, Ss, AqB, Sy, UMA, _
              SIT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
              SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
              ovp4, ovw4, Beds4, THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT, _
              THRS1t, THRS2t, THRS3t, THRS4t, THRS5t, THRCNTt, CT, DEC)                    'For Newton Solver.
Dim QCPx As Double, Stq As Double, Saq As Double, z As Integer
Dim THRS(1 To 5) As Double, THRSt(1 To 5) As Double
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
THRSt(1) = THRS1t
THRSt(2) = THRS2t
THRSt(3) = THRS3t
THRSt(4) = THRS4t
THRSt(5) = THRS5t
          If ic = "Steady" Then               'ic if.
            If ITS = "Deep Percolation" Then  'ITS if.
```

```
    Stq = SfuncCt(CT, SIT, ATC, (Hnat + UMAT), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
                     SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
                     ovp4, ovw4, Beds4, dx, dy)
    QCPx = -(Qd * dH / Qdmna) * Stq * dx * dy
   Else                                'ITS else.
    Saq = SfuncC(CT, SIT, ATC, (Hn + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                   ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy)
    QCPx = -(Qd * dH / Qdmna) * Saq * dx * dy
  End If                               'ITS end if.
ElseIf ic = "Instantaneous" Then     'ic else.
If ITS = "Deep Percolation" Then      'ITS if again.
Stq = SfuncCt(CT, SIT, ATC, (Hnat + UMAT), AqT, widp, lenp, widpp, Bed, bpp, bppb, Sigs, Sigma, _
             SB, UMAT, UMS, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, _
             ovp4, ovw4, Beds4, dx, dy)
If THRCNTt > 1 Then 'Confinement threshold present.
 If (Qd * dH / Qdmna) <= 0 Then 'H falling.
  For z = 1 To THRCNTt
   If (Hnat + UMAT) >= THRSt(z) Then
    If (Hnat + UMAT) + (Qd * dH / Qdmna) < THRSt(z) Then 'Depressurization.
       QCPx = QcDepres(THRSt(1), THRSt(2), THRSt(3), THRSt(4), THRSt(5), THRCNTt, _
                       Hnat, dx, dy, Sigs, Sigma, AqT, AqB, _
                       CT, SIT, ATC, ovp1, ovw1, Beds1, ovp2, ovw2, _
                       Beds2, ovp3, ovw3, Beds3, _
                       ovp4, ovw4, Beds4, z, Qd, dH, Qdmna, "T", widp, lenp, widpp, _
                       Bed, bpp, bppb, SB, UMAT, UMS)
       Exit For
      Else  'Hn-Qc...
       QCPx = -(Qd * dH / Qdmna) * Stq * dx * dy 'Pressurized.
       Exit For
     End If  'Hn-Qc...
   End If
   If z = THRCNTt Then
    QCPx = -(Qd * dH / Qdmna) * Stq * dx * dy 'Depressurized below lowest threshold.
   End If
  Next z
 ElseIf (Qd * dH / Qdmna) > 0 Then 'H rising.
  For z = THRCNTt To 1 Step -1
   If (Hnat + UMAT) < THRSt(z) Then
    If (Hnat + UMAT) + (Qd * dH / Qdmna) >= THRSt(z) Then 'Pressurization.
       QCPx = QcPres(THRSt(1), THRSt(2), THRSt(3), THRSt(4), THRSt(5), THRCNTt, _
                       Hnat, dx, dy, Sigs, Sigma, AqT, AqB, _
                       CT, SIT, ATC, ovp1, ovw1, Beds1, ovp2, ovw2, _
                       Beds2, ovp3, ovw3, Beds3, _
                       ovp4, ovw4, Beds4, z, Qd, dH, Qdmna, "T", widp, lenp, widpp, _
                       Bed, bpp, bppb, SB, UMAT, UMS)
       Exit For
      Else  'Hn-Qc...
       QCPx = -(Qd * dH / Qdmna) * Stq * dx * dy 'Depressurized.
       Exit For
     End If  'Hn-Qc...
   End If
   If z = 1 Then
    QCPx = -(Qd * dH / Qdmna) * Stq * dx * dy 'Pressurized Above highest threshold.
   End If
  Next z
End If 'H rise or fall.
```

```
        ElseIf THRCNTt = 1 Then 'No confinement threshold present.
         QCPx = -(Qd * dH / Qdmna) * Stq * dx * dy
       End If                                      'Confinement threshold end if.
          Else                                   'ITS else.
          Saq = SfuncC(CT, SIT, ATC, (Hn + UMA), AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                       ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy)
          If THRCNT > 0 Then 'Confinement threshold present.
           If (Qd * dH / Qdmna) <= 0 Then 'H falling.
            For z = 1 To THRCNT
             If (Hn + UMA) >= THRS(z) Then
              If (Hn + UMA) + (Qd * dH / Qdmna) < THRS(z) Then 'Depressurization.
                 QCPx = QcDepres(THRS(1), THRS(2), THRS(3), THRS(4), THRS(5), THRCNT, _
                                 Hn, dx, dy, Ss, Sy, AqT, AqB, _
                                 CT, SIT, ATC, ovp1, ovw1, Beds1, ovp2, ovw2, _
                                 Beds2, ovp3, ovw3, Beds3, _
                                 ovp4, ovw4, Beds4, z, Qd, dH, Qdmna, "A", widp, lenp, widpp, _
                                 Bed, bpp, bppb, SB, UMA, UMS)
                  Exit For
                Else  'Hn-Qc...
                  QCPx = -(Qd * dH / Qdmna) * Saq * dx * dy 'Pressurized.
                  Exit For
               End If  'Hn-Qc...
             End If
             If z = THRCNT Then
              QCPx = -(Qd * dH / Qdmna) * Saq * dx * dy 'Depressurized below lowest threshold.
             End If
            Next z
           ElseIf (Qd * dH / Qdmna) > 0 Then 'H rising.
            For z = THRCNT To 1 Step -1
             If (Hn + UMA) < THRS(z) Then
              If (Hn + UMA) + (Qd * dH / Qdmna) >= THRS(z) Then 'Pressurization.
                 QCPx = QcPres(THRS(1), THRS(2), THRS(3), THRS(4), THRS(5), THRCNT, _
                                 Hn, dx, dy, Ss, Sy, AqT, AqB, _
                                 CT, SIT, ATC, ovp1, ovw1, Beds1, ovp2, ovw2, _
                                 Beds2, ovp3, ovw3, Beds3, _
                                 ovp4, ovw4, Beds4, z, Qd, dH, Qdmna, "A", widp, lenp, widpp, _
                                 Bed, bpp, bppb, SB, UMA, UMS)
                  Exit For
                Else  'Hn-Qc...
                  QCPx = -(Qd * dH / Qdmna) * Saq * dx * dy 'Depressurized.
                  Exit For
               End If  'Hn-Qc...
             End If
             If z = 1 Then
              QCPx = -(Qd * dH / Qdmna) * Saq * dx * dy 'Pressurized Above highest threshold.
             End If
            Next z
          End If 'H rise or fall.
        ElseIf THRCNT = 0 Then 'No confinement threshold present.
         QCPx = -(Qd * dH / Qdmna) * Saq * dx * dy
       End If                                      'Confinement threshold end if.
          End If                                 'ITS end if.
          End If                                 'ic end if.
   QCFC = QCPx
End Function
'*****************************************************************************************************************************************************
```

```
'*************************************************************************************************************************************************
Function msghfun(Tip5, Tip4, Tip3, Tip2, Tip1)
Dim msghhh As String
   If Tip5 = 1 Then
   msghhh = "Aquifer Bottom above Water Table. "
   End If
   If Tip4 = 1 Then
   msghhh = msghhh & "Aquifer Bottom above Top. "
   End If
   If Tip3 = 1 Then
   msghhh = msghhh & "Principal Aquifer Top above Initial Piezometric Surface. "
   End If
   If Tip2 = 1 Then
   msghhh = msghhh & "Initial Head above Ground. "
   End If
   If Tip1 = 1 Then
   msghhh = msghhh & "Ground above Datum. "
   End If
   If Tip1 = 3 Then
   msghhh = msghhh & "Bed below Aquifer. "
   End If
   If Tip1 = 5 Then
   msghhh = msghhh & "Bank below Aquifer. "
   End If
msghfun = msghhh
End Function
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
Function msgh22(Tip5, Tip4, Tip3, Tip2, Tip1)
Dim msgh2 As Integer
   If Tip5 = 1 Then
   msgh2 = 2
   End If
   If Tip4 = 1 Then
   msgh2 = 2
   End If
   If Tip2 = 1 Then
   msgh2 = 2
   End If
   If Tip1 > 2 Then
   msgh2 = 2
   End If
msgh22 = msgh2
End Function
'*************************************************************************************************************************************************

'*************************************************************************************************************************************************
Function msgh222(msgim2, msgti, msgep, NegHead)
Dim msgh22x As Integer
   If msgim2 = "Impulse Tab 0s! " Then
   msgh22x = 2
   End If
If Left(msgti, 4) = "Disc" Then
msgh22x = 2
```

```
End If
   If msgti = "Impulse Exceeds Aquitard Capacity. " Then
   msgh22x = 3
   End If
'-
   If msgep <> "" Then
   msgh22x = 3
   End If
If NegHead = 2 Then
msgh22x = 3
End If
   If NegHead = 3 Then
   msgh22x = 3
   End If
If NegHead = 4 Then
msgh22x = 3
End If
   If NegHead = 6 Then
   msgh22x = 3
   End If
msgh222 = msgh22x
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function hrb222(msgim2, msgti, msgep, NegHead, NegHeadNM)
Dim hrbbx As Integer
   If msgim2 = "Impulse Tab 0s! " Then
   hrbbx = 38
   End If
If Right(msgti, 4) = "Disc" Then
hrbbx = 38
End If
   If msgti = "Impulse Exceeds Aquitard Capacity. " Then
   hrbbx = NegHeadNM - 1
   End If
'-
   If msgep <> "" Then
   hrbbx = NegHeadNM - 1
   End If
If NegHead = 2 Then
hrbbx = NegHeadNM - 1
End If
   If NegHead = 3 Then
   hrbbx = NegHeadNM - 1
   End If
If NegHead = 4 Then
hrbbx = NegHeadNM - 1
End If
   If NegHead = 6 Then
   hrbbx = NegHeadNM - 1
   End If
hrb222 = hrbbx
End Function
'*****************************************************************************************************************************************************
```

```
'*************************************************************************************************************************************************************
Function fun1011(NegHead, msgres, epse)
Dim tenelev As Variant
tenelev = 11
If NegHead = 3 Then
tenelev = epse
End If
If NegHead <> 2 Then
 If msgres <> "" Then
  tenelev = epse
 End If
End If
fun1011 = tenelev
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function FuncNegTT(NegHead)
Dim mtt As String
If NegHead = 2 Then
mtt = "Dewatered Cell(s)! "
ElseIf NegHead = 3 Then
mtt = "Iteration Limit Exceeded! "
Else
mtt = ""
End If
FuncNegTT = mtt
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function sidefunc1tc(NBRS, Bed, AqT, AqTa, SHD, hxt, hxat, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
Dim side As Double, hxts As Double, AqTs As Double
side = 0
If Lons <> SideIn Then
 If NBRS > 0 Then
   GoTo LineS99g
  ElseIf SEB = "None" Then
   GoTo LineS99g
 End If
End If
If Lons <> SideIn Then
  hxts = hxat
  AqTs = AqTa
 Else
  If widpp + 2 * bppb > widp Then
     hxts = hxat
     AqTs = AqTa
   Else
     hxts = hxt
     AqTs = AqT
  End If
End If
If widpp + 2 * bppb >= widp Then
  If WorksheetFunction.RoundDown(hxat, -1) <> 9999990 Then
   If Bed >= AqTa Then
```

```
    side = sidefunc1c(NBRS, Bed, SHD, hxts, hxat, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
    side = side / dxy
   ElseIf Bed < AqTa Then
    If SHD <= AqTa Then
     If hxat <= AqTa Then
      side = 0
     ElseIf hxat > AqTa Then
     side = 0
     End If
     ElseIf SHD > AqTa Then
      If hxat > AqTa Then
        If SHD > hxat Then
         side = hxat - AqTa
         ElseIf SHD <= hxat Then
           side = SHD - AqTa
        End If
       ElseIf hxat <= AqTa Then
        side = 0
      End If
    End If
  End If
  ElseIf WorksheetFunction.RoundDown(hxat, -1) = 9999990 Then
   side = 0
 End If
side = side * dxy
ElseIf widpp + 2 * bppb < widp Then
  If Bed >= AqTs Then
    side = sidefunc1c(NBRS, Bed, SHD, hxts, hxat, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
     If Lons = SideIn Then
       side = side / lenp
      ElseIf Lons <> SideIn Then
       side = side / widpp
     End If
   ElseIf Bed < AqTs Then
    If SHD <= AqTs Then
     If hxts <= AqTs Then
      side = 0
     ElseIf hxts > AqTs Then
     side = 0
     End If
     ElseIf SHD > AqTs Then
      If hxts > AqTs Then
        If SHD > hxts Then
         side = hxts - AqTs
         ElseIf SHD <= hxts Then
           side = SHD - AqTs
        End If
       ElseIf hxts <= AqTs Then
        side = 0
      End If
    End If
  End If
If Lons = SideIn Then
  side = side * lenp
 ElseIf Lons <> SideIn Then
  If WorksheetFunction.RoundDown(hxat, -1) = 9999990 Then
```

```
     side = 0
    Else
     side = side * widpp
   End If
 End If
End If
LineS99g:
sidefunc1tc = side
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sidefunc2tc(NBRS, Bed, AqT, AqTa, SHD, hxt, hxat, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
Dim side As Double, hxts As Double, AqTs As Double
side = 0
If Lons <> SideIn Then
 If NBRS > 0 Then
   GoTo LineS99h
  ElseIf SEB = "None" Then
   GoTo LineS99h
 End If
End If
If Lons <> SideIn Then
  hxts = hxat
  AqTs = AqTa
 Else
  If widpp + 2 * bppb > widp Then
     hxts = hxat
     AqTs = AqTa
   Else
     hxts = hxt
     AqTs = AqT
  End If
End If
If widpp + 2 * bppb >= widp Then
  If WorksheetFunction.RoundDown(hxat, -1) <> 9999990 Then
   If Bed >= AqTa Then
     side = sidefunc2c(NBRS, Bed, SHD, hxts, hxat, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
     side = side / dxy
    ElseIf Bed < AqTa Then
     If SHD <= AqTa Then
      If hxat <= AqTa Then
       side = 0
       ElseIf hxat > AqTa Then
       side = hxat - AqTa
      End If
      ElseIf SHD > AqTa Then
       If hxat > AqTa Then
         If SHD > hxat Then
          side = SHD - hxat
          ElseIf SHD <= hxat Then
            side = hxat - SHD
         End If
        ElseIf hxat <= AqTa Then
         side = SHD - AqTa
       End If
```

```
     End If
    End If
    ElseIf WorksheetFunction.RoundDown(hxat, -1) = 9999990 Then
     side = 0
  End If
 side = side * dxy
 ElseIf widpp + 2 * bppb < widp Then
   If Bed >= AqTs Then
     side = sidefunc2c(NBRS, Bed, SHD, hxts, hxat, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
      If Lons = SideIn Then
        side = side / lenp
       ElseIf Lons <> SideIn Then
        side = side / widpp
      End If
    ElseIf Bed < AqTs Then
     If SHD <= AqTs Then
      If hxts <= AqTs Then
       side = 0
       ElseIf hxts > AqTs Then
       side = hxts - AqTs
      End If
      ElseIf SHD > AqTs Then
       If hxts > AqTs Then
         If SHD > hxts Then
          side = SHD - hxts
          ElseIf SHD <= hxts Then
            side = hxts - SHD
         End If
        ElseIf hxts <= AqTs Then
         side = SHD - AqTs
       End If
     End If
   End If
 If Lons = SideIn Then
   side = side * lenp
  ElseIf Lons <> SideIn Then
   If WorksheetFunction.RoundDown(hxat, -1) = 9999990 Then
     side = 0
    Else
     side = side * widpp
   End If
 End If
End If
LineS99h:
sidefunc2tc = side
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sidefunc1ac(NBRS, Bed, AqT, AqTa, SHD, hx, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
Dim side As Double, hxs As Double, AqTs As Double
side = 0
If Lons <> SideIn Then
 If NBRS > 0 Then
   GoTo LineS99i
  ElseIf SEB = "None" Then
```

```
   GoTo LineS99i
 End If
End If
If Lons <> SideIn Then
  hxs = hxa
  AqTs = AqTa
 Else
  If widpp + 2 * bppb > widp Then
     hxs = hxa
     AqTs = AqTa
   Else
     hxs = hx
     AqTs = AqT
  End If
End If
If widpp + 2 * bppb >= widp Then
  If WorksheetFunction.RoundDown(hxa, -1) <> 9999990 Then
   If Bed >= AqTa Then
     side = 0
    ElseIf Bed < AqTa Then
     If SHD <= AqTa Then
      If hxa <= AqTa Then
       side = sidefunc1c(NBRS, Bed, SHD, hxs, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
       side = side / dxy
       ElseIf hxa > AqTa Then
       side = SHD - Bed
      End If
      ElseIf SHD > AqTa Then
       If hxa > AqTa Then
        side = AqTa - Bed
        ElseIf hxa <= AqTa Then
          If hxa <= Bed Then
            side = 0
           ElseIf hxa > Bed Then
            side = hxa - Bed
          End If
       End If
     End If
   End If
   ElseIf WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
    side = 0
  End If
 side = side * dxy
 ElseIf widpp + 2 * bppb < widp Then
   If Bed >= AqTs Then
     side = 0
    ElseIf Bed < AqTs Then
     If SHD <= AqTs Then
      If hxs <= AqTs Then
       side = sidefunc1c(NBRS, Bed, SHD, hxs, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
       If Lons = SideIn Then
         side = side / lenp
        ElseIf Lons <> SideIn Then
         side = side / widpp
       End If
       ElseIf hxs > AqTs Then
```

```
       side = SHD - Bed
      End If
      ElseIf SHD > AqTs Then
       If hxs > AqTs Then
        side = AqTs - Bed
        ElseIf hxs <= AqTs Then
          If hxs <= Bed Then
            side = 0
           ElseIf hxs > Bed Then
            side = hxs - Bed
          End If
       End If
     End If
   End If
 If Lons = SideIn Then
   side = side * lenp
  ElseIf Lons <> SideIn Then
   If WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
     side = 0
    Else
     side = side * widpp
   End If
 End If
End If
LineS99i:
sidefunc1ac = side
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sidefunc2ac(NBRS, Bed, AqT, AqTa, SHD, hx, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
Dim side As Double, hxs As Double, AqTs As Double
side = 0
If Lons <> SideIn Then
 If NBRS > 0 Then
   GoTo LineS99j
  ElseIf SEB = "None" Then
   GoTo LineS99j
 End If
End If
If Lons <> SideIn Then
  hxs = hxa
  AqTs = AqTa
 Else
  If widpp + 2 * bppb > widp Then
     hxs = hxa
     AqTs = AqTa
   Else
     hxs = hx
     AqTs = AqT
  End If
End If
If widpp + 2 * bppb >= widp Then
  If WorksheetFunction.RoundDown(hxa, -1) <> 9999990 Then
   If Bed >= AqTa Then
     side = 0
```

```
   ElseIf Bed < AqTa Then
    If SHD <= AqTa Then
     If hxa <= AqTa Then
      side = sidefunc2c(NBRS, Bed, SHD, hxs, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
      side = side / dxy
      ElseIf hxa > AqTa Then
      side = AqTa - SHD
     End If
     ElseIf SHD > AqTa Then
      If hxa > AqTa Then
       side = 0
       ElseIf hxa <= AqTa Then
         If hxa <= Bed Then
           side = AqTa - Bed
          ElseIf hxa > Bed Then
           side = AqTa - hxa
         End If
      End If
    End If
  End If
  ElseIf WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
   side = 0
 End If
side = side * dxy
ElseIf widpp + 2 * bppb < widp Then
  If Bed >= AqTs Then
    side = 0
   ElseIf Bed < AqTs Then
    If SHD <= AqTs Then
     If hxs <= AqTs Then
      side = sidefunc2c(NBRS, Bed, SHD, hxs, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
      If Lons = SideIn Then
        side = side / lenp
       ElseIf Lons <> SideIn Then
        side = side / widpp
      End If
      ElseIf hxs > AqTs Then
      side = AqTs - SHD
     End If
     ElseIf SHD > AqTs Then
      If hxs > AqTs Then
       side = 0
       ElseIf hxs <= AqTs Then
         If hxs <= Bed Then
           side = AqTs - Bed
          ElseIf hxs > Bed Then
           side = AqTs - hxs
         End If
      End If
    End If
  End If
If Lons = SideIn Then
  side = side * lenp
 ElseIf Lons <> SideIn Then
  If WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
    side = 0
```

```
    Else
     side = side * widpp
   End If
 End If
End If
LineS99j:
sidefunc2ac = side
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sidefunc1c(NBRS, Bed, SHD, hx, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
Dim side As Double, hxs As Double
side = 0
If Lons <> SideIn Then
 If NBRS > 0 Then
   GoTo LineS99k
  ElseIf SEB = "None" Then
   GoTo LineS99k
 End If
End If
If Lons <> SideIn Then
  hxs = hxa
 Else
  If widpp + 2 * bppb > widp Then
     hxs = hxa
   Else
     hxs = hx
  End If
End If
    If widpp + 2 * bppb >= widp Then
    If WorksheetFunction.RoundDown(hxa, -1) <> 9999990 Then
     If hxa > Bed Then
       If SHD > hxa Then
        side = hxa - Bed
        ElseIf SHD <= hxa Then
         If SHD > Bed Then
          side = SHD - Bed
          ElseIf SHD <= Bed Then
          side = 0
         End If
       End If
      ElseIf hxa <= Bed Then
       side = 0
     End If
    ElseIf WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
    side = 0
   End If
  side = side * dxy
  ElseIf widpp + 2 * bppb < widp Then
     If hxs > Bed Then
      If SHD > hxs Then
       side = hxs - Bed
       ElseIf SHD <= hxs Then
        If SHD > Bed Then
         side = SHD - Bed
```

```
         ElseIf SHD <= Bed Then
         side = 0
        End If
       End If
       ElseIf hxs <= Bed Then
          side = 0
       End If
    If Lons = SideIn Then
      side = side * lenp
     ElseIf Lons <> SideIn Then
    If WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
      side = 0
     Else
      side = side * widpp
    End If
    End If
  End If
LineS99k:
sidefunc1c = side
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sidefunc2c(NBRS, Bed, SHD, hx, hxa, dxy, Lons, SideIn, widpp, lenp, widp, bppb, SEB)
Dim side As Double, hxs As Double
side = 0
If Lons <> SideIn Then
 If NBRS > 0 Then
   GoTo LineS99l
  ElseIf SEB = "None" Then
   GoTo LineS99l
 End If
End If
If Lons <> SideIn Then
  hxs = hxa
 Else
  If widpp + 2 * bppb > widp Then
     hxs = hxa
   Else
     hxs = hx
  End If
End If
   If widpp + 2 * bppb >= widp Then
    If WorksheetFunction.RoundDown(hxa, -1) <> 9999990 Then
     If hxa > Bed Then
       If SHD > hxa Then
        side = SHD - hxa
        ElseIf SHD <= hxa Then
         If SHD > Bed Then
          side = hxa - SHD
          ElseIf SHD <= Bed Then
          side = hxa - Bed
         End If
       End If
      ElseIf hxa <= Bed Then
       side = SHD - Bed
```

```
     End If
    ElseIf WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
    side = 0
   End If
  side = side * dxy
  ElseIf widpp + 2 * bppb < widp Then
     If hxs > Bed Then
      If SHD > hxs Then
       side = SHD - hxs
       ElseIf SHD <= hxs Then
        If SHD > Bed Then
         side = hxs - SHD
         ElseIf SHD <= Bed Then
         side = hxs - Bed
        End If
      End If
      ElseIf hxs <= Bed Then
         side = SHD - Bed
      End If
    If Lons = SideIn Then
      side = side * lenp
     ElseIf Lons <> SideIn Then
   If WorksheetFunction.RoundDown(hxa, -1) = 9999990 Then
     side = 0
    Else
     side = side * widpp
   End If
    End If
   End If
LineS99l:
sidefunc2c = side
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sideE(Bed, UMA, SHD, hx, kpp, bpp, side11, side12)
Dim EEx  As Double
If UMA > 0 Then
If hx <= Bed - UMA Then
 ElseIf hx <= Bed Then
 ElseIf hx <= SHD - UMA Then
  If SHD - hx > 0 Then
   EEx = EEx + kpp * (side12) / (2 * bpp)
  End If
  EEx = EEx + kpp * (side11) / (bpp)
 ElseIf hx <= SHD Then
  EEx = EEx + kpp * (side12) / (bpp)
  EEx = EEx + kpp * (side11) / (bpp)
  ElseIf hx > SHD Then
  EEx = EEx + kpp * (side12) / (bpp * 2)
  EEx = EEx + kpp * (side11) / (bpp)
  End If
 ElseIf UMA = 0 Then
  If hx <= Bed Then
   ElseIf hx <= SHD Then
    EEx = EEx + kpp * side12 / (bpp * 2)
```

```
    EEx = EEx + kpp * side11 / bpp
   ElseIf hx > SHD Then
    EEx = EEx + kpp * side12 / (bpp * 2)
    EEx = EEx + kpp * side11 / bpp
   End If
 End If
sideE = EEx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sideB(Bed, UMA, SHD, hx, kpp, bpp, side11, side12)
Dim bx As Double
If UMA > 0 Then
If hx <= Bed - UMA Then
  bx = bx + kpp * (side12) * ((SHD - Bed) / 2 + UMA) / bpp
 ElseIf hx <= Bed Then
  If SHD - Bed > 0 Then
   bx = bx + kpp * side12 * (SHD - Bed + 2 * UMA - (((hx + UMA - Bed) ^ 2) / (SHD - Bed))) / (bpp * 2)
  End If
 ElseIf hx <= SHD - UMA Then
  If SHD - hx > 0 Then
   bx = bx + kpp * (side12) * (SHD + 2 * UMA - (UMA ^ 2) / (SHD - hx)) / (bpp * 2)
  End If
  bx = bx + kpp * (side11) * (SHD) / (bpp)
 ElseIf hx <= SHD Then
  bx = bx + kpp * (side12) * (SHD) / (bpp)
  bx = bx + kpp * (side11) * (SHD) / (bpp)
  ElseIf hx > SHD Then
  bx = bx + kpp * (side12) * (SHD) / (bpp * 2)
  bx = bx + kpp * (side11) * (SHD) / (bpp)
  End If
 ElseIf UMA = 0 Then
  If hx <= Bed Then
    bx = bx + kpp * side12 * (SHD - Bed) / (bpp * 2)
   ElseIf hx <= SHD Then
    bx = bx + kpp * side12 * SHD / (bpp * 2)
    bx = bx + kpp * side11 * SHD / bpp
   ElseIf hx > SHD Then
    bx = bx + kpp * side12 * SHD / (bpp * 2)
    bx = bx + kpp * side11 * SHD / bpp
   End If
 End If
sideB = bx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sideEE(AqT, Bed, UMA, SHD, hx, kpp, bpp, side11, side12)
Dim EEx As Double
If Bed >= AqT Then
EEx = 0
ElseIf Bed < AqT Then
 If SHD <= AqT Then
  If hx <= AqT Then
    EEx = EEx + sideE(Bed, UMA, SHD, hx, kpp, bpp, side11, side12)
```

```
   ElseIf hx > AqT Then
    EEx = EEx + kpp * side12 / bpp
    EEx = EEx + kpp * side11 / bpp
  End If
 ElseIf SHD > AqT Then
  If UMA > 0 Then
  If hx <= Bed - UMA Then
   ElseIf hx <= Bed Then
   ElseIf hx <= AqT - UMA Then
      EEx = EEx + kpp * side12 / (2 * bpp)
      EEx = EEx + kpp * side11 / bpp
   ElseIf hx <= AqT Then
      EEx = EEx + kpp * side12 / bpp
      EEx = EEx + kpp * side11 / bpp
   ElseIf hx > AqT Then
    EEx = EEx + kpp * side11 / bpp
  End If
 ElseIf UMA = 0 Then
  If hx <= Bed Then
   ElseIf hx <= AqT Then
    EEx = EEx + kpp * side12 / (bpp * 2)
    EEx = EEx + kpp * side11 / bpp
   ElseIf hx > AqT Then
    EEx = EEx + kpp * side11 / bpp
   End If
 End If
End If
End If
sideEE = EEx
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sideBB(AqT, Bed, UMA, SHD, hx, kpp, bpp, side11, side12)
Dim bx As Double
If Bed >= AqT Then
bx = 0
ElseIf Bed < AqT Then
 If SHD <= AqT Then
  If hx <= AqT Then
    bx = bx + sideB(Bed, UMA, SHD, hx, kpp, bpp, side11, side12)
   ElseIf hx > AqT Then
    bx = bx + kpp * side12 * ((AqT + SHD) / 2) / bpp
    bx = bx + kpp * side11 * SHD / bpp
  End If
 ElseIf SHD > AqT Then
 If UMA > 0 Then
  If hx <= Bed - UMA Then
      bx = bx + kpp * side12 * ((((SHD - AqT) + (SHD - Bed)) / 2) + UMA) / bpp
   ElseIf hx <= Bed Then
      bx = bx + kpp * side12 * ((((SHD - AqT) + (SHD - Bed)) / 2) + UMA - (((hx + UMA - Bed) ^ 2) / (2 * (AqT - Bed)))) / bpp
   ElseIf hx <= AqT - UMA Then
      bx = bx + kpp * side12 * ((((SHD - AqT) + (SHD)) / 2) + UMA - ((UMA ^ 2) / (2 * (AqT - hx)))) / bpp
      bx = bx + kpp * side11 * SHD / bpp
   ElseIf hx <= AqT Then
      bx = bx + kpp * side12 * SHD / bpp
```

```
      bx = bx + kpp * side11 * SHD / bpp
   ElseIf hx > AqT Then
    bx = bx + kpp * side11 * SHD / bpp
  End If
 ElseIf UMA = 0 Then
  If hx <= Bed Then
    bx = bx + kpp * side12 * (SHD - AqT + SHD - Bed) / (bpp * 2)
   ElseIf hx <= AqT Then
    bx = bx + kpp * side12 * (SHD - AqT + SHD) / (bpp * 2)
    bx = bx + kpp * side11 * SHD / bpp
   ElseIf hx > AqT Then
    bx = bx + kpp * side11 * SHD / bpp
   End If
 End If
End If
End If
sideBB = bx
End Function
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
Function sideEET(AqT, Bed, UMAT, SHD, hxt, kpp, bpp, side11, side12)
Dim EEx As Double
If Bed >= AqT Then
EEx = EEx + sideE(Bed, UMAT, SHD, hxt, kpp, bpp, side11, side12)
ElseIf Bed < AqT Then
 If SHD <= AqT Then
  If hxt <= AqT Then
    EEx = 0
   ElseIf hxt > AqT Then
    EEx = EEx + kpp * side12 / (2 * bpp)
  End If
 ElseIf SHD > AqT Then
 If UMAT > 0 Then
  If hxt <= SHD - UMAT Then
     If hxt < AqT Then
      Else
       If SHD - hxt > 0 Then
        EEx = EEx + kpp * side12 / (2 * bpp)
       End If
       EEx = EEx + kpp * side11 / bpp
      End If
   ElseIf hxt <= SHD Then
      EEx = EEx + kpp * side12 / bpp
      EEx = EEx + kpp * side11 / bpp
   ElseIf hxt > SHD Then
      EEx = EEx + kpp * side12 / (2 * bpp)
      EEx = EEx + kpp * side11 / bpp
  End If
  ElseIf UMAT = 0 Then
      EEx = EEx + kpp * side12 / (2 * bpp)
      EEx = EEx + kpp * side11 / bpp
  End If
End If
End If
sideEET = EEx
```

```
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function sideBBtt(AqT, Bed, UMAT, SHD, hxt, kpp, bpp, side11, side12)
Dim bx As Double
If Bed >= AqT Then
    bx = bx + sideB(Bed, UMAT, SHD, hxt, kpp, bpp, side11, side12)
ElseIf Bed < AqT Then
 If SHD <= AqT Then
  If hxt <= AqT Then
    bx = 0
   ElseIf hxt > AqT Then
    bx = bx + kpp * side12 * AqT / (2 * bpp)
  End If
 ElseIf SHD > AqT Then
 If UMAT > 0 Then
  If hxt <= SHD - UMAT Then
     If hxt < AqT Then
        bx = bx + kpp * side12 * (((SHD - AqT) / 2) + UMAT - (((hxt + UMAT - AqT) ^ 2) / (2 * (SHD - AqT)))) / bpp
      Else
       If SHD - hxt > 0 Then
         bx = bx + kpp * side12 * ((SHD / 2) + UMAT - ((UMAT ^ 2) / (2 * (SHD - hxt)))) / bpp
       End If
         bx = bx + kpp * side11 * SHD / bpp
     End If
   ElseIf hxt <= SHD Then
      bx = bx + kpp * side12 * SHD / bpp
      bx = bx + kpp * side11 * SHD / bpp
   ElseIf hxt > SHD Then
      bx = bx + kpp * side12 * SHD / (2 * bpp)
      bx = bx + kpp * side11 * SHD / bpp
  End If
  ElseIf UMAT = 0 Then
      bx = bx + kpp * side12 * SHD / (2 * bpp)
      bx = bx + kpp * side11 * SHD / bpp
  End If
End If
End If
sideBBtt = bx
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function fThick(SIT, ATC, Bed, bpp, AqT, AqB)
    If ATC = "Unconfined" Then
       fThick = Bed - bpp - AqB
    ElseIf ATC = "Confined" Then
     If Bed - bpp <= AqT Then
       fThick = Bed - bpp - AqB
       Else
       fThick = AqT - AqB
     End If
   End If
End Function
'*************************************************************************************************************************************************************
```

```
'*****************************************************************************************************************************************************
Function sThresh(UMA, Bed, SHD, hx)
Dim bedd As Double, Thresh As Integer
bedd = Bed
If UMA = 0 Then
 If hx <= bedd Then
   Thresh = 1
  ElseIf hx <= SHD Then
   Thresh = 2
  ElseIf hx > SHD Then
   Thresh = 3
 End If
ElseIf UMA <> 0 Then
 If hx <= bedd - UMA Then
   Thresh = 1
  ElseIf hx <= bedd Then
   Thresh = 2
  ElseIf hx <= SHD - UMA Then
   Thresh = 3
  ElseIf hx <= SHD Then
   Thresh = 4
  ElseIf hx > SHD Then
   Thresh = 5
 End If
End If
sThresh = Thresh
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sThreshT(UMAT, Bed, SHD, AqT1, AqT2, hx)
Dim bedd As Double, AqTT As Double, Thresh As Integer
bedd = Bed
If AqT1 > AqT2 Then
  AqTT = AqT1
 Else
  AqTT = AqT2
End If
If bedd >= AqTT Then
 Thresh = sThresh(UMAT, bedd, SHD, hx)
ElseIf bedd < AqTT Then
 If SHD <= AqTT Then
  If hx = AqTT Then
    Thresh = 10
   ElseIf hx > AqTT Then
    Thresh = 16
  End If
  ElseIf SHD > AqTT Then
   If hx = AqTT Then
     Thresh = 20
    ElseIf hx > AqTT Then
     If UMAT = 0 Then
       If hx <= SHD Then
         Thresh = 21
        ElseIf hx > SHD Then
```

```
        Thresh = 21
       End If
       ElseIf UMAT > 0 Then
       If hx <= SHD - UMAT Then
         Thresh = 31
        ElseIf hx <= SHD Then
         Thresh = 32
        ElseIf hx > SHD Then
         Thresh = 33
       End If
      End If
     ElseIf hx < AqTT Then
      If hx < SHD - UMAT Then
         Thresh = 34
        Else
         Thresh = 32
      End If
   End If
 End If
End If
sThreshT = Thresh
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function sThreshQ(UMA, Bed, SHD, AqT1, AqT2, hxq)
Dim bedd As Double, AqTT As Double, Thresh As Integer
bedd = Bed
If AqT1 > AqT2 Then
  AqTT = AqT1
 Else
  AqTT = AqT2
End If
If bedd >= AqTT Then
 Thresh = 10
ElseIf bedd < AqTT Then
 If SHD <= AqTT Then
  If hxq <= AqTT Then
    Thresh = sThresh(UMA, Bed, SHD, hxq)
   ElseIf hxq > AqTT Then
    Thresh = 16
  End If
  ElseIf SHD > AqTT Then
     If UMA = 0 Then
       If hxq <= bedd Then
         Thresh = 21
        ElseIf hxq <= AqTT Then
         Thresh = 22
        ElseIf hxq > AqTT Then
         Thresh = 23
       End If
      ElseIf UMA > 0 Then
       If hxq <= bedd - UMA Then
         Thresh = 31
        ElseIf hxq <= bedd Then
         Thresh = 32
```

```
        ElseIf hxq <= AqTT - UMA Then
         Thresh = 33
        ElseIf hxq <= AqTT Then
         Thresh = 34
        ElseIf hxq > AqTT Then
         Thresh = 35
       End If
     End If
 End If
End If
sThreshQ = Thresh
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub IROT(RSPT As Double, RSPTA As Double, IMPST As Double, n As Long, nm As Long, nn As Long, _
         o As Long, RSPO() As Double, RSPOA() As Double, nc As Long, RSP() As Double, RSPA() As Double, _
         Qm() As Double, IMPSMX As Double, IMPSMN As Double)
RSPT = 0
RSPTA = 0
IMPST = 0
n = 1
nm = 1
Do While n < nn + 1
 o = 1
 RSPO(nm) = 0
 RSPOA(nm) = 0
 Do While o < nc + 1
  RSPO(nm) = RSPO(nm) + RSP(n)
  RSPOA(nm) = RSPOA(nm) + RSPA(n)
  o = o + 1
  n = n + 1
 Loop
 IMPST = IMPST + Qm(nm)
 RSPT = RSPT + RSPO(nm)
 RSPTA = RSPTA + RSPOA(nm)
 nm = nm + 1
Loop
  If IMPSMX > 0 Then
   If IMPSMN < 0 Then
    IMPSMX = 999999999.123457
   End If
  End If
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function QnoCside(SIT, AvTp, AqT, AqT2, hxat2, h2, UMAT, kh, kh2, dyy, dxx, dxx2, WRAT, WRAT2, Bed, Bed2, bpp, bpp2, _
                  Lons, Lons2, side, h1, SHD2, SHD1, UMS2, UMS1, DEC, SB, UMA, UMA2, hxat, UMAT2, Knocker, CT2, CT1, DEC2, DEC1, _
                  rpc, AqB2, AqB1, hx2, hx1, Hn2, Hn1) 'Function calculates seepage through dry vertical cell face from wet aquitard cell to aquifer
beneath dry aquitard cell.
Dim Qo As Double, h1d As Double, ddd As Double
If Knocker = "Yes." Then
If h1 = 0 Then
```

```
If CT1 <> 2 Then
  If h2 > 0 Then
    h1d = 0.00001 + UMAT - UMA
    If AqT + h1d > DEC Then
      h1d = DEC - AqT
    End If
    ddd = khbarat(SIT, SB, AvTp, kh, h1d, dxx, Lons, WRAT, Bed, bpp, AqT, hxat, UMS1, kh2, h2, dxx2, Lons2, WRAT2, Bed2, bpp2, AqT2, hxat2, UMS2, _
              side, dyy, UMAT, UMAT2, CT1, CT2, DEC1, DEC2, rpc, AqB1, AqB2, hx1, hx2, Hn1, Hn2, UMA2)
    Qo = ddd * (hxat2 - hxat)
   ElseIf h2 = 0 Then
   Qo = 0
  End If
ElseIf SHD1 > AqT Then
  Qo = 0
ElseIf SHD1 <= AqT Then
  If h2 > 0 Then
    h1d = 0.00001 + UMAT - UMA
    If AqT + h1d > DEC Then
      h1d = DEC - AqT
    End If
    ddd = khbarat(SIT, SB, AvTp, kh, h1d, dxx, Lons, WRAT, Bed, bpp, AqT, hxat, UMS1, kh2, h2, dxx2, Lons2, WRAT2, Bed2, bpp2, AqT2, hxat2, UMS2, _
              side, dyy, UMAT, UMAT2, CT1, CT2, DEC1, DEC2, rpc, AqB1, AqB2, hx1, hx2, Hn1, Hn2, UMA2)
    Qo = ddd * (hxat2 - hxat)
  ElseIf h2 = 0 Then
  Qo = 0
  End If
End If
ElseIf h1 > 0 Then
Qo = 0
End If
End If
QnoCside = Qo
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function QnoCsideC(AvTp, CT1, AqT, AqT2, hxat2, h2, UMAT, kh, kh2, dyy, dxx, dxx2, h1, SHD2, SHD1, DEC, UMA, hxat, Knocker As String)   'Function
calculates seepage through dry vertical cell face from wet aquitard cell to aquifer beneath dry aquitard cell.
Dim Qo As Double, h1d As Double, ddd As Double
If Knocker = "Yes." Then
If h1 = 0 Then
If CT1 <> 2 Then
  If h2 > 0 Then
    h1d = 0.00001 + UMAT - UMA
    If AqT + h1d > DEC Then
      h1d = DEC - AqT
    End If
    ddd = khbarC(AvTp, kh, h1d, dxx, kh2, h2, dxx2, dyy)
    Qo = ddd * (hxat2 - hxat)
   ElseIf h2 = 0 Then
   Qo = 0
  End If
ElseIf SHD1 > AqT Then
  Qo = 0
ElseIf SHD1 <= AqT Then
  If h2 > 0 Then
```

```
    h1d = 0.00001 + UMAT - UMA
    If AqT + h1d > DEC Then
      h1d = DEC - AqT
    End If
    ddd = khbarC(AvTp, kh, h1d, dxx, kh2, h2, dxx2, dyy)
    Qo = ddd * (hxat2 - hxat)
  ElseIf h2 = 0 Then
  Qo = 0
  End If
End If
ElseIf h1 > 0 Then
Qo = 0
End If
End If
QnoCsideC = Qo
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub SidesC(side11t As Double, side12t As Double, side21t As Double, side22t As Double, side31t As Double, _
          side32t As Double, side41t As Double, side42t As Double, side11a As Double, side12a As Double, _
          side21a As Double, side22a As Double, side31a As Double, side32a As Double, side41a As Double, _
          side42a As Double, Bed As Double, AqT As Double, AqTa As Double, AqtB As Double, AqTc As Double, _
          AqTd As Double, SHD As Double, hxt As Double, hxx As Double, hxat As Double, hxbt As Double, _
          hxct As Double, hxdt As Double, dx As Double, dy As Double, Lons As Integer, widpp As Double, _
          lenp As Double, hxa As Double, hxb As Double, hxc As Double, hxd As Double, _
          widp As Double, bppb As Double, NBRS1 As Integer, NBRS2 As Integer, NBRS3 As Integer, _
          NBRS4 As Integer, SEB As String)
        side11t = sidefunc1tc(NBRS1, Bed, AqT, AqTa, SHD, hxt, hxat, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side12t = sidefunc2tc(NBRS1, Bed, AqT, AqTa, SHD, hxt, hxat, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side21t = sidefunc1tc(NBRS2, Bed, AqT, AqtB, SHD, hxt, hxbt, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
        side22t = sidefunc2tc(NBRS2, Bed, AqT, AqtB, SHD, hxt, hxbt, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
        side31t = sidefunc1tc(NBRS3, Bed, AqT, AqTc, SHD, hxt, hxct, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side32t = sidefunc2tc(NBRS3, Bed, AqT, AqTc, SHD, hxt, hxct, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side41t = sidefunc1tc(NBRS4, Bed, AqT, AqTd, SHD, hxt, hxdt, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
        side42t = sidefunc2tc(NBRS4, Bed, AqT, AqTd, SHD, hxt, hxdt, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
        side11a = sidefunc1ac(NBRS1, Bed, AqT, AqTa, SHD, hxx, hxa, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side12a = sidefunc2ac(NBRS1, Bed, AqT, AqTa, SHD, hxx, hxa, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side21a = sidefunc1ac(NBRS2, Bed, AqT, AqtB, SHD, hxx, hxb, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
        side22a = sidefunc2ac(NBRS2, Bed, AqT, AqtB, SHD, hxx, hxb, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
        side31a = sidefunc1ac(NBRS3, Bed, AqT, AqTc, SHD, hxx, hxc, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side32a = sidefunc2ac(NBRS3, Bed, AqT, AqTc, SHD, hxx, hxc, dx, Lons, 2, widpp, lenp, widp, bppb, SEB)
        side41a = sidefunc1ac(NBRS4, Bed, AqT, AqTd, SHD, hxx, hxd, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
        side42a = sidefunc2ac(NBRS4, Bed, AqT, AqTd, SHD, hxx, hxd, dy, Lons, 1, widpp, lenp, widp, bppb, SEB)
End Sub
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function OVPfun(widpp, Lons, widp2, widpp2, bppb2, Lons2, side, dxx, dyy, SIT, NBRS2, SEB) 'Function calculates overlap of stream bank from response
cell to neighbor.
Dim OverP As Double
OverP = 0
If Lons2 <> side Then
 If NBRS2 > 0 Then
  GoTo LineO99a
```

```
 End If
End If
If widpp > 0 Then
  OverP = 0
 Else
 If widpp2 > 0 Then
 If (widpp2 + 2 * bppb2) < widp2 Then
   If Lons2 = side Then
     OverP = 0
    ElseIf Lons2 <> side Then
     If SEB = "None" Then
       OverP = 0
      Else
       OverP = bppb2
     End If
   End If 'Lons2
  ElseIf (widpp2 + 2 * bppb2) >= widp2 Then
   If Lons2 = side Then
     OverP = (widpp2 + 2 * bppb2 - widp2) / 2
    ElseIf Lons2 <> side Then
     If SEB = "None" Then
       OverP = 0
      Else
       OverP = bppb2
     End If
   End If 'Lons2
  End If 'widp2
  Else
   OverP = 0
 End If 'widpp2.
End If 'widpp.
LineO99a:
OVPfun = OverP
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function OVWfun(widpp, Lons, widp2, widpp2, bppb2, Lons2, side, dxx, dyy, SIT, NBRS2, SEB) 'Function calculates width of overlap of stream bank from
response cell to neighbor.
Dim OverW As Double
OverW = 0
If Lons2 <> side Then
 If NBRS2 > 0 Then
  GoTo LineW99a
 End If
End If
If widpp > 0 Then
  OverW = 0
 Else
 If widpp2 > 0 Then
 If (widpp2 + 2 * bppb2) < widp2 Then
   If Lons2 = side Then
     OverW = 0
    ElseIf Lons2 <> side Then
     If SEB = "None" Then
       OverW = 0
```

```
      Else
       OverW = widpp2
      End If
    End If 'Lons2
   ElseIf (widpp2 + 2 * bppb2) >= widp2 Then
    If Lons2 = side Then
     If side = 1 Then
       OverW = dyy
      ElseIf side = 2 Then
       OverW = dxx
      End If
     ElseIf Lons2 <> side Then
      If SEB = "None" Then
        OverW = 0
       Else
        OverW = widpp2
      End If
    End If 'Lons2
   End If 'widp2
   Else
    OverW = 0
 End If 'widpp2.
End If 'widpp.
LineW99a:
OVWfun = OverW
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, hs, Bed1) 'Function calculates area of volume affected by bank overlap along side.
Dim Lv As Double, A1 As Double
If hs >= Bed1 Then
If ovp1 > 0 Then
 If ovp2 > 0 Then
   If ovw2 / 2 > dyy / 2 - ovp1 Then
     If ovw1 / 2 > dxx / 2 - ovp2 Then
       Lv = dxx / 2 - ovp2
      ElseIf ovw1 / 2 <= dxx / 2 - ovp2 Then
       Lv = ovw1 / 2
     End If
    ElseIf ovw2 / 2 <= dyy / 2 - ovp1 Then
       Lv = ovw1 / 2
   End If
  ElseIf ovp2 = 0 Then
   Lv = ovw1 / 2
 End If
 If ovp4 > 0 Then
   If ovw4 / 2 > dyy / 2 - ovp1 Then
     If ovw1 / 2 > dxx / 2 - ovp4 Then
       Lv = Lv + dxx / 2 - ovp4
      ElseIf ovw1 / 2 <= dxx / 2 - ovp4 Then
       Lv = Lv + ovw1 / 2
     End If
    ElseIf ovw4 / 2 <= dyy / 2 - ovp1 Then
       Lv = Lv + ovw1 / 2
   End If
```

```
  ElseIf ovp4 = 0 Then
   Lv = Lv + ovw1 / 2
 End If
     A1 = Lv * ovp1
End If 'OVP1.
End If
A1fun = A1
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function A5Afun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Beds1, Beds2, hs) 'Function calculates area of volume affected by bank overlap in corner, part A.
Dim A5 As Double
   If ovw2 / 2 > dyy / 2 - ovp1 Then
     If ovw1 / 2 > dxx / 2 - ovp2 Then
 If hs >= Beds1 Then
  A5 = (ovw1 / 2 - (dxx / 2 - ovp2)) * ovp1
 End If
     End If
   End If
A5Afun = A5
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function A5Bfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Beds1, Beds2, hs) 'Function calculates area of volume affected by bank overlap in corner, part B.
Dim A5 As Double
   If ovw2 / 2 > dyy / 2 - ovp1 Then
     If ovw1 / 2 > dxx / 2 - ovp2 Then
 If hs >= Beds2 Then
  A5 = (ovw2 / 2 - (dyy / 2 - ovp1)) * (ovp2 - (ovw1 / 2 - (dxx / 2 - ovp2)))
 End If
     End If
   End If
A5Bfun = A5
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function A5Cfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Beds1, Beds2, hs) 'Function calculates area of volume affected by bank overlap in corner, part C,
just blocked, not occupied.
Dim A5 As Double
   If ovw2 / 2 >= dyy / 2 - ovp1 Then
     If ovw1 / 2 >= dxx / 2 - ovp2 Then
 If hs >= Beds2 Then
  A5 = (ovp1 - (ovw2 / 2 - (dyy / 2 - ovp1))) * (ovp2 - (ovw1 / 2 - (dxx / 2 - ovp2)))
 End If
     End If
   End If
A5Cfun = A5
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function Vminus(ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dxx, dyy, hs)
Dim V1 As Double, V2 As Double, V3 As Double, V4 As Double, V5 As Double, V6 As Double, V7 As Double, V8 As Double
```

```
Dim Vm As Double          'Function calculates total volume occupied or blocked by bank overlap from adjacent cells.
        V1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, hs, Bed1) * (hs - Bed1)
      If Bed1 < Bed2 Then
        V5 = A5Afun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed1)
        V5 = V5 + A5Bfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed2)
        V5 = V5 + A5Cfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed2)
       ElseIf Bed2 <= Bed1 Then
        V5 = A5Afun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed2)
        V5 = V5 + A5Bfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed1)
        V5 = V5 + A5Cfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed1)
      End If
        V2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dxx, dyy, hs, Bed2) * (hs - Bed2)
      If Bed2 < Bed3 Then
        V6 = A5Afun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed2)
        V6 = V6 + A5Bfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed3)
        V6 = V6 + A5Cfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed3)
       ElseIf Bed3 <= Bed2 Then
        V6 = A5Afun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed3)
        V6 = V6 + A5Bfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed2)
        V6 = V6 + A5Cfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed2)
      End If
        V3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dyy, dxx, hs, Bed3) * (hs - Bed3)
      If Bed3 < Bed4 Then
        V7 = A5Afun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed3)
        V7 = V7 + A5Bfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed4)
        V7 = V7 + A5Cfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed4)
       ElseIf Bed4 <= Bed3 Then
        V7 = A5Afun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed4)
        V7 = V7 + A5Bfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed3)
        V7 = V7 + A5Cfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed3)
      End If
        V4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dxx, dyy, hs, Bed4) * (hs - Bed4)
      If Bed4 < Bed1 Then
        V8 = A5Afun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed4)
        V8 = V8 + A5Bfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed1)
        V8 = V8 + A5Cfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed1)
       ElseIf Bed1 <= Bed4 Then
        V8 = A5Afun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed1)
        V8 = V8 + A5Bfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed4)
        V8 = V8 + A5Cfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed4)
      End If
     Vm = V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8
Vminus = Vm
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function VminusB(ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dxx, dyy, hs, side4)
Dim V1 As Double, V2 As Double, V3 As Double, V4 As Double, V5 As Double, V6 As Double, V7 As Double, V8 As Double
Dim Vm As Double          'Function calculates total volume occupied and blocked by bank overlap from adjacent cells, depending on side specified.
If side4 = 1 Then
        V1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, hs, Bed1) * (hs - Bed1)
        V2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dxx, dyy, hs, Bed2) * (hs - Bed2)
        V4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dxx, dyy, hs, Bed4) * (hs - Bed4)
      If Bed1 < Bed2 Then
        V5 = A5Afun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed1)
```

```
      V5 = V5 + A5Bfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed2)
      V5 = V5 + A5Cfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed2)
     ElseIf Bed2 <= Bed1 Then
      V5 = A5Afun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed2)
      V5 = V5 + A5Bfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed1)
      V5 = V5 + A5Cfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed1)
    End If
    If Bed4 < Bed1 Then
      V8 = A5Afun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed4)
      V8 = V8 + A5Bfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed1)
      V8 = V8 + A5Cfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed1)
     ElseIf Bed1 <= Bed4 Then
      V8 = A5Afun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed1)
      V8 = V8 + A5Bfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed4)
      V8 = V8 + A5Cfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed4)
    End If
   Vm = V1 + V2 / 2 + V4 / 2 + V5 + V8
ElseIf side4 = 2 Then
      V2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dxx, dyy, hs, Bed2) * (hs - Bed2)
      V3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dyy, dxx, hs, Bed3) * (hs - Bed3)
      V1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, hs, Bed1) * (hs - Bed1)
    If Bed2 < Bed3 Then
      V6 = A5Afun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed2)
      V6 = V6 + A5Bfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed3)
      V6 = V6 + A5Cfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed3)
     ElseIf Bed3 <= Bed2 Then
      V6 = A5Afun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed3)
      V6 = V6 + A5Bfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed2)
      V6 = V6 + A5Cfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed2)
    End If
    If Bed1 < Bed2 Then
      V5 = A5Afun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed1)
      V5 = V5 + A5Bfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed2)
      V5 = V5 + A5Cfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (hs - Bed2)
     ElseIf Bed2 <= Bed1 Then
      V5 = A5Afun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed2)
      V5 = V5 + A5Bfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed1)
      V5 = V5 + A5Cfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (hs - Bed1)
    End If
   Vm = V2 + V3 / 2 + V1 / 2 + V6 + V5
ElseIf side4 = 3 Then
      V3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dyy, dxx, hs, Bed3) * (hs - Bed3)
      V4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dxx, dyy, hs, Bed4) * (hs - Bed4)
      V2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dxx, dyy, hs, Bed2) * (hs - Bed2)
    If Bed3 < Bed4 Then
      V7 = A5Afun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed3)
      V7 = V7 + A5Bfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed4)
      V7 = V7 + A5Cfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed4)
     ElseIf Bed4 <= Bed3 Then
      V7 = A5Afun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed4)
      V7 = V7 + A5Bfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed3)
      V7 = V7 + A5Cfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed3)
    End If
    If Bed2 < Bed3 Then
      V6 = A5Afun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed2)
      V6 = V6 + A5Bfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed3)
```

```
        V6 = V6 + A5Cfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (hs - Bed3)
       ElseIf Bed3 <= Bed2 Then
        V6 = A5Afun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed3)
        V6 = V6 + A5Bfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed2)
        V6 = V6 + A5Cfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (hs - Bed2)
       End If
      Vm = V3 + V4 / 2 + V2 / 2 + V7 + V6
 ElseIf side4 = 4 Then
        V4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dxx, dyy, hs, Bed4) * (hs - Bed4)
        V1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, hs, Bed1) * (hs - Bed1)
        V3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp1, ovw1, dyy, dxx, hs, Bed3) * (hs - Bed3)
      If Bed4 < Bed1 Then
        V8 = A5Afun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed4)
        V8 = V8 + A5Bfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed1)
        V8 = V8 + A5Cfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (hs - Bed1)
       ElseIf Bed1 <= Bed4 Then
        V8 = A5Afun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed1)
        V8 = V8 + A5Bfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed4)
        V8 = V8 + A5Cfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (hs - Bed4)
      End If
      If Bed3 < Bed4 Then
        V7 = A5Afun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed3)
        V7 = V7 + A5Bfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed4)
        V7 = V7 + A5Cfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (hs - Bed4)
       ElseIf Bed4 <= Bed3 Then
        V7 = A5Afun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed4)
        V7 = V7 + A5Bfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed3)
        V7 = V7 + A5Cfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (hs - Bed3)
      End If
      Vm = V4 + V1 / 2 + V3 / 2 + V8 + V7
End If
VminusB = Vm
End Function
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
Function AConfSs(ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dxx, dyy, hs, AqB, Ss)
Dim A1 As Double, A2 As Double, A3 As Double, A4 As Double, A5 As Double, A6 As Double, A7 As Double, A8 As Double
Dim A9 As Double, A10 As Double, A11 As Double, A12 As Double, ACSs As Double 'Function calculates elastic storage coefficient times area confined by
bank overlap from adjacent cells.
Dim Cama1 As Double, Cama2 As Double, Cama3 As Double, Cama4 As Double
  Cama1 = Bed1
  Cama2 = Bed2
  Cama3 = Bed3
  Cama4 = Bed4
  If Cama1 < AqB Then
   Cama1 = AqB
  End If
  If Cama2 < AqB Then
   Cama2 = AqB
  End If
  If Cama3 < AqB Then
   Cama3 = AqB
  End If
  If Cama4 < AqB Then
   Cama4 = AqB
```

```
End If
      If Bed1 <= hs Then
      A1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, hs, Bed1) * (Cama1 - AqB) * Ss
      End If
    If Bed1 < Bed2 Then
      If Bed1 <= hs Then
      A5 = A5Afun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (Cama1 - AqB) * Ss
      End If
      If Bed2 <= hs Then
      A9 = A5Bfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs) * (Cama2 - AqB) * Ss
      End If
     ElseIf Bed2 <= Bed1 Then
      If Bed2 <= hs Then
      A5 = A5Afun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (Cama2 - AqB) * Ss
      End If
      If Bed1 <= hs Then
      A9 = A5Bfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs) * (Cama1 - AqB) * Ss
      End If
    End If
      If Bed2 <= hs Then
      A2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dxx, dyy, hs, Bed2) * (Cama2 - AqB) * Ss
      End If
    If Bed2 < Bed3 Then
      If Bed2 <= hs Then
      A6 = A5Afun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (Cama2 - AqB) * Ss
      End If
      If Bed3 <= hs Then
      A10 = A5Bfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs) * (Cama3 - AqB) * Ss
      End If
     ElseIf Bed3 <= Bed2 Then
      If Bed3 <= hs Then
      A6 = A5Afun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (Cama3 - AqB) * Ss
      End If
      If Bed2 <= hs Then
      A10 = A5Bfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs) * (Cama2 - AqB) * Ss
      End If
    End If
      If Bed3 <= hs Then
      A3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dyy, dxx, hs, Bed3) * (Cama3 - AqB) * Ss
      End If
    If Bed3 < Bed4 Then
      If Bed3 <= hs Then
      A7 = A5Afun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (Cama3 - AqB) * Ss
      End If
      If Bed4 <= hs Then
      A11 = A5Bfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs) * (Cama4 - AqB) * Ss
      End If
     ElseIf Bed4 <= Bed3 Then
      If Bed4 <= hs Then
      A7 = A5Afun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (Cama4 - AqB) * Ss
      End If
      If Bed3 <= hs Then
      A11 = A5Bfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs) * (Cama3 - AqB) * Ss
      End If
    End If
      If Bed4 <= hs Then
```

```
        A4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dxx, dyy, hs, Bed4) * (Cama4 - AqB) * Ss
        End If
      If Bed4 < Bed1 Then
        If Bed4 <= hs Then
        A8 = A5Afun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (Cama4 - AqB) * Ss
        End If
        If Bed1 <= hs Then
        A12 = A5Bfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs) * (Cama1 - AqB) * Ss
        End If
       ElseIf Bed1 <= Bed4 Then
        If Bed1 <= hs Then
        A8 = A5Afun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (Cama1 - AqB) * Ss
        End If
        If Bed4 <= hs Then
        A12 = A5Bfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs) * (Cama4 - AqB) * Ss
        End If
      End If
     ACSs = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8 + A9 + A10 + A11 + A12
AConfSs = ACSs
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function AConf(ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dxx, dyy, hs, AqB)
Dim A1 As Double, A2 As Double, A3 As Double, A4 As Double, A5 As Double, A6 As Double, A7 As Double, A8 As Double
Dim A9 As Double, A10 As Double, A11 As Double, A12 As Double, ACn As Double
      If Bed1 > AqB Then                      'Function calculates area confined by bank overlap from adjacent cells.
        If Bed1 <= hs Then
        A1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, hs, Bed1)
        End If
      End If
      If Bed1 < Bed2 Then
       If Bed1 > AqB Then
        If Bed1 <= hs Then
        A5 = A5Afun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs)
        End If
       End If
       If Bed2 > AqB Then
        If Bed2 <= hs Then
        A9 = A5Bfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bed1, Bed2, hs)
        End If
       End If
       ElseIf Bed2 <= Bed1 Then
       If Bed2 > AqB Then
        If Bed2 <= hs Then
        A5 = A5Afun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs)
        End If
       End If
       If Bed1 > AqB Then
        If Bed1 <= hs Then
        A9 = A5Bfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bed2, Bed1, hs)
        End If
       End If
      End If
      If Bed2 > AqB Then
        If Bed2 <= hs Then
```

```
  A2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dxx, dyy, hs, Bed2)
  End If
End If
If Bed2 < Bed3 Then
 If Bed2 > AqB Then
  If Bed2 <= hs Then
  A6 = A5Afun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs)
  End If
 End If
 If Bed3 > AqB Then
  If Bed3 <= hs Then
  A10 = A5Bfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bed2, Bed3, hs)
  End If
 End If
 ElseIf Bed3 <= Bed2 Then
 If Bed3 > AqB Then
  If Bed3 <= hs Then
  A6 = A5Afun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs)
  End If
 End If
 If Bed2 > AqB Then
  If Bed2 <= hs Then
  A10 = A5Bfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bed3, Bed2, hs)
  End If
 End If
End If
If Bed3 > AqB Then
  If Bed3 <= hs Then
  A3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dyy, dxx, hs, Bed3)
  End If
End If
If Bed3 < Bed4 Then
 If Bed3 > AqB Then
  If Bed3 <= hs Then
  A7 = A5Afun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs)
  End If
 End If
 If Bed4 > AqB Then
  If Bed4 <= hs Then
  A11 = A5Bfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bed3, Bed4, hs)
  End If
 End If
 ElseIf Bed4 <= Bed3 Then
  If Bed4 > AqB Then
  If Bed4 <= hs Then
  A7 = A5Afun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs)
  End If
  End If
  If Bed3 > AqB Then
  If Bed3 <= hs Then
  A11 = A5Bfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bed4, Bed3, hs)
  End If
 End If
End If
If Bed4 > AqB Then
  If Bed4 <= hs Then
```

```
        A4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dxx, dyy, hs, Bed4)
        End If
      End If
      If Bed4 < Bed1 Then
       If Bed4 > AqB Then
        If Bed4 <= hs Then
        A8 = A5Afun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs)
        End If
       End If
       If Bed1 > AqB Then
        If Bed1 <= hs Then
        A12 = A5Bfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bed4, Bed1, hs)
        End If
       End If
       ElseIf Bed1 <= Bed4 Then
       If Bed1 > AqB Then
        If Bed1 <= hs Then
        A8 = A5Afun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs)
        End If
       End If
       If Bed4 > AqB Then
        If Bed4 <= hs Then
        A12 = A5Bfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bed1, Bed4, hs)
        End If
       End If
      End If
     ACn = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8 + A9 + A10 + A11 + A12
AConf = ACn
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function Aminusf(ovp1, ovw1, Bedd1, ovp2, ovw2, Bedd2, ovp3, ovw3, Bedd3, ovp4, ovw4, Bedd4, dxx, dyy, AqT)
Dim A1 As Double, A2 As Double, A3 As Double, A4 As Double, A5 As Double, A6 As Double, A7 As Double, A8 As Double
Dim A9 As Double, A10 As Double, A11 As Double, A12 As Double, Am As Double
Dim Hsm As Double
Hsm = AqT + 0.01    'Function calculates area of aquitard lost to bank overlap from adjacent cells.
      If Bedd1 = AqT Then
        A1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dyy, dxx, Hsm, Bedd1)
      End If
      If Bedd1 < Bedd2 Then
       If Bedd1 = AqT Then
        A5 = A5Afun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bedd1, Bedd2, Hsm)
       End If
       If Bedd2 = AqT Then
        A9 = A5Bfun(ovp1, ovw1, ovp2, ovw2, dyy, dxx, Bedd1, Bedd2, Hsm)
       End If
       ElseIf Bedd2 <= Bedd1 Then
       If Bedd2 = AqT Then
        A5 = A5Afun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bedd2, Bedd1, Hsm)
       End If
       If Bedd1 = AqT Then
        A9 = A5Bfun(ovp2, ovw2, ovp1, ovw1, dxx, dyy, Bedd2, Bedd1, Hsm)
       End If
      End If
      If Bedd2 = AqT Then
```

```
        A2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dxx, dyy, Hsm, Bedd2)
       End If
       If Bedd2 < Bedd3 Then
        If Bedd2 = AqT Then
         A6 = A5Afun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bedd2, Bedd3, Hsm)
        End If
        If Bedd3 = AqT Then
         A10 = A5Bfun(ovp2, ovw2, ovp3, ovw3, dxx, dyy, Bedd2, Bedd3, Hsm)
        End If
        ElseIf Bedd3 <= Bedd2 Then
        If Bedd3 = AqT Then
         A6 = A5Afun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bedd3, Bedd2, Hsm)
        End If
        If Bedd2 = AqT Then
         A10 = A5Bfun(ovp3, ovw3, ovp2, ovw2, dyy, dxx, Bedd3, Bedd2, Hsm)
        End If
       End If
       If Bedd3 = AqT Then
         A3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dyy, dxx, Hsm, Bedd3)
       End If
       If Bedd3 < Bedd4 Then
        If Bedd3 = AqT Then
         A7 = A5Afun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bedd3, Bedd4, Hsm)
        End If
        If Bedd4 = AqT Then
         A11 = A5Bfun(ovp3, ovw3, ovp4, ovw4, dyy, dxx, Bedd3, Bedd4, Hsm)
        End If
        ElseIf Bedd4 <= Bedd3 Then
        If Bedd4 = AqT Then
         A7 = A5Afun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bedd4, Bedd3, Hsm)
        End If
        If Bedd3 = AqT Then
         A11 = A5Bfun(ovp4, ovw4, ovp3, ovw3, dxx, dyy, Bedd4, Bedd3, Hsm)
        End If
       End If
       If Bedd4 = AqT Then
         A4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dxx, dyy, Hsm, Bedd4)
       End If
       If Bedd4 < Bedd1 Then
        If Bedd4 = AqT Then
         A8 = A5Afun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bedd4, Bedd1, Hsm)
        End If
        If Bedd1 = AqT Then
         A12 = A5Bfun(ovp4, ovw4, ovp1, ovw1, dxx, dyy, Bedd4, Bedd1, Hsm)
        End If
        ElseIf Bedd1 <= Bedd4 Then
        If Bedd1 = AqT Then
         A8 = A5Afun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bedd1, Bedd4, Hsm)
        End If
        If Bedd4 = AqT Then
         A12 = A5Bfun(ovp1, ovw1, ovp4, ovw4, dyy, dxx, Bedd1, Bedd4, Hsm)
        End If
       End If
      Am = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8 + A9 + A10 + A11 + A12
Aminusf = Am
End Function
```

```
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************
'Subroutine calculates, sorts, and counts confinement thresholds within aquitard cells.
Sub THstackT(ovp() As Double, ovw() As Double, beds() As Double, dx() As Double, dy() As Double, _
             AqT() As Double, Bed() As Double, bpp() As Double, UMS() As Double, UMAT() As Double, _
             THRSt() As Double, THRCNTt() As Integer, m As Integer, L As Integer, CT() As Integer, _
             SIT, widp() As Double, bppb() As Double, widpp() As Double, UMA() As Double)
Dim im As Integer, jm As Integer
Dim TH() As Double, THR() As Double
Dim TH1MX As Double, TH1MN As Double
Dim Incr As Double
Dim i, j, k As Integer
For jm = 1 To m
For im = 1 To L
ReDim TH(1 To 5), THR(1 To 5)
Incr = 0
TH1MX = 0
TH1MN = 0
 If CT(jm, im) = 2 Then
  If SIT <> "None" Then
           If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
            If UMS(jm, im) > UMAT(jm, im) Then
             Incr = UMS(jm, im) - UMAT(jm, im)
            End If
           End If
   If widpp(jm, im) < widp(jm, im) Then
     If bppb(jm, im) > 0 Then
      TH(1) = Bed(jm, im)
      TH(2) = Bed(jm, im) - bpp(jm, im) - Incr
      Else
      TH(1) = Bed(jm, im) - bpp(jm, im) - Incr
     End If
    ElseIf widpp(jm, im) = widp(jm, im) Then
      TH(1) = Bed(jm, im) - bpp(jm, im) - Incr
   End If
  End If
 ElseIf CT(jm, im) <> 2 Then
 If ovp(jm, im, 1) > 0 Then
   TH(1) = beds(jm, im, 1)
 End If
 If ovp(jm, im, 2) > 0 Then
   TH(2) = beds(jm, im, 2)
 End If
 If ovp(jm, im, 3) > 0 Then
   TH(3) = beds(jm, im, 3)
 End If
 If ovp(jm, im, 4) > 0 Then
   TH(4) = beds(jm, im, 4)
 End If
 End If 'CT.
 TH(5) = AqT(jm, im) + UMAT(jm, im) - UMA(jm, im)
 For i = 1 To 5
  If TH(i) < TH(5) Then
    TH(i) = TH(5)
```

```
  End If
 Next i
For i = 1 To 5
      If TH(i) <> 0 Then
      TH1MX = TH(i)
      TH1MN = TH(i)
      Exit For
      End If
Next i
 For i = 1 To 5
  THR(i) = TH(i)
  THRSt(jm, im, i) = -999999
  If TH(i) <> 0 Then
  If TH(i) < TH1MN Then
   TH1MN = TH(i)
  End If
  If TH(i) > TH1MX Then
   TH1MX = TH(i)
  End If
  End If
 Next i
If TH1MX <> 0 Then
For i = 2 To 5              'This Block Sorts in Decending Order from TH to THR.
For j = 1 To i - 1
If TH(i) > THR(j) Then
 For k = i To j + 1 Step -1
  THR(k) = THR(k - 1)
 Next k
 THR(j) = TH(i)
Exit For
End If
Next j
Next i
 THRSt(jm, im, 1) = TH1MX
j = 1
For i = 1 To 5
If THR(i) < THRSt(jm, im, j) Then
 j = j + 1
  THRSt(jm, im, j) = THR(i)
End If
Next i
THRCNTt(jm, im) = j
End If
Next im
Next jm
End Sub
'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************
'Subroutine calculates, sorts, and counts confinement thresholds within cells.
Sub THstack(ovp() As Double, ovw() As Double, beds() As Double, dx() As Double, dy() As Double, AqB() As Double, _
            AqT() As Double, Bed() As Double, bpp() As Double, UMS() As Double, UMA() As Double, ATC As String, _
            THRS() As Double, THRCNT() As Integer, m As Integer, L As Integer, CT() As Integer, SIT, widp() As Double, bppb() As Double, widpp() As
Double)
```

```
Dim im As Integer, jm As Integer
Dim TH() As Double, THR() As Double
Dim TH1MX As Double, TH1MN As Double
Dim i, j, k As Integer
Dim Incr As Double
For jm = 1 To m
For im = 1 To L
ReDim TH(1 To 5), THR(1 To 5)
Incr = 0
TH1MX = 0
TH1MN = 0
 If ATC = "Confined" Then
   TH(1) = AqT(jm, im)
 End If
 If CT(jm, im) = 2 Then
  If SIT <> "None" Then
           If Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
            If UMS(jm, im) > UMA(jm, im) Then
             Incr = UMS(jm, im) - UMA(jm, im)
            End If
           End If
   If ATC = "Confined" Then
    If widpp(jm, im) < widp(jm, im) Then
      If bppb(jm, im) > 0 Then
       TH(2) = Bed(jm, im)
       TH(3) = Bed(jm, im) - bpp(jm, im) - Incr
      Else
       TH(2) = Bed(jm, im) - bpp(jm, im) - Incr
      End If
     ElseIf widpp(jm, im) = widp(jm, im) Then
      TH(1) = Bed(jm, im) - bpp(jm, im) - Incr
    End If
   Else
   If widpp(jm, im) < widp(jm, im) Then
     If bppb(jm, im) > 0 Then
      TH(1) = Bed(jm, im)
      TH(2) = Bed(jm, im) - bpp(jm, im) - Incr
      Else
      TH(1) = Bed(jm, im) - bpp(jm, im) - Incr
     End If
    ElseIf widpp(jm, im) = widp(jm, im) Then
      TH(1) = Bed(jm, im) - bpp(jm, im) - Incr
   End If
  End If
  End If
 ElseIf CT(jm, im) <> 2 Then
 If ovp(jm, im, 1) > 0 Then
   TH(2) = beds(jm, im, 1)
 End If
 If ovp(jm, im, 2) > 0 Then
   TH(3) = beds(jm, im, 2)
 End If
 If ovp(jm, im, 3) > 0 Then
   TH(4) = beds(jm, im, 3)
 End If
 If ovp(jm, im, 4) > 0 Then
```

```
   TH(5) = beds(jm, im, 4)
 End If
 End If 'CT.
For i = 1 To 5
 If TH(i) <> 0 Then
 If TH(i) > AqT(jm, im) Then
  TH(i) = AqT(jm, im)
 End If
 End If
Next i
For i = 1 To 5
      If TH(i) <> 0 Then
      TH1MX = TH(i)
      TH1MN = TH(i)
      Exit For
      End If
Next i
 For i = 1 To 5
  THR(i) = TH(i)
  THRS(jm, im, i) = -999999
  If TH(i) <> 0 Then
  If TH(i) < TH1MN Then
   TH1MN = TH(i)
  End If
  If TH(i) > TH1MX Then
   TH1MX = TH(i)
  End If
  End If
 Next i
If TH1MX <> 0 Then
For i = 2 To 5              'This Block Sorts in Decending Order from TH to THR.
For j = 1 To i - 1
If TH(i) > THR(j) Then
 For k = i To j + 1 Step -1
  THR(k) = THR(k - 1)
 Next k
 THR(j) = TH(i)
Exit For
End If
Next j
Next i
 THRS(jm, im, 1) = TH1MX
j = 1
For i = 1 To 5
If THR(i) < THRS(jm, im, j) Then
 j = j + 1
  THRS(jm, im, j) = THR(i)
End If
Next i
THRCNT(jm, im) = j
End If
Next im
Next jm
End Sub
'*************************************************************************************************************************************************************
```

```
'*************************************************************************************************************************************************************
Function HnDepres(THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT, _
               Qc, Hn, dx, dy, Ss, Sy, AqT, AqB, _
               CT, SIT, ATC, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, _
               ovp4, ovw4, Bed4, z, typ, widp, lenp, widpp, Bed, bpp, bppb, SB, UMA, UMS) 'Function converts instantaneous impulse to head for
depressurization case.
Dim Shde As Double, Qr As Double, Hnn As Double, i As Integer, THRS(1 To 5) As Double, flag As Integer
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
                If typ = "T" Then
                Shde = SfuncCt(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Shde = SfuncC(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
Qr = Qc - ((Hn + UMA) - THRS(z)) * dx * dy * Shde
                If typ = "T" Then
                Shde = SfuncCt(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Shde = SfuncC(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
If z < THRCNT Then
For i = z To THRCNT - 1
  If (THRS(i) - THRS(i + 1)) * dx * dy * Shde < Qr Then
      Qr = Qr - (THRS(i) - THRS(i + 1)) * dx * dy * Shde
                If typ = "T" Then
                Shde = SfuncCt(CT, SIT, ATC, THRS(i + 1) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Shde = SfuncC(CT, SIT, ATC, THRS(i + 1) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
  Else
   Hnn = THRS(i) - Qr / (dx * dy * Shde)
   flag = 1
   Exit For
 End If
Next i
End If
If flag <> 1 Then
  Hnn = THRS(THRCNT) - Qr / (dx * dy * Shde)
End If
Hnn = Hnn - UMA
HnDepres = Hnn
End Function
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
Function HnPres(THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT, _
               Qc, Hn, dx, dy, Ss, Sy, AqT, AqB, _
```

```
              CT, SIT, ATC, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, _
              ovp4, ovw4, Bed4, z, typ, widp, lenp, widpp, Bed, bpp, bppb, SB, UMA, UMS) 'Function converts instantaneous impulse to head for
pressurization case.
Dim Shpr As Double, Qr As Double, Hnn As Double, i As Integer, THRS(1 To 5) As Double, flag As Integer
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
                If typ = "T" Then
                Shpr = SfuncCt(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Shpr = SfuncC(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
Qr = Qc - ((Hn + UMA) - THRS(z)) * dx * dy * Shpr
                If typ = "T" Then
                Shpr = SfuncCt(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Shpr = SfuncC(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
If z > 1 Then
For i = z To 2 Step -1
  If (THRS(i) - THRS(i - 1)) * dx * dy * Shpr > Qr Then
      Qr = Qr - (THRS(i) - THRS(i - 1)) * dx * dy * Shpr
                If typ = "T" Then
                Shpr = SfuncCt(CT, SIT, ATC, THRS(i - 1) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Shpr = SfuncC(CT, SIT, ATC, THRS(i - 1) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
    Else
      Hnn = THRS(i) - Qr / (dx * dy * Shpr)
      flag = 1
      Exit For
  End If
Next i
End If
If flag <> 1 Then
  Hnn = THRS(1) - Qr / (dx * dy * Shpr)
End If
Hnn = Hnn - UMA
HnPres = Hnn
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function QcDepres(THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT, Hn, dx, dy, Ss, Sy, AqT, AqB, _
              CT, SIT, ATC, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, _
              ovp4, ovw4, Bed4, z, Qd, dH, Qdmna, typ, widp, lenp, widpp, Bed, bpp, bppb, SB, UMA, UMS) 'Function aggregates impulse volume for
Newton solver where depressurization occurs.
Dim Sqd As Double, dHr As Double, Qcc As Double, i As Integer, THRS(1 To 5) As Double, flag As Integer
```

```
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
THRS(4) = THRS4
THRS(5) = THRS5
                If typ = "T" Then
                Sqd = SfuncCt(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Sqd = SfuncC(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
Qcc = ((Hn + UMA) - THRS(z)) * Sqd * dx * dy
dHr = (Qd * dH / Qdmna) + ((Hn + UMA) - THRS(z))
                If typ = "T" Then
                Sqd = SfuncCt(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Sqd = SfuncC(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
If z < THRCNT Then
For i = z To THRCNT - 1
  If -(THRS(i) - THRS(i + 1)) > dHr Then
      Qcc = Qcc + (THRS(i) - THRS(i + 1)) * dx * dy * Sqd
      dHr = dHr + (THRS(i) - THRS(i + 1))
                If typ = "T" Then
                Sqd = SfuncCt(CT, SIT, ATC, THRS(i + 1) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Sqd = SfuncC(CT, SIT, ATC, THRS(i + 1) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
   Else
      Qcc = Qcc - dHr * dx * dy * Sqd
      flag = 1
      Exit For
  End If
Next i
End If
If flag <> 1 Then
  Qcc = Qcc - dHr * dx * dy * Sqd
End If
QcDepres = Qcc
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function QcPres(THRS1, THRS2, THRS3, THRS4, THRS5, THRCNT, Hn, dx, dy, Ss, Sy, AqT, AqB, _
               CT, SIT, ATC, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, _
               ovp4, ovw4, Bed4, z, Qd, dH, Qdmna, typ, widp, lenp, widpp, Bed, bpp, bppb, SB, UMA, UMS)   'Function aggregates impulse volume for
Newton solver where pressurization occurs.
Dim Sqp As Double, dHr As Double, Qcc As Double, i As Integer, THRS(1 To 5) As Double, flag As Integer
THRS(1) = THRS1
THRS(2) = THRS2
THRS(3) = THRS3
```

```
THRS(4) = THRS4
THRS(5) = THRS5
                If typ = "T" Then
                Sqp = SfuncCt(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Sqp = SfuncC(CT, SIT, ATC, THRS(z) - 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
Qcc = ((Hn + UMA) - THRS(z)) * Sqp * dx * dy
dHr = (Qd * dH / Qdmna) + ((Hn + UMA) - THRS(z))
                If typ = "T" Then
                Sqp = SfuncCt(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Sqp = SfuncC(CT, SIT, ATC, THRS(z) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
If z > 1 Then
For i = z To 2 Step -1
  If -(THRS(i) - THRS(i - 1)) < dHr Then
      Qcc = Qcc + (THRS(i) - THRS(i - 1)) * dx * dy * Sqp
      dHr = dHr + (THRS(i) - THRS(i - 1))
                If typ = "T" Then
                Sqp = SfuncCt(CT, SIT, ATC, THRS(i - 1) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, _
                SB, UMA, UMS, ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                ElseIf typ = "A" Then
                Sqp = SfuncC(CT, SIT, ATC, THRS(i - 1) + 0.0000001, AqT, widp, lenp, widpp, Bed, bpp, bppb, Ss, Sy, AqB, SB, UMA, UMS, _
                ovp1, ovw1, Bed1, ovp2, ovw2, Bed2, ovp3, ovw3, Bed3, ovp4, ovw4, Bed4, dx, dy)
                End If
   Else
      Qcc = Qcc - dHr * dx * dy * Sqp
      flag = 1
      Exit For
  End If
Next i
End If
If flag <> 1 Then
  Qcc = Qcc - dHr * dx * dy * Sqp
End If
QcPres = Qcc
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'Subroutine to calculate stream response flow from solved heads.
Sub QnoutC(RSP() As Double, RSPA() As Double, Qnno() As Double, i As Integer, jm As Integer, im As Integer, m As Integer, L As Integer, CT() As
Integer, _
          SB As String, Bed() As Double, bpp() As Double, AqT() As Double, hx() As Double, UMS() As Double, kpp() As Double, widpp() As Double, _
          lenp() As Double, SHD() As Double, msgbg As String, SIT As String, Hnnat() As Double, Hnn() As Double, csbt() As Double, _
          csba() As Double, cset() As Double, csea() As Double, CC() As Double, DD() As Double, AA() As Double, BB() As Double, _
          GG() As Double, UMA() As Double, AqTi() As Double, CCt() As Double, DDt() As Double, AAt() As Double, BBt() As Double, zone() As String, _
          qrs As Double, dt As Double, UCF As Double, n As Long, qnoc() As Double, r As Integer, widp() As Double, bppb() As Double, Lons() As
Integer)
 i = 1
```

```
For jm = 1 To m
  For im = 1 To L                    'Solved heads allow explicit calculation of flow through streambed.
    If CT(jm, im) = 2 Then             'Restrictive bed flow by Darcy's law.
      If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
       If hx(i) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then          'Check for drawdown below bottom of bed, bed in aquitard.
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnnat(jm, im) - SHD(jm, im)) / bpp(jm, im)  'Full bed flux term.
       ElseIf hx(i) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im) - UMS(jm, im)) / bpp(jm,
im) 'Bed flux limited by gradient to bottom of bed.
       msgbg = "Bed Flux Limited by Desaturation. "
       End If
      ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
       If hx(i + r / 2) > (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then          'Check for drawdown below bottom of bed, bed below aquitard.
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Hnn(jm, im) - SHD(jm, im)) / bpp(jm, im)  'Full bed flux term.
       ElseIf hx(i + r / 2) <= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
       Qnno(jm, im) = Qnno(jm, im) + kpp(jm, im) * widpp(jm, im) * lenp(jm, im) * (Bed(jm, im) - bpp(jm, im) - SHD(jm, im) - UMS(jm, im)) / bpp(jm,
im) 'Bed flux limited by gradient to bottom of bed.
       msgbg = "Bed Flux Limited by Desaturation. "
       End If
      End If
        If csbt(jm, im, 1) + csba(jm, im, 1) <> 0 Then
        If Lons(jm, im) = 2 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm, im) * cset(jm, im, 1) - csbt(jm, im, 1))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 1) - csba(jm, im, 1))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a1q
         End If
         ElseIf Lons(jm, im) = 1 Then
Line9a1q:     '=========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm - 1, im) * cset(jm, im, 1) - csbt(jm, im, 1))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm - 1, im) * csea(jm, im, 1) - csba(jm, im, 1))
          End If 'Lons
         End If
        If csbt(jm, im, 2) + csba(jm, im, 2) <> 0 Then
        If Lons(jm, im) = 1 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm, im) * cset(jm, im, 2) - csbt(jm, im, 2))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 2) - csba(jm, im, 2))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a2q
         End If
         ElseIf Lons(jm, im) = 2 Then
Line9a2q:     '=========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm, im - 1) * cset(jm, im, 2) - csbt(jm, im, 2))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im - 1) * csea(jm, im, 2) - csba(jm, im, 2))
          End If 'Lons
          End If
        If csbt(jm, im, 3) + csba(jm, im, 3) <> 0 Then
        If Lons(jm, im) = 2 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm, im) * cset(jm, im, 3) - csbt(jm, im, 3))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 3) - csba(jm, im, 3))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a3q
         End If
```

```
         ElseIf Lons(jm, im) = 1 Then
Line9a3q:     '==========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm + 1, im) * cset(jm, im, 3) - csbt(jm, im, 3))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm + 1, im) * csea(jm, im, 3) - csba(jm, im, 3))
           End If 'Lons
          End If
        If csbt(jm, im, 4) + csba(jm, im, 4) <> 0 Then
        If Lons(jm, im) = 1 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm, im) * cset(jm, im, 4) - csbt(jm, im, 4))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im) * csea(jm, im, 4) - csba(jm, im, 4))
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a4q
         End If
         ElseIf Lons(jm, im) = 2 Then
Line9a4q:     '==========================
               Qnno(jm, im) = Qnno(jm, im) + (Hnnat(jm, im + 1) * cset(jm, im, 4) - csbt(jm, im, 4))
               Qnno(jm, im) = Qnno(jm, im) + (Hnn(jm, im + 1) * csea(jm, im, 4) - csba(jm, im, 4))
          End If 'Lons
         End If
     Qnno(jm, im) = qrs * dt * Qnno(jm, im) / UCF  'Flow given correct sign, multiplied by dt to convert to units of volume, converted to acre-feet
if applicable.
    RSP(n) = RSP(n) + Qnno(jm, im) 'Aggregate.
    If zone(jm, im) = 1 Then       'Differentiate.
     RSPA(n) = RSPA(n) + Qnno(jm, im) '.....................................................................................
    End If
   End If                                      'CT end if.
    i = i + 1
  Next im
Next jm
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub Neigh(CT() As Integer, L As Integer, m As Integer, NBRS() As Integer, widp() As Double, _
          widpp() As Double, bppb() As Double)

Dim jm As Integer, im As Integer

For jm = 1 To m
For im = 1 To L
If CT(jm, im) = 2 Then

   If jm - 1 > 0 Then
    If im - 1 > 0 Then
     If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
     If CT(jm, im - 1) = 2 Then
      NBRS(jm, im, 1) = NBRS(jm, im, 1) + 1
     End If
     End If
    End If
    If im + 1 < L + 1 Then
    If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
     If CT(jm, im + 1) = 2 Then
```

```
   NBRS(jm, im, 1) = NBRS(jm, im, 1) + 1
  End If
  End If
 End If
End If

If im - 1 > 0 Then
 If jm - 1 > 0 Then
  If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
  If CT(jm - 1, im) = 2 Then
   NBRS(jm, im, 2) = NBRS(jm, im, 2) + 1
  End If
  End If
 End If
 If jm + 1 < m + 1 Then
  If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
  If CT(jm + 1, im) = 2 Then
   NBRS(jm, im, 2) = NBRS(jm, im, 2) + 1
  End If
  End If
 End If
End If

If jm + 1 < m + 1 Then
 If im + 1 < L + 1 Then
  If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
  If CT(jm, im + 1) = 2 Then
   NBRS(jm, im, 3) = NBRS(jm, im, 3) + 1
  End If
  End If
 End If
 If im - 1 > 0 Then
  If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
  If CT(jm, im - 1) = 2 Then
   NBRS(jm, im, 3) = NBRS(jm, im, 3) + 1
  End If
  End If
 End If
End If

If im + 1 < L + 1 Then
 If jm + 1 < m + 1 Then
  If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
  If CT(jm + 1, im) = 2 Then
   NBRS(jm, im, 4) = NBRS(jm, im, 4) + 1
  End If
  End If
 End If
 If jm - 1 > 0 Then
  If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
  If CT(jm - 1, im) = 2 Then
   NBRS(jm, im, 4) = NBRS(jm, im, 4) + 1
  End If
  End If
 End If
End If
```

```
End If
Next im
Next jm

End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub FortyFive(AqT() As Double, CT() As Integer, DEC() As Double, flaw As Integer, Hni() As Double, Hniat() As Double, Hnn() As Double, _
              Hnnat() As Double, Hno() As Double, Hnoat() As Double, ho() As Double, hoat() As Double, hoo() As Double, hx() As Double, _
              ic As String, IMS As String, ITS As String, msgat As String, msgc As String, msgmt As String, msgt As String, _
              msgtg As String, msgti As String, msgumat As String, MSTA As Double, nc As Long, nm As Long, o As Long, _
              Qc() As Double, QCP() As Double, Qm() As Double, qs As Double, Trans() As Integer, Transnm() As Integer, _
              Transpn() As Integer, Transpnm() As Integer, m As Integer, L As Integer, r As Integer, NegHead As Integer, NegHeadNM As Integer, _
              subsflag As Integer, eps As Double, TransB() As Integer)
Dim i As Long, jm As Long, im As Long
 i = 1
For jm = 1 To m
  For im = 1 To L
    Hnn(jm, im) = hx(i + r / 2)                       'Convert next head solution from system domain vector to model domain array.
    Hnnat(jm, im) = hx(i)
    Hni(i + r / 2) = Hno(jm, im)                      'Revert single index vector to step-original head for storage check below.
    Hniat(i) = Hnoat(jm, im)
     If hoat(jm, im) = DEC(jm, im) - AqT(jm, im) Then
       If msgc = "" Then
       msgc = "Moist Surface! "
       End If
       flaw = 4
     End If
    If ITS = "Deep Percolation" Then  'ITS if.
    If Transpn(i) = 1 Then
     If Trans(i) = 2 Then
      If ic = "Instantaneous" Then
       GoTo Line46
      End If
     End If
    ElseIf Transpn(i) = 2 Then
     If Trans(i) = 1 Then
      If ic = "Instantaneous" Then   'ic if
        GoTo Line46
      End If                         'ic end if.
     End If
    End If
    Else
    If Transpn(i + r / 2) = 1 Then
     If Trans(i + r / 2) = 2 Then
      If ic = "Instantaneous" Then
       GoTo Line46
      End If
     End If
    ElseIf Transpn(i + r / 2) = 2 Then
     If Trans(i + r / 2) = 1 Then
      If ic = "Instantaneous" Then  'ic if
```

```
Line46: '==========================
          msgtg = "Status Oscillation (Instantaneous Impulse Arrival). "
      End If                          'ic end if.
     End If
    End If
    End If                           'ITS end if.
    If Trans(i + r / 2) > Transnm(jm, im) Then
     Transnm(jm, im) = Trans(i + r / 2)
     ElseIf Trans(i) > Transnm(jm, im) Then
     Transnm(jm, im) = Trans(i)
    End If
          If Trans(i) <> 0 Then
            If Abs(Hnnat(jm, im) - Hnoat(jm, im)) > (eps / 10) Then
             msgat = "Aquitard Cell(s) Status Change. "
            End If
            If Trans(i) = 1 Then
             If ITS = "Deep Percolation" Then
              If CT(jm, im) = 1 Then
              If Qc(jm, im) > 0 Then
                flaw = 3
                NegHead = 6
                NegHeadNM = nm + 38
                msgti = "Impulse Exceeds Aquitard Capacity. "
                subsflag = 50
              End If
              End If
             End If
            End If
            ElseIf TransB(i) <> 0 Then
            If Abs(Hnnat(jm, im) - Hnoat(jm, im)) > (eps / 10) Then
             msgat = "Aquitard Cell(s) Status Change. "
            End If
          End If
    If CT(jm, im) = 1 Then
        If Transnm(jm, im) > 0 Then
         If Right(IMS, 4) = "Head" Then
          If ic = "Steady" Then
           If msgti = "" Then
             If ITS = "Deep Percolation" Then
              If Trans(i) > 0 Then
               msgti = "Impulse Distribution Through Confinement Transition. "
              End If
              Else
              If Trans(i + r / 2) > 0 Then
               msgti = "Impulse Distribution Through Confinement Transition. "
              End If
           End If
          End If
         End If
         End If
          If nm > 1 Then
          If Transpnm(jm, im) > 0 Then
          If qs * Qm(nm) > qs * Qm(nm - 1) Then
           If Qc(jm, im) < QCP(jm, im) Then
             msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
           End If
```

```
           ElseIf qs * Qm(nm) <= qs * Qm(nm - 1) Then
           If Qc(jm, im) > QCP(jm, im) Then
             msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
           End If
          End If
          End If
        End If
       End If
    End If
    If CT(jm, im) <> 4 Then
    If Hnnat(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
            msgc = "Inundated Ground! "
            If ITS = "Deep Percolation" Then
              msgti = "Impulse Exceeds Aquitard Capacity. "
             Else
              msgti = "Impulse Exceeds System Capacity. "
            End If
            flaw = 3
            NegHead = 6
            NegHeadNM = nm + 38
            subsflag = 50
            Exit Sub
    End If
    End If
   If CT(jm, im) <> 4 Then
       If MSTA <> 0 Then
        If ho(jm, im) < hoo(jm, im) * MSTA / 100 Then                                    'Check minimum allowed saturated thickness.
          msgmt = "Min Thickness! "                                                           'Update warning message for minimum allowed
saturated thickness.
        End If                                                                             'End if.
       End If
       If msgt = "" Then
       If Trans(i + r / 2) <> 0 Then
         msgt = "Aquifer Cell(s) Confinement Change. "
        ElseIf TransB(i + r / 2) <> 0 Then
         msgt = "Aquifer Cell(s) Confinement Change. "
       End If
     End If
    End If
    If o = nc Then
     QCP(jm, im) = Qc(jm, im)
     Transpnm(jm, im) = Transnm(jm, im)
    End If
    i = i + 1
  Next im
Next jm
End Sub
'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************

'*****************************************************************************************************************************************************************
'*****************************************************************************************************************************************************************
Sub FortyFiveC(AqT() As Double, CT() As Integer, DEC() As Double, flaw As Integer, Hni() As Double, Hniat() As Double, Hnn() As Double, _
             Hnnat() As Double, Hno() As Double, Hnoat() As Double, ho() As Double, hoat() As Double, hoo() As Double, hx() As Double, _
             ic As String, IMS As String, ITS As String, msgat As String, msgc As String, msgmt As String, msgt As String, _
             msgtg As String, msgti As String, msgumat As String, MSTA As Double, nc As Long, nm As Long, o As Long, _
```

```
            Qc() As Double, QCP() As Double, Qm() As Double, qs As Double, Trans() As Integer, Transnm() As Integer, _
            Transpn() As Integer, Transpnm() As Integer, m As Integer, L As Integer, r As Integer, NegHead As Integer, NegHeadNM As Integer, _
            subsflag As Integer, Dewflag() As Integer)
Dim i As Long, jm As Long, im As Long
 i = 1
For jm = 1 To m
  For im = 1 To L
    Hnn(jm, im) = hx(i + r / 2)                       'Convert next head solution from system domain vector to model domain array.
    Hnnat(jm, im) = hx(i)
    Hni(i + r / 2) = Hno(jm, im)                      'Revert single index vector to step-original head for storage check below.
    Hniat(i) = Hnoat(jm, im)
     If hoat(jm, im) = DEC(jm, im) - AqT(jm, im) Then
       If msgc = "" Then
       msgc = "Moist Surface! "
       End If
       flaw = 4
     End If
    If ITS = "Deep Percolation" Then  'ITS if.
    If Transpn(i) \ 100 = 1 Then
     If Trans(i) \ 100 = 2 Then
      If ic = "Instantaneous" Then
       GoTo Line46
      End If
     End If
    ElseIf Transpn(i) \ 100 = 2 Then
     If Trans(i) \ 100 = 1 Then
      If ic = "Instantaneous" Then   'ic if
        GoTo Line46
      End If                         'ic end if.
     End If
    End If
    Else
    If Transpn(i + r / 2) \ 100 = 1 Then
     If Trans(i + r / 2) \ 100 = 2 Then
      If ic = "Instantaneous" Then
       GoTo Line46
      End If
     End If
    ElseIf Transpn(i + r / 2) \ 100 = 2 Then
     If Trans(i + r / 2) \ 100 = 1 Then
      If ic = "Instantaneous" Then  'ic if
Line46: '=========================
          msgtg = "Status Oscillation (Instantaneous Impulse Arrival). "
      End If                         'ic end if.
     End If
    End If
    End If                           'ITS end if.
    If Trans(i + r / 2) > Transnm(jm, im) Then
     Transnm(jm, im) = Trans(i + r / 2)
     ElseIf Trans(i) > Transnm(jm, im) Then
     Transnm(jm, im) = Trans(i)
    End If
          If Trans(i) <> 0 Then
            msgat = "Aquitard Cell(s) Status Change. "
            If Dewflag(i) = 1 Then
             If ITS = "Deep Percolation" Then
```

```
          If CT(jm, im) = 1 Then
          If Qc(jm, im) > 0 Then
            flaw = 3
            NegHead = 6
            NegHeadNM = nm + 38
            msgti = "Impulse Exceeds Aquitard Capacity. "
            subsflag = 50
          End If
          End If
         End If
        End If
       End If
If CT(jm, im) = 1 Then
    If Transnm(jm, im) > 0 Then
     If Right(IMS, 4) = "Head" Then
      If ic = "Steady" Then
       If msgti = "" Then
         If ITS = "Deep Percolation" Then
          If Trans(i) > 0 Then
           msgti = "Impulse Distribution Through Confinement Transition. "
          End If
          Else
          If Trans(i + r / 2) > 0 Then
           msgti = "Impulse Distribution Through Confinement Transition. "
          End If
        End If
       End If
      End If
     End If
       If nm > 1 Then
       If Transpnm(jm, im) > 0 Then
       If qs * Qm(nm) > qs * Qm(nm - 1) Then
        If Qc(jm, im) < QCP(jm, im) Then
          msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
        End If
        ElseIf qs * Qm(nm) <= qs * Qm(nm - 1) Then
        If Qc(jm, im) > QCP(jm, im) Then
          msgti = "Impulse Distribution Instable Through Confinement Transitions, Distribute by Volume Instead. "
        End If
       End If
       End If
     End If
    End If
End If
If CT(jm, im) <> 4 Then
If Hnnat(jm, im) > DEC(jm, im) + 0.00001 Then 'Check that earth cover is not inundated.
         msgc = "Inundated Ground! "
         If ITS = "Deep Percolation" Then
           msgti = "Impulse Exceeds Aquitard Capacity. "
          Else
           msgti = "Impulse Exceeds System Capacity. "
         End If
         flaw = 3
         NegHead = 6
         NegHeadNM = nm + 38
         subsflag = 50
```

```
            Exit Sub
    End If
    End If
   If CT(jm, im) <> 4 Then
       If MSTA <> 0 Then
        If ho(jm, im) < hoo(jm, im) * MSTA / 100 Then                                   'Check minimum allowed saturated thickness.
          msgmt = "Min Thickness! "                                                          'Update warning message for minimum allowed
saturated thickness.
        End If                                                                            'End if.
       End If
       If msgt = "" Then
       If Trans(i + r / 2) <> 0 Then
       msgt = "Aquifer Cell(s) Confinement Change. "
       End If
     End If
    End If
    If o = nc Then
     QCP(jm, im) = Qc(jm, im)
     Transpnm(jm, im) = Transnm(jm, im)
    End If
    i = i + 1
  Next im
Next jm
End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub pload(St As String, SB As String, SIT As String, ITS As String, IMS As String, ic As String, ATC As String, AvTp As String, _
          HTII As Integer, HTIJ As Integer, flaw As Integer)
St = WorksheetFunction.Proper(Sheet11.Cells(12, 5).Value)     'Load schedule type.
SB = WorksheetFunction.Proper(Sheet12.Cells(7, 6).Value)      'Load relative permeability class of streambed.
SIT = WorksheetFunction.Proper(Sheet12.Cells(8, 6).Value)     'Load whether streambed incision represented.
ITS = WorksheetFunction.Proper(Sheet3.Cells(7, 8).Value)      'Load Impulse Type Specification.
IMS = WorksheetFunction.Proper(Sheet8.Cells(7, 6).Value)      'Load Impulse Magnitude Specification.
ic = WorksheetFunction.Proper(Sheet11.Cells(17, 5).Value)     'Load Impulse Character.
ATC = "Confined"                                              'Load Aquifer Thickness Character.
AvTp = WorksheetFunction.Proper(Sheet3.Cells(10, 8).Value)    'Load characteristic average type.
HTII = Sheet11.Cells(24, 5).Value                             'Load head time series output location im index.
HTIJ = Sheet11.Cells(25, 5).Value                             'Load head time series output location jm index.
flaw = 1
End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub KickOut(r As Integer, m As Integer, L As Integer, CT() As Integer, xpo() As Double, hx() As Double, rpc() As Integer, SB As String, SIT As
String, _
            UMS() As Double, BedMat() As Integer, UMA() As Double, AqT() As Double, Bed() As Double, widpp() As Double, bpp() As Double, _
            bppb() As Double, widp() As Double, UMAT() As Double, SHD() As Double, KOC As Integer, AqBi() As Double, ITS As String)
Dim i As Integer, im As Integer, jm As Integer
KOC = 0
If SB = "Restrictive" Then
i = 1
```

```
For jm = 1 To m
For im = 1 To L
 If CT(jm, im) = 2 Then
 If SIT <> "None" Then
 If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
    If hx(i) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      If BedMat(jm, im) <> 1 Then
       KOC = 1
       Exit Sub
      End If
     ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
       If hx(i) > AqT(jm, im) - UMA(jm, im) Then
         If BedMat(jm, im) <> 2 Then
       KOC = 1
       Exit Sub
         End If
        ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
         If BedMat(jm, im) <> 5 Then
       KOC = 1
       Exit Sub
         End If
       End If
    End If
    ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
    If hx(i + r / 2) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      If BedMat(jm, im) <> 3 Then
       KOC = 1
       Exit Sub
      End If
     ElseIf hx(i + r / 2) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      If BedMat(jm, im) <> 4 Then
       KOC = 1
       Exit Sub
      End If
    End If
    End If
    Else 'SIT
    If Bed(jm, im) > AqT(jm, im) Then
    If hx(i) >= (Bed(jm, im) - bpp(jm, im)) Then
      If BedMat(jm, im) <> 1 Then
       KOC = 1
       Exit Sub
      End If
     ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im)) Then
       If hx(i) > AqT(jm, im) - UMA(jm, im) Then
         If BedMat(jm, im) <> 2 Then
       KOC = 1
       Exit Sub
         End If
        ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
         If BedMat(jm, im) <> 5 Then
       KOC = 1
       Exit Sub
         End If
       End If
    End If
```

```
    ElseIf Bed(jm, im) <= AqT(jm, im) Then
    If hx(i + r / 2) >= (Bed(jm, im) - bpp(jm, im)) Then
      If BedMat(jm, im) <> 3 Then
       KOC = 1
       Exit Sub
      End If
     ElseIf hx(i + r / 2) < (Bed(jm, im) - bpp(jm, im)) Then
      If BedMat(jm, im) <> 4 Then
       KOC = 1
       Exit Sub
      End If
    End If
    End If
    End If
 End If
i = i + 1
Next im
Next jm
End If  'Restrictive
End Sub
'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************

'*************************************************************************************************************************************************************
'*************************************************************************************************************************************************************
Sub KickOutC(r As Integer, m As Integer, L As Integer, CT() As Integer, xpo() As Double, hx() As Double, rpc() As Integer, SB As String, SIT As
String, _
            UMS() As Double, BedMat() As Integer, UMA() As Double, AqT() As Double, Bed() As Double, side11a() As Double, side12a() As Double, _
            side11t() As Double, side12t() As Double, side21a() As Double, side22a() As Double, side21t() As Double, side22t() As Double, _
            side31a() As Double, side32a() As Double, side31t() As Double, side32t() As Double, side41a() As Double, side42a() As Double, _
            side41t() As Double, side42t() As Double, Lons() As Integer, widpp() As Double, bpp() As Double, bppb() As Double, widp() As Double, _
            SidMat() As Integer, UMAT() As Double, SHD() As Double, KOC As Integer, AqBi() As Double, ITS As String)
Dim i As Integer, im As Integer, jm As Integer
KOC = 0
i = 1
For jm = 1 To m
For im = 1 To L
 If CT(jm, im) = 2 Then
 If Bed(jm, im) - bpp(jm, im) > AqT(jm, im) Then
    If hx(i) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      If BedMat(jm, im) <> 1 Then
       KOC = 1
       Exit Sub
      End If
     ElseIf hx(i) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
       If hx(i) > AqT(jm, im) - UMA(jm, im) Then
         If BedMat(jm, im) <> 2 Then
       KOC = 1
       Exit Sub
         End If
        ElseIf hx(i) <= AqT(jm, im) - UMA(jm, im) Then
         If BedMat(jm, im) <> 5 Then
       KOC = 1
       Exit Sub
         End If
       End If
```

```
    End If
    ElseIf Bed(jm, im) - bpp(jm, im) <= AqT(jm, im) Then
    If hx(i + r / 2) >= (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      If BedMat(jm, im) <> 3 Then
       KOC = 1
       Exit Sub
      End If
     ElseIf hx(i + r / 2) < (Bed(jm, im) - bpp(jm, im) - UMS(jm, im)) Then
      If BedMat(jm, im) <> 4 Then
       KOC = 1
       Exit Sub
      End If
    End If
    End If
  Dim hxa As Double, hxat As Double
  Dim hxb As Double, hxbt As Double
  Dim hxc As Double, hxct As Double
  Dim hxd As Double, hxdt As Double
       If side11a(jm, im) + side12a(jm, im) + side11t(jm, im) + side12t(jm, im) > 0 Then
       If Lons(jm, im) = 2 Then
        If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
             If SidMat(jm, im, 1, 1) <> sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i)) Then
      KOC = 1
      Exit Sub
            End If
            If SidMat(jm, im, 2, 1) <> sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2)) Then
      KOC = 1
      Exit Sub
            End If
         ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
          GoTo Line9a1k
        End If
        ElseIf Lons(jm, im) = 1 Then
Line9a1k:     '=========================
              hxa = 9999997
              hxat = 9999997
              If CT(jm - 1, im) <> 2 Then
                 hxa = hx(r / 2 + i - L)
                 hxat = hx(i - L)
               Else
                 hxa = 9999992
                 hxat = 9999992
              End If
              If SidMat(jm, im, 1, 1) <> sThreshT(UMAT(jm - 1, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm - 1, im), hxat) Then
      KOC = 1
      Exit Sub
              End If
              If SidMat(jm, im, 2, 1) <> sThreshQ(UMA(jm - 1, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm - 1, im), hxa) Then
      KOC = 1
      Exit Sub
              End If
         End If 'Lons
        End If
       If side21a(jm, im) + side22a(jm, im) + side21t(jm, im) + side22t(jm, im) > 0 Then
       If Lons(jm, im) = 1 Then
        If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
```

```
            If SidMat(jm, im, 1, 2) <> sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i)) Then
      KOC = 1
      Exit Sub
            End If
            If SidMat(jm, im, 2, 2) <> sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2)) Then
      KOC = 1
      Exit Sub
            End If
         ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
          GoTo Line9a2k
        End If
        ElseIf Lons(jm, im) = 2 Then
Line9a2k:     '==========================
              hxb = 9999997
              hxbt = 9999997
              If CT(jm, im - 1) <> 2 Then
                hxb = hx(r / 2 + i - 1)
                hxbt = hx(i - 1)
               Else
                hxb = 9999992
                hxbt = 9999992
              End If
            If SidMat(jm, im, 1, 2) <> sThreshT(UMAT(jm, im - 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im - 1), hxbt) Then
      KOC = 1
      Exit Sub
            End If
            If SidMat(jm, im, 2, 2) <> sThreshQ(UMA(jm, im - 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im - 1), hxb) Then
      KOC = 1
      Exit Sub
            End If
         End If 'Lons
         End If
       If side31a(jm, im) + side32a(jm, im) + side31t(jm, im) + side32t(jm, im) > 0 Then
       If Lons(jm, im) = 2 Then
        If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
            If SidMat(jm, im, 1, 3) <> sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i)) Then
      KOC = 1
      Exit Sub
            End If
            If SidMat(jm, im, 2, 3) <> sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2)) Then
      KOC = 1
      Exit Sub
            End If
         ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
          GoTo Line9a3k
        End If
        ElseIf Lons(jm, im) = 1 Then
Line9a3k:     '==========================
              hxc = 9999997
              hxct = 9999997
              If CT(jm + 1, im) <> 2 Then
                hxc = hx(r / 2 + i + L)
                hxct = hx(i + L)
               Else
                hxc = 9999992
                hxct = 9999992
```

```
              End If
             If SidMat(jm, im, 1, 3) <> sThreshT(UMAT(jm + 1, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm + 1, im), hxct) Then
       KOC = 1
       Exit Sub
             End If
             If SidMat(jm, im, 2, 3) <> sThreshQ(UMA(jm + 1, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm + 1, im), hxc) Then
       KOC = 1
       Exit Sub
             End If
          End If 'Lons
         End If
        If side41a(jm, im) + side42a(jm, im) + side41t(jm, im) + side42t(jm, im) > 0 Then
        If Lons(jm, im) = 1 Then
         If widpp(jm, im) + 2 * bppb(jm, im) < widp(jm, im) Then
              If SidMat(jm, im, 1, 4) <> sThreshT(UMAT(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i)) Then
       KOC = 1
       Exit Sub
             End If
             If SidMat(jm, im, 2, 4) <> sThreshQ(UMA(jm, im), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im), hx(i + r / 2)) Then
       KOC = 1
       Exit Sub
             End If
          ElseIf widpp(jm, im) + 2 * bppb(jm, im) >= widp(jm, im) Then
           GoTo Line9a4k
         End If
         ElseIf Lons(jm, im) = 2 Then
Line9a4k:     '==========================
              hxd = 9999997
              hxdt = 9999997
              If CT(jm, im + 1) <> 2 Then
                hxd = hx(r / 2 + i + 1)
                hxdt = hx(i + 1)
               Else
                hxd = 9999992
                hxdt = 9999992
              End If
             If SidMat(jm, im, 1, 4) <> sThreshT(UMAT(jm, im + 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im + 1), hxdt) Then
       KOC = 1
       Exit Sub
             End If
             If SidMat(jm, im, 2, 4) <> sThreshQ(UMA(jm, im + 1), Bed(jm, im), SHD(jm, im), AqT(jm, im), AqT(jm, im + 1), hxd) Then
       KOC = 1
       Exit Sub
             End If
          End If 'Lons
         End If
 End If
i = i + 1
Next im
Next jm
End Sub
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
```

```
Sub Lon12(Lon1 As Integer, Lon2 As Integer, ph As Integer, jm As Integer, im As Integer, CT() As Integer, L As Integer, m As Integer)
   Lon1 = 0            'Subroutine accounts for neighboring response cells, in case default cell dimension determination of
   Lon2 = 0            'length and width orientation needs to be overridden to account for stream continuity.
   ph = 0
   If jm - 1 > 0 Then
    If CT(jm - 1, im) = 2 Then
     Lon1 = Lon1 + 2
    End If
    If im - 1 > 0 Then
     If CT(jm - 1, im - 1) = 2 Then
      ph = 1
     End If
    End If
    If im + 1 < L + 1 Then
     If CT(jm - 1, im + 1) = 2 Then
      ph = 1
     End If
    End If
    If ph > 0 Then
     Lon1 = Lon1 + 1
     ph = 0
    End If
   End If
   If jm + 1 < m + 1 Then
    If CT(jm + 1, im) = 2 Then
      Lon1 = Lon1 + 2
    End If
    If im - 1 > 0 Then
     If CT(jm + 1, im - 1) = 2 Then
      ph = 1
     End If
    End If
    If im + 1 < L + 1 Then
     If CT(jm + 1, im + 1) = 2 Then
      ph = 1
     End If
    End If
    If ph > 0 Then
     Lon1 = Lon1 + 1
     ph = 0
    End If
   End If
   If im - 1 > 0 Then
    If CT(jm, im - 1) = 2 Then
     Lon2 = Lon2 + 2
    End If
    If jm - 1 > 0 Then
     If CT(jm - 1, im - 1) = 2 Then
      ph = 1
     End If
    End If
    If jm + 1 < m + 1 Then
     If CT(jm + 1, im - 1) = 2 Then
      ph = 1
     End If
    End If
```

```
    If ph > 0 Then
     Lon2 = Lon2 + 1
     ph = 0
    End If
   End If
   If im + 1 < L + 1 Then
    If CT(jm, im + 1) = 2 Then
     Lon2 = Lon2 + 2
    End If
    If jm - 1 > 0 Then
     If CT(jm - 1, im + 1) = 2 Then
      ph = 1
     End If
    End If
    If jm + 1 < m + 1 Then
     If CT(jm + 1, im + 1) = 2 Then
      ph = 1
     End If
    End If
    If ph > 0 Then
     Lon2 = Lon2 + 1
     ph = 0
    End If
   End If
End Sub
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function CTPflagger(CT, CTP, widpp, widp, SHD, AqBb, Bed)
Dim CTPflgr As Integer, xyz As Double
If (SHD - AqBb) > 1 Then
   xyz = (SHD - AqBb)
 Else
   xyz = 1
End If
     If widpp < 0.25 * widp Then
       If (SHD - Bed) < 0.4 * xyz Then
        CTPflgr = 1
       End If
      ElseIf widpp < 0.5 * widp Then
       If (SHD - Bed) < 0.2 * xyz Then
        CTPflgr = 1
       End If
      ElseIf (SHD - Bed) < 0.1 * xyz Then
        CTPflgr = 1
     End If
CTPflagger = CTPflgr
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
Sub CTPload(CTPflg As Integer, m As Integer, L As Integer, CTP() As Integer, CT() As Integer, SB As String, SIT As String, _
            Lons() As Integer, SEB As String, flaw As Integer, msgctp As String, widpp() As Double, widp() As Double, _
            SHD() As Double, AqBctp() As Double, Bed() As Double, BedCP() As Double, zone() As String, msgrza As String, zone1 As Integer)
Dim i As Integer, jm As Integer, im As Integer, rzac As Integer, CTPflgW As Integer
```

```
CTPflg = 0
msgctp = ""
i = 1
For jm = 1 To m
For im = 1 To L
 rzac = 0
 CTP(jm, im) = CT(jm, im)
 BedCP(jm, im) = Bed(jm, im)
 If SB = "Permissive" Then
  If CT(jm, im) = 2 Then
   If widpp(jm, im) < 0.5 * widp(jm, im) Then
    msgctp = "Channel small for cell to have permissive bed; reconfigure. "
    If SIT = "Complete" Then
     CTPflg = CTPflagger(CT(jm, im), CTP(jm, im), widpp(jm, im), widp(jm, im), SHD(jm, im), AqBctp(jm, im), Bed(jm, im))
    End If
   End If
  End If
  If SIT = "Complete" Then
   If CT(jm, im) = 1 Then
     GoTo LineCTPL0
    ElseIf CT(jm, im) = 3 Then
LineCTPL0:
    If jm - 1 > 0 Then
     If CT(jm - 1, im) = 2 Then
      If Lons(jm - 1, im) = 1 Then
        If SEB = "Simple" Then
         GoTo LineCTPL1
        End If
       ElseIf Lons(jm - 1, im) = 2 Then
        If widpp(jm - 1, im) = widp(jm - 1, im) Then
LineCTPL1:
         If CT(jm, im) = 1 Then
         CTPflgW = 1
         Else
         rzac = rzac + 1
         CTP(jm, im) = 2
         BedCP(jm, im) = Bed(jm - 1, im)
         zone(jm, im) = zone(jm - 1, im)
         CTPflg = CTPflagger(CT(jm, im), CTP(jm, im), widpp(jm - 1, im), widp(jm - 1, im), SHD(jm - 1, im), AqBctp(jm - 1, im), Bed(jm - 1, im))
         End If
        End If
      End If
     End If
    End If
    If im - 1 > 0 Then
     If CT(jm, im - 1) = 2 Then
      If Lons(jm, im - 1) = 1 Then
        If widpp(jm, im - 1) = widp(jm, im - 1) Then
          GoTo LineCTPL2
        End If
       ElseIf Lons(jm, im - 1) = 2 Then
        If SEB = "Simple" Then
LineCTPL2:
         If CT(jm, im) = 1 Then
         CTPflgW = 1
         Else
```

```
          rzac = rzac + 1
          CTP(jm, im) = 2
          BedCP(jm, im) = Bed(jm, im - 1)
          If rzac > 1 Then
            If zone(jm, im) <> zone(jm, im - 1) Then
             msgrza = "21"
            End If
           ElseIf rzac <= 1 Then
            zone(jm, im) = zone(jm, im - 1)
          End If
          CTPflg = CTPflagger(CT(jm, im), CTP(jm, im), widpp(jm, im - 1), widp(jm, im - 1), SHD(jm, im - 1), AqBctp(jm, im - 1), Bed(jm, im - 1))
          End If
         End If
       End If
      End If
     End If
     If jm + 1 < m + 1 Then
      If CT(jm + 1, im) = 2 Then
       If Lons(jm + 1, im) = 1 Then
         If SEB = "Simple" Then
          GoTo LineCTPL3
         End If
        ElseIf Lons(jm + 1, im) = 2 Then
         If widpp(jm + 1, im) = widp(jm + 1, im) Then
LineCTPL3:
          If CT(jm, im) = 1 Then
          CTPflgW = 1
          Else
          rzac = rzac + 1
          CTP(jm, im) = 2
          BedCP(jm, im) = Bed(jm + 1, im)
          If rzac > 1 Then
            If zone(jm, im) <> zone(jm + 1, im) Then
             msgrza = "21"
            End If
           ElseIf rzac <= 1 Then
            zone(jm, im) = zone(jm + 1, im)
          End If
          CTPflg = CTPflagger(CT(jm, im), CTP(jm, im), widpp(jm + 1, im), widp(jm + 1, im), SHD(jm + 1, im), AqBctp(jm + 1, im), Bed(jm + 1, im))
          End If
         End If
       End If
      End If
     End If
     If im + 1 < L + 1 Then
      If CT(jm, im + 1) = 2 Then
       If Lons(jm, im + 1) = 1 Then
         If widpp(jm, im + 1) = widp(jm, im + 1) Then
          GoTo LineCTPL4
         End If
        ElseIf Lons(jm, im + 1) = 2 Then
         If SEB = "Simple" Then
LineCTPL4:
          If CT(jm, im) = 1 Then
          CTPflgW = 1
          Else
```

```
         rzac = rzac + 1
         CTP(jm, im) = 2
         BedCP(jm, im) = Bed(jm, im + 1)
         If rzac > 1 Then
           If zone(jm, im) <> zone(jm, im + 1) Then
            msgrza = "21"
           End If
          ElseIf rzac <= 1 Then
           zone(jm, im) = zone(jm, im + 1)
         End If
         CTPflg = CTPflagger(CT(jm, im), CTP(jm, im), widpp(jm, im + 1), widp(jm, im + 1), SHD(jm, im + 1), AqBctp(jm, im + 1), Bed(jm, im + 1))
         End If
        End If
      End If
     End If
    End If
   End If
  End If
 End If
 If msgrza = "21" Then
   msgrza = "Response zone ambiguity due to proximity of permissive banks! "
 End If
i = i + 1
Next im
Next jm
If CTPflgW = 1 Then
 msgctp = "Impulse cell next to permissive channel side; reconfigure. "
ElseIf CTPflg = 1 Then
 msgctp = "Channel small for cell to have permissive sides; reconfigure. "
ElseIf SB = "Permissive" Then
 If SIT = "Complete" Then
  msgctp = ""
 End If
End If
If msgctp <> "" Then
 flaw = 4
End If
End Sub
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function GGG13U(kva, ATP, AqT, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy) 'Function calculates vertical flux
Dim GGGG As Double                                                                                                  'coefficient for bank-overlap-
Dim Amm As Double                                                                                                   'affected non-stream cells without
Dim A0 As Double                                                                                                    'head loss in aquifer.
Dim A1 As Double, A5a As Double, A5b As Double
Dim A2 As Double, A6a As Double, A6b As Double
Dim A3 As Double, A7a As Double, A7b As Double
Dim A4 As Double, A8a As Double, A8b As Double
    A1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dy, dx, AqT + ATP, Beds1)
      If Beds1 < Beds2 Then
        A5a = A5Afun(ovp1, ovw1, ovp2, ovw2, dy, dx, Beds1, Beds2, AqT + ATP)
        A5b = A5Bfun(ovp1, ovw1, ovp2, ovw2, dy, dx, Beds1, Beds2, AqT + ATP)
       ElseIf Beds2 <= Beds1 Then
        A5a = A5Afun(ovp2, ovw2, ovp1, ovw1, dx, dy, Beds2, Beds1, AqT + ATP)
        A5b = A5Bfun(ovp2, ovw2, ovp1, ovw1, dx, dy, Beds2, Beds1, AqT + ATP)
```

```
      End If
    A2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dx, dy, AqT + ATP, Beds2)
      If Beds2 < Beds3 Then
        A6a = A5Afun(ovp2, ovw2, ovp3, ovw3, dx, dy, Beds2, Beds3, AqT + ATP)
        A6b = A5Bfun(ovp2, ovw2, ovp3, ovw3, dx, dy, Beds2, Beds3, AqT + ATP)
       ElseIf Beds3 <= Beds2 Then
        A6a = A5Afun(ovp3, ovw3, ovp2, ovw2, dy, dx, Beds3, Beds2, AqT + ATP)
        A6b = A5Bfun(ovp3, ovw3, ovp2, ovw2, dy, dx, Beds3, Beds2, AqT + ATP)
      End If
    A3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dy, dx, AqT + ATP, Beds3)
      If Beds3 < Beds4 Then
        A7a = A5Afun(ovp3, ovw3, ovp4, ovw4, dy, dx, Beds3, Beds4, AqT + ATP)
        A7b = A5Bfun(ovp3, ovw3, ovp4, ovw4, dy, dx, Beds3, Beds4, AqT + ATP)
       ElseIf Beds4 <= Beds3 Then
        A7a = A5Afun(ovp4, ovw4, ovp3, ovw3, dx, dy, Beds4, Beds3, AqT + ATP)
        A7b = A5Bfun(ovp4, ovw4, ovp3, ovw3, dx, dy, Beds4, Beds3, AqT + ATP)
      End If
    A4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dx, dy, AqT + ATP, Beds4)
      If Beds4 < Beds1 Then
        A8a = A5Afun(ovp4, ovw4, ovp1, ovw1, dx, dy, Beds4, Beds1, AqT + ATP)
        A8b = A5Bfun(ovp4, ovw4, ovp1, ovw1, dx, dy, Beds4, Beds1, AqT + ATP)
       ElseIf Beds1 <= Beds4 Then
        A8a = A5Afun(ovp1, ovw1, ovp4, ovw4, dy, dx, Beds1, Beds4, AqT + ATP)
        A8b = A5Bfun(ovp1, ovw1, ovp4, ovw4, dy, dx, Beds1, Beds4, AqT + ATP)
      End If
If Beds1 < AqT + ATP Then
 If Beds1 > AqT Then
    GGGG = GGGG + kva * A1 / (Beds1 - AqT)
    Amm = Amm + A1
  If Beds1 < Beds2 Then
    GGGG = GGGG + kva * A5a / (Beds1 - AqT)
    Amm = Amm + A5a
   Else '1>2.
    GGGG = GGGG + kva * A5b / (Beds1 - AqT)
    Amm = Amm + A5b
  End If '1v2.
  If Beds4 < Beds1 Then
    GGGG = GGGG + kva * A8b / (Beds1 - AqT)
    Amm = Amm + A8b
   Else '4>1.
    GGGG = GGGG + kva * A8a / (Beds1 - AqT)
    Amm = Amm + A8a
  End If '4v1.
  Else 'Beds<AqT
    Amm = Amm + A1
  If Beds1 < Beds2 Then
    Amm = Amm + A5a
   Else '1>2.
    Amm = Amm + A5b
  End If '1v2.
  If Beds4 < Beds1 Then
    Amm = Amm + A8b
   Else '4>1.
    Amm = Amm + A8a
  End If '4v1.
 End If 'Beds v AqT.
```

```
End If
If Beds2 < AqT + ATP Then
 If Beds2 > AqT Then
    GGGG = GGGG + kva * A2 / (Beds2 - AqT)
    Amm = Amm + A2
  If Beds2 < Beds3 Then
    GGGG = GGGG + kva * A6a / (Beds2 - AqT)
    Amm = Amm + A6a
   Else
    GGGG = GGGG + kva * A6b / (Beds2 - AqT)
    Amm = Amm + A6b
  End If
  If Beds1 < Beds2 Then
    GGGG = GGGG + kva * A5b / (Beds2 - AqT)
    Amm = Amm + A5b
   Else
    GGGG = GGGG + kva * A5a / (Beds2 - AqT)
    Amm = Amm + A5a
  End If
  Else
    Amm = Amm + A2
  If Beds2 < Beds3 Then
    Amm = Amm + A6a
   Else
    Amm = Amm + A6b
  End If
  If Beds1 < Beds2 Then
    Amm = Amm + A5b
   Else
    Amm = Amm + A5a
  End If
 End If
End If
If Beds3 < AqT + ATP Then
 If Beds3 > AqT Then
    GGGG = GGGG + kva * A3 / (Beds3 - AqT)
    Amm = Amm + A3
  If Beds3 < Beds4 Then
    GGGG = GGGG + kva * A7a / (Beds3 - AqT)
    Amm = Amm + A7a
   Else
    GGGG = GGGG + kva * A7b / (Beds3 - AqT)
    Amm = Amm + A7b
  End If
  If Beds2 < Beds3 Then
    GGGG = GGGG + kva * A6b / (Beds3 - AqT)
    Amm = Amm + A6b
   Else
    GGGG = GGGG + kva * A6a / (Beds3 - AqT)
    Amm = Amm + A6a
  End If
  Else
    Amm = Amm + A3
  If Beds3 < Beds4 Then
    Amm = Amm + A7a
   Else
```

```
    Amm = Amm + A7b
  End If
  If Beds2 < Beds3 Then
    Amm = Amm + A6b
   Else
    Amm = Amm + A6a
  End If
 End If
End If
If Beds4 < AqT + ATP Then
 If Beds4 > AqT Then
    GGGG = GGGG + kva * A4 / (Beds4 - AqT)
    Amm = Amm + A4
  If Beds4 < Beds1 Then
    GGGG = GGGG + kva * A8a / (Beds4 - AqT)
    Amm = Amm + A8a
   Else
    GGGG = GGGG + kva * A8b / (Beds4 - AqT)
    Amm = Amm + A8b
  End If
  If Beds3 < Beds4 Then
    GGGG = GGGG + kva * A7b / (Beds4 - AqT)
    Amm = Amm + A7b
   Else
    GGGG = GGGG + kva * A7a / (Beds4 - AqT)
    Amm = Amm + A7a
  End If
  Else
    Amm = Amm + A4
  If Beds4 < Beds1 Then
    Amm = Amm + A8a
   Else
    Amm = Amm + A8b
  End If
  If Beds3 < Beds4 Then
    Amm = Amm + A7b
   Else
    Amm = Amm + A7a
  End If
 End If
End If
A0 = dy * dx - Amm
GGGG = GGGG + kva * A0 / ATP
GGG13U = GGGG
End Function
'*****************************************************************************************************************************************************

'*****************************************************************************************************************************************************
Function GGG13F(AvTp, kva, kvqp, ATP, AQP, AqT, ovp1, ovw1, Beds1, ovp2, ovw2, Beds2, ovp3, ovw3, Beds3, ovp4, ovw4, Beds4, dx, dy, vfhd) 'Function
calculates vertical flux
Dim GGGG As Double                                                                                                    'coefficient for bank-
overlap-
Dim Atmm As Double                                                                                                     'affected non-stream
cells with
Dim A0 As Double                                                                                                      'head loss in aquifer.
Dim A1 As Double, A5a As Double, A5b As Double, A5c As Double
```

```
Dim A2 As Double, A6a As Double, A6b As Double, A6c As Double
Dim A3 As Double, A7a As Double, A7b As Double, A7c As Double
Dim A4 As Double, A8a As Double, A8b As Double, A8c As Double
Dim Aqadj As Double, Vqmm As Double
Dim FPM As Double
If vfhd = "All" Then
      FPM = 1
     Else
      FPM = 0.5
End If
Atmm = 0
Vqmm = 0
GGGG = 0
    A1 = A1fun(ovp1, ovw1, ovp2, ovw2, ovp4, ovw4, dy, dx, AqT + ATP, Beds1)
      If Beds1 < Beds2 Then
        A5a = A5Afun(ovp1, ovw1, ovp2, ovw2, dy, dx, Beds1, Beds2, AqT + ATP)
        A5b = A5Bfun(ovp1, ovw1, ovp2, ovw2, dy, dx, Beds1, Beds2, AqT + ATP)
        A5c = A5Cfun(ovp1, ovw1, ovp2, ovw2, dy, dx, Beds1, Beds2, AqT + ATP)
       ElseIf Beds2 <= Beds1 Then
        A5a = A5Afun(ovp2, ovw2, ovp1, ovw1, dx, dy, Beds2, Beds1, AqT + ATP)
        A5b = A5Bfun(ovp2, ovw2, ovp1, ovw1, dx, dy, Beds2, Beds1, AqT + ATP)
        A5c = A5Cfun(ovp2, ovw2, ovp1, ovw1, dx, dy, Beds2, Beds1, AqT + ATP)
      End If
    A2 = A1fun(ovp2, ovw2, ovp3, ovw3, ovp1, ovw1, dx, dy, AqT + ATP, Beds2)
      If Beds2 < Beds3 Then
        A6a = A5Afun(ovp2, ovw2, ovp3, ovw3, dx, dy, Beds2, Beds3, AqT + ATP)
        A6b = A5Bfun(ovp2, ovw2, ovp3, ovw3, dx, dy, Beds2, Beds3, AqT + ATP)
        A6c = A5Cfun(ovp2, ovw2, ovp3, ovw3, dx, dy, Beds2, Beds3, AqT + ATP)
       ElseIf Beds3 <= Beds2 Then
        A6a = A5Afun(ovp3, ovw3, ovp2, ovw2, dy, dx, Beds3, Beds2, AqT + ATP)
        A6b = A5Bfun(ovp3, ovw3, ovp2, ovw2, dy, dx, Beds3, Beds2, AqT + ATP)
        A6c = A5Cfun(ovp3, ovw3, ovp2, ovw2, dy, dx, Beds3, Beds2, AqT + ATP)
      End If
    A3 = A1fun(ovp3, ovw3, ovp4, ovw4, ovp2, ovw2, dy, dx, AqT + ATP, Beds3)
      If Beds3 < Beds4 Then
        A7a = A5Afun(ovp3, ovw3, ovp4, ovw4, dy, dx, Beds3, Beds4, AqT + ATP)
        A7b = A5Bfun(ovp3, ovw3, ovp4, ovw4, dy, dx, Beds3, Beds4, AqT + ATP)
        A7c = A5Cfun(ovp3, ovw3, ovp4, ovw4, dy, dx, Beds3, Beds4, AqT + ATP)
       ElseIf Beds4 <= Beds3 Then
        A7a = A5Afun(ovp4, ovw4, ovp3, ovw3, dx, dy, Beds4, Beds3, AqT + ATP)
        A7b = A5Bfun(ovp4, ovw4, ovp3, ovw3, dx, dy, Beds4, Beds3, AqT + ATP)
        A7c = A5Cfun(ovp4, ovw4, ovp3, ovw3, dx, dy, Beds4, Beds3, AqT + ATP)
      End If
    A4 = A1fun(ovp4, ovw4, ovp1, ovw1, ovp3, ovw3, dx, dy, AqT + ATP, Beds4)
      If Beds4 < Beds1 Then
        A8a = A5Afun(ovp4, ovw4, ovp1, ovw1, dx, dy, Beds4, Beds1, AqT + ATP)
        A8b = A5Bfun(ovp4, ovw4, ovp1, ovw1, dx, dy, Beds4, Beds1, AqT + ATP)
        A8c = A5Cfun(ovp4, ovw4, ovp1, ovw1, dx, dy, Beds4, Beds1, AqT + ATP)
       ElseIf Beds1 <= Beds4 Then
        A8a = A5Afun(ovp1, ovw1, ovp4, ovw4, dy, dx, Beds1, Beds4, AqT + ATP)
        A8b = A5Bfun(ovp1, ovw1, ovp4, ovw4, dy, dx, Beds1, Beds4, AqT + ATP)
        A8c = A5Cfun(ovp1, ovw1, ovp4, ovw4, dy, dx, Beds1, Beds4, AqT + ATP)
      End If
If Beds1 < AqT + ATP Then
 If Beds1 > AqT Then
    If A1 > 0 Then
```

```
    GGGG = GGGG + khbave(AvTp, kva, A1, kvqp, A1, FPM * (Beds1 - AqT), AQP) / (FPM * (Beds1 - AqT) + AQP)
    Atmm = Atmm + A1
    Vqmm = Vqmm + A1 * AQP
    End If
  If Beds1 < Beds2 Then
    If A5a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A5a, kvqp, A5a, FPM * (Beds1 - AqT), AQP) / (FPM * (Beds1 - AqT) + AQP)
    Atmm = Atmm + A5a
    Vqmm = Vqmm + A5a * AQP
    End If
   Else '1>2.
    If A5b > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A5b, kvqp, A5b, FPM * (Beds1 - AqT), AQP) / (FPM * (Beds1 - AqT) + AQP)
    Atmm = Atmm + A5b
    Vqmm = Vqmm + A5b * AQP
    End If
  End If '1v2.
  If Beds4 < Beds1 Then
    If A8b > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A8b, kvqp, A8b, FPM * (Beds1 - AqT), AQP) / (FPM * (Beds1 - AqT) + AQP)
    Atmm = Atmm + A8b
    Vqmm = Vqmm + A8b * AQP
    End If
   Else '4>1.
    If A8a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A8a, kvqp, A8a, FPM * (Beds1 - AqT), AQP) / (FPM * (Beds1 - AqT) + AQP)
    Atmm = Atmm + A8a
    Vqmm = Vqmm + A8a * AQP
    End If
  End If '4v1.
  Else 'Beds<AqT
    Atmm = Atmm + A1
    Vqmm = Vqmm + A1 * (AqT - Beds1)
  If Beds1 < Beds2 Then
    Atmm = Atmm + A5a
    Vqmm = Vqmm + A5a * (AqT - Beds1)
   Else '1>2.
    Atmm = Atmm + A5b
    Vqmm = Vqmm + A5b * (AqT - Beds1)
  End If '1v2.
  If Beds4 < Beds1 Then
    Atmm = Atmm + A8b
    Vqmm = Vqmm + A8b * (AqT - Beds1)
   Else '4>1.
    Atmm = Atmm + A8a
    Vqmm = Vqmm + A8a * (AqT - Beds1)
  End If '4v1.
 End If 'Beds v AqT.
End If
If Beds2 < AqT + ATP Then
 If Beds2 > AqT Then
    If A2 > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A2, kvqp, A2, FPM * (Beds2 - AqT), AQP) / (FPM * (Beds2 - AqT) + AQP)
    Atmm = Atmm + A2
    Vqmm = Vqmm + A2 * AQP
    End If
```

```
  If Beds2 < Beds3 Then
    If A6a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A6a, kvqp, A6a, FPM * (Beds2 - AqT), AQP) / (FPM * (Beds2 - AqT) + AQP)
    Atmm = Atmm + A6a
    Vqmm = Vqmm + A6a * AQP
    End If
   Else
    If A6b > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A6b, kvqp, A6b, FPM * (Beds2 - AqT), AQP) / (FPM * (Beds2 - AqT) + AQP)
    Atmm = Atmm + A6b
    Vqmm = Vqmm + A6b * AQP
    End If
  End If
  If Beds1 < Beds2 Then
    If A5b > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A5b, kvqp, A5b, FPM * (Beds2 - AqT), AQP) / (FPM * (Beds2 - AqT) + AQP)
    Atmm = Atmm + A5b
    Vqmm = Vqmm + A5b * AQP
    End If
   Else
    If A5a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A5a, kvqp, A5a, FPM * (Beds2 - AqT), AQP) / (FPM * (Beds2 - AqT) + AQP)
    Atmm = Atmm + A5a
    Vqmm = Vqmm + A5a * AQP
    End If
  End If
  Else
    Atmm = Atmm + A2
    Vqmm = Vqmm + A2 * (AqT - Beds2)
  If Beds2 < Beds3 Then
    Atmm = Atmm + A6a
    Vqmm = Vqmm + A6a * (AqT - Beds2)
   Else
    Atmm = Atmm + A6b
    Vqmm = Vqmm + A6b * (AqT - Beds2)
  End If
  If Beds1 < Beds2 Then
    Atmm = Atmm + A5b
    Vqmm = Vqmm + A5b * (AqT - Beds2)
   Else
    Atmm = Atmm + A5a
    Vqmm = Vqmm + A5a * (AqT - Beds2)
  End If
 End If
End If
If Beds3 < AqT + ATP Then
 If Beds3 > AqT Then
    If A3 > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A3, kvqp, A3, FPM * (Beds3 - AqT), AQP) / (FPM * (Beds3 - AqT) + AQP)
    Atmm = Atmm + A3
    Vqmm = Vqmm + A3 * AQP
    End If
  If Beds3 < Beds4 Then
    If A7a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A7a, kvqp, A7a, FPM * (Beds3 - AqT), AQP) / (FPM * (Beds3 - AqT) + AQP)
    Atmm = Atmm + A7a
```

```
    Vqmm = Vqmm + A7a * AQP
    End If
   Else
    If A7b > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A7b, kvqp, A7b, FPM * (Beds3 - AqT), AQP) / (FPM * (Beds3 - AqT) + AQP)
    Atmm = Atmm + A7b
    Vqmm = Vqmm + A7b * AQP
    End If
  End If
  If Beds2 < Beds3 Then
    If A6b > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A6b, kvqp, A6b, FPM * (Beds3 - AqT), AQP) / (FPM * (Beds3 - AqT) + AQP)
    Atmm = Atmm + A6b
    Vqmm = Vqmm + A6b * AQP
    End If
   Else
    If A6a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A6a, kvqp, A6a, FPM * (Beds3 - AqT), AQP) / (FPM * (Beds3 - AqT) + AQP)
    Atmm = Atmm + A6a
    Vqmm = Vqmm + A6a * AQP
    End If
  End If
  Else
    Atmm = Atmm + A3
    Vqmm = Vqmm + A3 * (AqT - Beds3)
  If Beds3 < Beds4 Then
    Atmm = Atmm + A7a
    Vqmm = Vqmm + A7a * (AqT - Beds3)
   Else
    Atmm = Atmm + A7b
    Vqmm = Vqmm + A7b * (AqT - Beds3)
  End If
  If Beds2 < Beds3 Then
    Atmm = Atmm + A6b
    Vqmm = Vqmm + A6b * (AqT - Beds3)
   Else
    Atmm = Atmm + A6a
    Vqmm = Vqmm + A6a * (AqT - Beds3)
  End If
 End If
End If
If Beds4 < AqT + ATP Then
 If Beds4 > AqT Then
    If A4 > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A4, kvqp, A4, FPM * (Beds4 - AqT), AQP) / (FPM * (Beds4 - AqT) + AQP)
    Atmm = Atmm + A4
    Vqmm = Vqmm + A4 * AQP
    End If
  If Beds4 < Beds1 Then
    If A8a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A8a, kvqp, A8a, FPM * (Beds4 - AqT), AQP) / (FPM * (Beds4 - AqT) + AQP)
    Atmm = Atmm + A8a
    Vqmm = Vqmm + A8a * AQP
    End If
   Else
    If A8b > 0 Then
```

```
    GGGG = GGGG + khbave(AvTp, kva, A8b, kvqp, A8b, FPM * (Beds4 - AqT), AQP) / (FPM * (Beds4 - AqT) + AQP)
    Atmm = Atmm + A8b
    Vqmm = Vqmm + A8b * AQP
    End If
  End If
  If Beds3 < Beds4 Then
    If A7b > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A7b, kvqp, A7b, FPM * (Beds4 - AqT), AQP) / (FPM * (Beds4 - AqT) + AQP)
    Atmm = Atmm + A7b
    Vqmm = Vqmm + A7b * AQP
    End If
   Else
    If A7a > 0 Then
    GGGG = GGGG + khbave(AvTp, kva, A7a, kvqp, A7a, FPM * (Beds4 - AqT), AQP) / (FPM * (Beds4 - AqT) + AQP)
    Atmm = Atmm + A7a
    Vqmm = Vqmm + A7a * AQP
    End If
  End If
  Else
    Atmm = Atmm + A4
    Vqmm = Vqmm + A4 * (AqT - Beds4)
  If Beds4 < Beds1 Then
    Atmm = Atmm + A8a
    Vqmm = Vqmm + A8a * (AqT - Beds4)
   Else
    Atmm = Atmm + A8b
    Vqmm = Vqmm + A8b * (AqT - Beds4)
  End If
  If Beds3 < Beds4 Then
    Atmm = Atmm + A7b
    Vqmm = Vqmm + A7b * (AqT - Beds4)
   Else
    Atmm = Atmm + A7a
    Vqmm = Vqmm + A7a * (AqT - Beds4)
  End If
 End If
End If
If A5c > 0 Then
GGGG = GGGG + khbave(AvTp, kva, A5c, kvqp, A5c, ATP, AQP) / (ATP + AQP)
Atmm = Atmm + A5c
Vqmm = Vqmm + A5c * AQP
End If
If A6c > 0 Then
GGGG = GGGG + khbave(AvTp, kva, A6c, kvqp, A6c, ATP, AQP) / (ATP + AQP)
Atmm = Atmm + A6c
Vqmm = Vqmm + A6c * AQP
End If
If A7c > 0 Then
GGGG = GGGG + khbave(AvTp, kva, A7c, kvqp, A7c, ATP, AQP) / (ATP + AQP)
Atmm = Atmm + A7c
Vqmm = Vqmm + A7c * AQP
End If
If A8c > 0 Then
GGGG = GGGG + khbave(AvTp, kva, A8c, kvqp, A8c, ATP, AQP) / (ATP + AQP)
Atmm = Atmm + A8c
Vqmm = Vqmm + A8c * AQP
```

```
End If
A0 = dy * dx - Atmm
Aqadj = (dy * dx * AQP - Vqmm) / AQP
GGGG = GGGG + khbave(AvTp, kva, A0, kvqp, Aqadj, ATP, AQP) / (ATP + AQP)
GGG13F = GGGG
End Function
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************
'*****************************************************************************************************************************************************


THE END!
```