Appendix C
Visual Basic Code for HSPF Post Processor
Main Form (frmFloodDischarges)

Option Base 1
Public BaseFile As String, ScenarioFile As String, CurrentFile As String, HSIFile As String
Dim TempFile As String
Dim CompareFile As String
Dim Yr() As Integer
Dim Mo() As Integer
Dim Dy() As Integer
Dim Hr() As Integer
Dim Mn() As Integer
Dim q() As Single
Dim TotalLines As Long
Dim BaseYr() As Integer
Dim BaseMo() As Integer
Dim BaseDy() As Integer
Dim BaseHr() As Integer
Dim BaseMn() As Integer
Dim Baseq() As Single
Dim BaseTotalLines As Long
Dim ChartArray() As Double
Dim qStormHydrograph() As Single
Dim BaseqStormHydrograph() As Single
Dim BaseHydrographCount() As Single
Dim HydrographCount() As Single
Dim StartEnd() As Long
Dim Skew() As Single
Dim StdDev() As Single
'Dim PeakDate() As Integer '****
'Dim Peak() As Single '****
'Dim PeakReader As String '****
Dim ChartArrayFreq() As Double
Dim ChartArrayFreqShift() As Double
Dim ChartArrayStats() As Double
Private Sub StdNorm(z, Tr)
Dim u As Double
Dim q As Double
Dim p As Double

p = 1 / Tr
If p < 0.5 Then
q = p
Else
    q = 1 - p
End If

u = -Log(2 * q)
z = -((u ^ 2 * ((4 * u + 100) * u + 205)) / ((2 * u + 56) * u + 192) * u + 131)) ^ 0.5

If p < 0.5 Then
    z = -z
End If

Private Sub cmdCompare_Click()
    Call Compare
End Sub

Private Sub Form_Load()
    Percent = 0.1
    TimeLimit = 10
    Gm = 0.5
    ProgressBar1.Value = 0
    CommonDialog1.Filter = "All Files (*.*)|*.*|HSPF Files (*.PLT)|*.PLT|CSV Files (*.CSV)|*.CSV"
End Sub

Private Sub mnuAboutDescription_Click()
    MsgBox "HSPF Post Processor was created by Brendan Lockard." & vbCrLf & "Copyright 2002." & vbCrLf & _
    vbCrLf & "User accepts all risks with using this software." & vbCrLf & _
    vbCrLf & "Results obtained from this software are in no way guaranteed or endorsed." & vbCrLf & _
    vbCrLf & "Any and all damages incurred are the responsibility of the user." & vbCrLf & _
    vbCrLf & "No support is available."
End Sub

Private Sub mnuBase_Click()
    Dim i As Long
    Dim j As Long
    Dim Fileline As String

    CommonDialog1.Filter = "All Files (*.*)|*.*|HSPF Files (*.PLT)|*.PLT|Comma Separated Values (*.CSV)|*.CSV"
CommonDialog1.FilterIndex = 2
CommonDialog1.FileName = ""
CommonDialog1.ShowDialog
If CommonDialog1.FileName <> "" Then
    CurrentFile = CommonDialog1.FileName
    BaseFile = CurrentFile
    txtBaseFile.Text = CommonDialog1.FileName
    Open CurrentFile For Input As #1

    ProgressBar1.Value = 1 / 10
    For i = 1 To 26
        Line Input #1, Fileline  
    Next i

    j = 0
    Do While Not EOF(1)
        Line Input #1, Fileline
        j = j + 1
    Loop
    Close #1

    ReDim BaseYr(j), BaseMo(j), BaseDy(j), BaseHr(j), BaseMn(j), Baseq(10, j)  
    Open CurrentFile For Input As #1
    For i = 1 To 26
        Line Input #1, Fileline  
    Next i
    For i = 1 To j
        ProgressBar1.Value = (i / j * 100)
        Line Input #1, Fileline

        BaseYr(i) = val(Trim(Mid(Fileline, 6, 5)))
        BaseMo(i) = val(Trim(Mid(Fileline, 12, 2)))
        BaseDy(i) = val(Trim(Mid(Fileline, 15, 2)))
        BaseHr(i) = val(Trim(Mid(Fileline, 18, 2)))
        BaseMn(i) = val(Trim(Mid(Fileline, 21, 2)))
        Baseq(1, i) = val(Trim(Mid(Fileline, 25, 14)))
Baseq(2, i) = val(Trim(Mid(Fileline, 38, 14)))
Baseq(3, i) = val(Trim(Mid(Fileline, 52, 14)))
Baseq(4, i) = val(Trim(Mid(Fileline, 66, 14)))
Baseq(5, i) = val(Trim(Mid(Fileline, 80, 14)))
Baseq(6, i) = val(Trim(Mid(Fileline, 94, 14)))
Baseq(7, i) = val(Trim(Mid(Fileline, 108, 14)))
Baseq(8, i) = val(Trim(Mid(Fileline, 122, 14)))
Baseq(9, i) = val(Trim(Mid(Fileline, 136, 14)))
Baseq(10, i) = val(Trim(Mid(Fileline, 150, 14)))
Next i
BaseTotalLines = j
Close #1
ProgressBar1.Value = 0
Call BaseFlood
End If
End Sub
Private Sub BaseFlood()

' Flood Discharge Program
'
' Created 7/02/2001 by Brendan Lockard
'
' This program calculates the flood discharge for selected return periods using
' Log Pearson Type III Analysis
'
' Define Variables
Dim Qtemp As Single 'Current Annual Peak
Dim Yrtemp As Integer
Dim Motemp As Integer
Dim Dytemp As Integer
Dim Hrtemp As Integer
Dim Mntemp As Integer
Dim Peak() As Single 'Peak annual flowrates
Dim PeakDate() As Integer
Dim LogPeak() As Single 'Log base 10 Peak annual flowrates
Dim M As Single 'Mean of LogPeak
Dim S As Single 'Standard Deviation of LogPeak
Dim G As Single 'Skew of LogPeak
Dim Gw As Single 'Weighted Skew Coefficient
Dim Vs As Single 'Gw computation parameter
Dim Vm As Single 'Gw computation parameter
'Dim Gm As Single 'Gw computation parameter
Dim a As Single 'Gw computation parameter
Dim b As Single 'Gw computation parameter
Dim Omega As Single 'Gw computation parameter
Dim SumDiffSquared As Double '(Log10 (Q) - M)^2
Dim SumDiffCubed As Double '(Log10 (Q) - M)^3
Dim n As Double 'Counter
Dim TotalQ As Double 'Sum of Annual Peaks
Dim smalla As Double ' a parameter
Dim smallb As Double ' b parameter
Dim smallc As Double ' c parameter
Dim i As Long
Dim j As Long
Dim k As Long
Dim LowStatus As String
Dim KT() As Double
Dim QT() As Double
Dim T As Integer
Dim Tr() As Single
Dim z() As Double

'Fill Annual Peaks Array
j = 0 'Annual Peaks Array Counter
Qtemp = 0
i = 1 'Total Dataset Array Counter
Do While i < BaseTotalLines + 1
    If BaseMo(i) = 10 And BaseDy(i) = 1 And BaseHr(i) = 1 And BaseMn(i) = 0 Then
        Qtemp = 0
        Motemp = 0
        Dytemp = 0
        Hrtemp = 0
        Mntemp = 0
    End If
    i = i + 1
End Do
j = j + 1
ReDim Preserve PeakDate(6, j), Peak(j)  'Resize Array to Unknown Size
PeakDate(1, j) = BaseYr(i) + 1
End If
If Baseq(10, i) > Qtemp Then
    Qtemp = Baseq(10, i)
    Yrtemp = BaseYr(i)
    Motemp = BaseMo(i)
    Dytemp = BaseDy(i)
    Hrtemp = BaseHr(i)
    Mntemp = BaseMn(i)
End If
PeakDate(2, j) = Yrtemp
PeakDate(3, j) = Motemp
PeakDate(4, j) = Dytemp
PeakDate(5, j) = Hrtemp
PeakDate(6, j) = Mntemp
Peak(j) = Qtemp
i = i + 1
Loop
'Temp Reader
'Else
'j = BaseTotalLines
'End If

TotalQ = 0
ReDim LogPeak(j)  'Resize Array
For k = 1 To j
    LogPeak(k) = Log(Peak(k)) / Log(10)  'Log base 10
    TotalQ = TotalQ + LogPeak(k)
Next k
M = TotalQ / j
For k = 1 To j
    SumDiffSquared = (LogPeak(k) - M) ^ 2 + SumDiffSquared
SumDiffCubed = (LogPeak(k) - M) ^ 3 + SumDiffCubed
Next k

S = (SumDiffSquared / (j - 1)) ^ 0.5
G = j / (j - 1) / (j - 2) * SumDiffCubed / S ^ 3

ReDim Preserve StdDev(2)
ReDim Preserve Skew(2)

StdDev(2) = S
Skew(2) = G

smallb = 4 / G ^ 2
smalla = G * smallb ^ 0.5 / (G * S)
smallc = M - (smallb / smalla)

'Gm = 0.5 '****
Vm = 0.302

If Abs(G) > 0.9 Then
    a = -0.52 + 0.3 * Abs(G)
Else
    a = -0.33 + 0.08 * Abs(G)
End If

If Abs(G) > 1.5 Then
    b = 0.55
Else
    b = 0.94 - 0.26 * Abs(G)
End If

Omega = a - b * (Log(j / 10) / Log(10))
Vs = 10 ^ Omega
Gw = (Vs * Gm + G * Vm) / (Vs + Vm)

ReDim Tr(7), z(7), KT(7), QT(7)
Tr(1) = 100
Tr(2) = 50
Tr(3) = 25
Tr(4) = 10
Tr(5) = 5
Tr(6) = 2
Tr(7) = 1.67 'Bankfull

For i = 1 To 7
   Call StdNorm(z(i), Tr(i))
   KT(i) = 2 / Gw * (1 + Gw * z(i) / 6 - Gw ^ 2 / 36) ^ 3 - 2 / Gw
   QT(i) = 10 ^ (M + KT(i) * S)
Next i
lstBaseOutput.AddItem ""

For i = 1 To 6
   lstBaseOutput.AddItem Tr(i) & " Year Flood Discharge = " & Int(QT(i)) & " cfs"
Next i

ReDim Preserve ChartArray(6, 4)
For i = 1 To 6
   ChartArray(i, 1) = Tr(i)
   ChartArray(i, 2) = QT(i)
Next i
Bankfull = QT(7)
End Sub

Private Sub ScenarioFlood()
   ' Flood Discharge Program
   ' Created 7/02/2001 by Brendan Lockard
   ' This program calculates the flood discharge for selected return periods using
   ' Log Pearson Type III Analysis
   ' Define Variables
Dim Qtemp As Single 'Current Annual Peak
Dim Yrtemp As Integer
Dim Motemp As Integer
Dim Dytemp As Integer
Dim Hrtemp As Integer
Dim Mntemp As Integer
Dim Peak() As Single 'Peak annual flowrates
Dim PeakDate() As Integer
Dim LogPeak() As Single 'Log base 10 Peak annual Flowrates
Dim M As Single 'Mean of LogPeak
Dim S As Single 'Standard Deviation of LogPeak
Dim G As Single 'Skew of LogPeak
Dim Gw As Single 'Weighted Skew Coefficient
Dim Vs As Single 'Gw computation parameter
Dim Vm As Single 'Gw computation parameter
'Dim Gm As Single 'Gw computation parameter
Dim a As Single 'Gw computation parameter
Dim b As Single 'Gw computation parameter
Dim Omega As Single 'Gw computation parameter
Dim SumDiffSquared As Double '(Log10 (Q) - M)^2
Dim SumDiffCubed As Double '(Log10 (Q) - M)^3
Dim n As Double 'Counter
Dim TotalQ As Double 'Sum of Annual Peaks
Dim smalla As Double ' a parameter
Dim smallb As Double ' b parameter
Dim smallc As Double ' c parameter
Dim i As Long
Dim j As Long
Dim k As Long
Dim LowStatus As String
Dim KT() As Double
Dim QT() As Double
Dim T As Integer
Dim Tr() As Single
Dim z() As Double
j = 0 'Annual Peaks Array Counter
Qtemp = 0
i = 1 'Total Dataset Array Counter
Do While i < TotalLines + 1
    If Mo(i) = 10 And Dy(i) = 1 And Hr(i) = 1 And Mn(i) = 0 Then
        Qtemp = 0
        Motemp = 0
        Dytemp = 0
        Hrtemp = 0
        Mntemp = 0
        j = j + 1
        ReDim Preserve PeakDate(6, j), Peak(j) 'Resize Array to Unknown Size
        PeakDate(1, j) = Yr(i) + 1
    End If
    If q(10, i) > Qtemp Then
        Qtemp = q(10, i)
        Yrtemp = Yr(i)
        Motemp = Mo(i)
        Dytemp = Dy(i)
        Hrtemp = Hr(i)
        Mntemp = Mn(i)
    End If
    PeakDate(2, j) = Yrtemp
    PeakDate(3, j) = Motemp
    PeakDate(4, j) = Dytemp
    PeakDate(5, j) = Hrtemp
    PeakDate(6, j) = Mntemp
    Peak(j) = Qtemp
    i = i + 1
Loop

TotalQ = 0
ReDim LogPeak(j) 'Resize Array
For k = 1 To j
LogPeak(k) = Log(Peak(k)) / Log(10) 'Log base 10
    TotalQ = TotalQ + LogPeak(k)
Next k

M = TotalQ / j

For k = 1 To j
    SumDiffSquared = (LogPeak(k) - M) ^ 2 + SumDiffSquared
    SumDiffCubed = (LogPeak(k) - M) ^ 3 + SumDiffCubed
Next k

S = (SumDiffSquared / (j - 1)) ^ 0.5
G = j / (j - 1) / (j - 2) * SumDiffCubed / S ^ 3

ReDim Preserve StdDev(2)
ReDim Preserve Skew(2)

StdDev(1) = S
Skew(1) = G

smallb = 4 / G ^ 2
smalla = G * smallb ^ 0.5 / (G * S)
smallc = M - (smallb / smalla)

Vm = 0.302

If Abs(G) > 0.9 Then
    a = -0.52 + 0.3 * Abs(G)
Else
    a = -0.33 + 0.08 * Abs(G)
End If

If Abs(G) > 1.5 Then
    b = 0.55
Else
    b = 0.94 - 0.26 * Abs(G)
End If
Omega = a - b * (Log(j / 10) / Log(10))
Vs = 10 ^ Omega
Gw = (Vs * Gm + G * Vm) / (Vs + Vm)

ReDim Tr(6), z(6), KT(6), QT(6)
Tr(1) = 100
Tr(2) = 50
Tr(3) = 25
Tr(4) = 10
Tr(5) = 5
Tr(6) = 2

For i = 1 To 6
Call StdNorm(z(i), Tr(i))
KT(i) = 2 / Gw * (1 + Gw * z(i) / 6 - Gw ^ 2 / 36) ^ 3 - 2 / Gw
QT(i) = 10 ^ (M + KT(i) * S)
Next i
lstScenarioOutput.AddItem ""
For i = 1 To 6
lstScenarioOutput.AddItem Tr(i) & " Year Flood Discharge = " & Int(QT(i)) & " cfs"
Next i
ReDim Preserve ChartArray(6, 4)
For i = 1 To 6
    ChartArray(i, 3) = Tr(i)
    ChartArray(i, 4) = QT(i)
Next i
End Sub
Private Sub Compare()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim LineString As String
Dim LineString2 As String
Dim LineString2a As String
Dim LineString3 As String
Dim HSIReach1V() As Single
Dim HSIReach2V() As Single
Dim HSIReach3V() As Single
Dim HSIReach4V() As Single
Dim HSIReach5V() As Single
Dim HSIReach6V() As Single
Dim HSIReach7V() As Single
Dim HSIReach8V() As Single
Dim HSIReach9V() As Single
Dim HSIReach10V() As Single
Dim HSIReach1D() As Single
Dim HSIReach2D() As Single
Dim HSIReach3D() As Single
Dim HSIReach4D() As Single
Dim HSIReach5D() As Single
Dim HSIReach6D() As Single
Dim HSIReach7D() As Single
Dim HSIReach8D() As Single
Dim HSIReach9D() As Single
Dim HSIReach10D() As Single
Dim HSIReach1DV() As Single
Dim HSIReach2DV() As Single
Dim HSIReach3DV() As Single
Dim HSIReach4DV() As Single
Dim HSIReach5DV() As Single
Dim HSIReach6DV() As Single
Dim HSIReach7DV() As Single
Dim HSIReach8DV() As Single
Dim HSIReach9DV() As Single
Dim HSIReach10DV() As Single
Dim Index() As Single
Dim AvgIndex() As Single
ReDim Index(2, 7)
ReDim AvgIndex(2)

ReDim HSIReach1V(20, 8)
ReDim HSIReach2V(20, 8)
ReDim HSIReach3V(20, 8)
ReDim HSIReach4V(20, 8)
ReDim HSIReach5V(20, 8)
ReDim HSIReach6V(20, 8)
ReDim HSIReach7V(20, 8)
ReDim HSIReach8V(20, 8)
ReDim HSIReach9V(20, 8)
ReDim HSIReach10V(20, 8)
ReDim HSIReach1D(20, 8)
ReDim HSIReach2D(20, 8)
ReDim HSIReach3D(20, 8)
ReDim HSIReach4D(20, 8)
ReDim HSIReach5D(20, 8)
ReDim HSIReach6D(20, 8)
ReDim HSIReach7D(20, 8)
ReDim HSIReach8D(20, 8)
ReDim HSIReach9D(20, 8)
ReDim HSIReach10D(20, 8)
ReDim HSIReach1DV(20, 8)
ReDim HSIReach2DV(20, 8)
ReDim HSIReach3DV(20, 8)
ReDim HSIReach4DV(20, 8)
ReDim HSIReach5DV(20, 8)
ReDim HSIReach6DV(20, 8)
ReDim HSIReach7DV(20, 8)
ReDim HSIReach8DV(20, 8)
ReDim HSIReach9DV(20, 8)
ReDim HSIReach10DV(20, 8)

ProgressBar1.Value = 5
Call Low_Flow
ProgressBar1.Value = 10
Call Variability
ProgressBar1.Value = 15
Call SelectStorms
ProgressBar1.Value = 20
Call BaseHSI
ProgressBar1.Value = 60
Call ScenarioHSI
ProgressBar1.Value = 90
Call Statistics
ProgressBar1.Value = 100

lstBaseOutput.AddItem ""
lstBaseOutput.AddItem "Base File Low Flow"
lstBaseOutput.AddItem ""
lstBaseOutput.AddItem "  " & "  2 Yr" & "  10 Yr" & "  50 Yr"
lstBaseOutput.AddItem "  1 Day" & "  " & Round(PrintLowFlow(1, 1), 2) & "  " & Round(PrintLowFlow(1, 2), 2) & "  " & Round(PrintLowFlow(1, 3), 2)
lstBaseOutput.AddItem "  7 Day" & "  " & Round(PrintLowFlow(2, 1), 2) & "  " & Round(PrintLowFlow(2, 2), 2) & "  " & Round(PrintLowFlow(2, 3), 2)
lstBaseOutput.AddItem " 15 Day" & "  " & Round(PrintLowFlow(3, 1), 2) & "  " & Round(PrintLowFlow(3, 2), 2) & "  " & Round(PrintLowFlow(3, 3), 2)
lstBaseOutput.AddItem " 30 Day" & "  " & Round(PrintLowFlow(4, 1), 2) & "  " & Round(PrintLowFlow(4, 2), 2) & "  " & Round(PrintLowFlow(4, 3), 2)
lstBaseOutput.AddItem " 60 Day" & "  " & Round(PrintLowFlow(5, 1), 2) & "  " & Round(PrintLowFlow(5, 2), 2) & "  " & Round(PrintLowFlow(5, 3), 2)

lstScenarioOutput.AddItem ""
lstScenarioOutput.AddItem "Scenario File Low Flow"
lstScenarioOutput.AddItem ""
lstScenarioOutput.AddItem "  " & "  2 Yr" & "  10 Yr" & "  50 Yr"
lstScenarioOutput.AddItem "  1 Day" & "  " & Round(PrintLowFlow(1, 4), 2) & "  " & Round(PrintLowFlow(1, 5), 2) & "  " & Round(PrintLowFlow(1, 6), 2)
lstScenarioOutput.AddItem "  7 Day" & "  " & Round(PrintLowFlow(2, 4), 2) & "  " & Round(PrintLowFlow(2, 5), 2) & "  " & Round(PrintLowFlow(2, 6), 2)
lstScenarioOutput.AddItem " 15 Day" & "  " & Round(PrintLowFlow(3, 4), 2) & "  " & Round(PrintLowFlow(3, 5), 2) & "  " & Round(PrintLowFlow(3, 6), 2)
lstScenarioOutput.AddItem " 30 Day" & "  " & Round(PrintLowFlow(4, 4), 2) & "  " & Round(PrintLowFlow(4, 5), 2) & "  " & Round(PrintLowFlow(4, 6), 2)
lstScenarioOutput.AddItem "60 Day" & "   " & Round(PrintLowFlow(5, 4), 2) & "   " & Round(PrintLowFlow(5, 5), 2) & "   " & Round(PrintLowFlow(5, 6), 2)

frmOutput.ChrtFlood.Title.Text = "Flood Frequency Curve"
frmOutput.ChrtFlood.Plot.SeriesCollection(1).LegendText = "Baseline"
frmOutput.ChrtFlood.Plot.SeriesCollection(3).LegendText = "Scenario"
frmOutput.ChrtFlood.Plot.Axis(1).AxisTitle = "Discharge (cfs)"
frmOutput.ChrtFlood.Plot.Axis(VtChAxisIdX).AxisTitle = "Return Period (Yrs)"
frmOutput.ChrtFlood.Plot.UniformAxis = False
frmOutput.ChrtFlood.ChartData = ChartArray

frmOutput.ChrtFrequency1.Title.Text = "Log Pearson 3 Density"
frmOutput.ChrtFrequency1.Plot.SeriesCollection(1).LegendText = "Baseline"
frmOutput.ChrtFrequency1.Plot.SeriesCollection(3).LegendText = "Scenario"
frmOutput.ChrtFrequency1.Plot.Axis(1).AxisTitle = "Relative Frequency"
frmOutput.ChrtFrequency1.Plot.Axis(VtChAxisIdX).AxisTitle = "Ln (Discharge (cfs))"
frmOutput.ChrtFrequency1.Plot.UniformAxis = False
frmOutput.ChrtFrequency1.ChartData = ChartArrayFreq

frmOutput.ChrtFrequency2.Title.Text = "Shifted Log Pearson 3 Density"
frmOutput.ChrtFrequency2.Plot.Axis(1).AxisTitle = "Relative Frequency"
frmOutput.ChrtFrequency2.Plot.Axis(VtChAxisIdX).AxisTitle = "Ln (Q (cfs)) - Mean Ln(Q(cfs))"
frmOutput.ChrtFrequency2.Plot.UniformAxis = False
frmOutput.ChrtFrequency2.ChartData = ChartArrayFreqShift

For j = 1 To 2
    If VariabilityArray(j, 1) <= 2.7 Then Index(j, 1) = 1
    If VariabilityArray(j, 1) > 2.7 And VariabilityArray(j, 1) < 3.2 Then Index(j, 1) = 3
    If VariabilityArray(j, 1) >= 3.2 Then Index(j, 1) = 5

    If VariabilityArray(j, 2) <= 90.81 Then Index(j, 2) = 1
    If VariabilityArray(j, 2) > 90.81 And VariabilityArray(j, 2) < 102.79 Then Index(j, 2) = 3
    If VariabilityArray(j, 2) >= 102.79 Then Index(j, 2) = 5
If VariabilityArray(j, 3) <= 7034 Then Index(j, 3) = 1
If VariabilityArray(j, 3) > 7034 And VariabilityArray(j, 3) < 10030 Then Index(j, 3) = 3
If VariabilityArray(j, 3) >= 10030 Then Index(j, 3) = 5

If VariabilityArray(j, 4) <= 1558 Then Index(j, 4) = 1
If VariabilityArray(j, 4) > 1558 And VariabilityArray(j, 4) < 2852 Then Index(j, 4) = 3
If VariabilityArray(j, 4) >= 2852 Then Index(j, 4) = 5

If VariabilityArray(j, 5) <= 73896 Then Index(j, 5) = 1
If VariabilityArray(j, 5) > 73896 And VariabilityArray(j, 5) < 140124 Then Index(j, 5) = 3
If VariabilityArray(j, 5) >= 140124 Then Index(j, 5) = 5

If VariabilityArray(j, 6) <= 103.33 Then Index(j, 6) = 1
If VariabilityArray(j, 6) > 103.33 And VariabilityArray(j, 6) < 145.67 Then Index(j, 6) = 3
If VariabilityArray(j, 6) >= 145.67 Then Index(j, 6) = 5

If VariabilityArray(j, 7) <= 15507.33 Then Index(j, 7) = 1
If VariabilityArray(j, 7) > 15507.33 And VariabilityArray(j, 7) < 19921.67 Then Index(j, 7) = 3
If VariabilityArray(j, 7) >= 19921.67 Then Index(j, 7) = 5

AvgIndex(j) = ((Index(j, 1) + Index(j, 2)) / 2 + (Index(j, 3) + Index(j, 4) + Index(j, 5)) / 3 +
(Index(j, 6) + Index(j, 7)) / 2) / 3
Next j

frmVariability.lstBaseMeasure.AddItem "Coefficient of Variation"
frmVariability.lstBaseMeasure.AddItem "90-10 Range"
frmVariability.lstBaseMeasure.AddItem "Number of Storms 3x Median"
frmVariability.lstBaseMeasure.AddItem "Number of High Flow Events"
frmVariability.lstBaseMeasure.AddItem "High Flow Duration (hrs)"
frmVariability.lstBaseMeasure.AddItem "Number of Low Flow Events"
frmVariability.lstBaseMeasure.AddItem "Low Flow Duration (hrs)"

frmVariability.lstBaseValue.AddItem Round(VariabilityArray(1, 1), 2)
frmVariability.lstBaseValue.AddItem Round(VariabilityArray(1, 2), 2)
frmVariability.lstBaseValue.AddItem Round(VariabilityArray(1, 3), 2)
frmVariability.lstBaseValue.AddItem Round(VariabilityArray(1, 4), 2)
frmVariability.lstBaseValue.AddItem Round(VariabilityArray(1, 5), 2)
frmVariability.Show

' HSI
' Velocity
' Reach 1
For j = 1 To 20
    HSIReach1V(j, 1) = ChartBaseSortedReachAcceptAreaV(1, 1, j)
    HSIReach1V(j, 2) = ChartBaseSortedReachAcceptAreaV(1, 2, j)
    HSIReach1V(j, 3) = ChartSortedReachAcceptAreaV(1, 1, j)
    HSIReach1V(j, 4) = ChartSortedReachAcceptAreaV(1, 2, j)
    HSIReach1V(j, 5) = ChartBaseSortedReachOptAreaV(1, 1, j)
    HSIReach1V(j, 6) = ChartBaseSortedReachOptAreaV(1, 2, j)
    HSIReach1V(j, 7) = ChartSortedReachOptAreaV(1, 1, j)
    HSIReach1V(j, 8) = ChartSortedReachOptAreaV(1, 2, j)
Next j

'Reach 2
For j = 1 To 20
    HSIReach2V(j, 1) = ChartBaseSortedReachAcceptAreaV(2, 1, j)
    HSIReach2V(j, 2) = ChartBaseSortedReachAcceptAreaV(2, 2, j)
    HSIReach2V(j, 3) = ChartSortedReachAcceptAreaV(2, 1, j)
    HSIReach2V(j, 4) = ChartSortedReachAcceptAreaV(2, 2, j)
    HSIReach2V(j, 5) = ChartBaseSortedReachOptAreaV(2, 1, j)
    HSIReach2V(j, 6) = ChartBaseSortedReachOptAreaV(2, 2, j)
    HSIReach2V(j, 7) = ChartSortedReachOptAreaV(2, 1, j)
    HSIReach2V(j, 8) = ChartSortedReachOptAreaV(2, 2, j)
Next j

'Reach 3
For j = 1 To 20
    HSIReach3V(j, 1) = ChartBaseSortedReachAcceptAreaV(3, 1, j)
    HSIReach3V(j, 2) = ChartBaseSortedReachAcceptAreaV(3, 2, j)
    HSIReach3V(j, 3) = ChartSortedReachAcceptAreaV(3, 1, j)
    HSIReach3V(j, 4) = ChartSortedReachAcceptAreaV(3, 2, j)
    HSIReach3V(j, 5) = ChartBaseSortedReachOptAreaV(3, 1, j)
    HSIReach3V(j, 6) = ChartBaseSortedReachOptAreaV(3, 2, j)
    HSIReach3V(j, 7) = ChartSortedReachOptAreaV(3, 1, j)
    HSIReach3V(j, 8) = ChartSortedReachOptAreaV(3, 2, j)
Next j
'Reach 4
For j = 1 To 20
    HSIReach4V(j, 1) = ChartBaseSortedReachAcceptAreaV(4, 1, j)
    HSIReach4V(j, 2) = ChartBaseSortedReachAcceptAreaV(4, 2, j)
    HSIReach4V(j, 3) = ChartSortedReachAcceptAreaV(4, 1, j)
    HSIReach4V(j, 4) = ChartSortedReachAcceptAreaV(4, 2, j)
    HSIReach4V(j, 5) = ChartBaseSortedReachOptAreaV(4, 1, j)
    HSIReach4V(j, 6) = ChartBaseSortedReachOptAreaV(4, 2, j)
    HSIReach4V(j, 7) = ChartSortedReachOptAreaV(4, 1, j)
    HSIReach4V(j, 8) = ChartSortedReachOptAreaV(4, 2, j)
Next j
'
Reach 5
For j = 1 To 20
    HSIReach5V(j, 1) = ChartBaseSortedReachAcceptAreaV(5, 1, j)
    HSIReach5V(j, 2) = ChartBaseSortedReachAcceptAreaV(5, 2, j)
    HSIReach5V(j, 3) = ChartSortedReachAcceptAreaV(5, 1, j)
    HSIReach5V(j, 4) = ChartSortedReachAcceptAreaV(5, 2, j)
    HSIReach5V(j, 5) = ChartBaseSortedReachOptAreaV(5, 1, j)
    HSIReach5V(j, 6) = ChartBaseSortedReachOptAreaV(5, 2, j)
    HSIReach5V(j, 7) = ChartSortedReachOptAreaV(5, 1, j)
    HSIReach5V(j, 8) = ChartSortedReachOptAreaV(5, 2, j)
Next j
'
Reach 6
For j = 1 To 20
    HSIReach6V(j, 1) = ChartBaseSortedReachAcceptAreaV(6, 1, j)
    HSIReach6V(j, 2) = ChartBaseSortedReachAcceptAreaV(6, 2, j)
    HSIReach6V(j, 3) = ChartSortedReachAcceptAreaV(6, 1, j)
    HSIReach6V(j, 4) = ChartSortedReachAcceptAreaV(6, 2, j)
    HSIReach6V(j, 5) = ChartBaseSortedReachOptAreaV(6, 1, j)
    HSIReach6V(j, 6) = ChartBaseSortedReachOptAreaV(6, 2, j)
    HSIReach6V(j, 7) = ChartSortedReachOptAreaV(6, 1, j)
    HSIReach6V(j, 8) = ChartSortedReachOptAreaV(6, 2, j)
Next j
'
Reach 7
For j = 1 To 20
    HSIReach7V(j, 1) = ChartBaseSortedReachAcceptAreaV(7, 1, j)
    HSIReach7V(j, 2) = ChartBaseSortedReachAcceptAreaV(7, 2, j)
    HSIReach7V(j, 3) = ChartSortedReachAcceptAreaV(7, 1, j)

252
HSIReach7V(j, 4) = ChartSortedReachAcceptAreaV(7, 2, j)
HSIReach7V(j, 5) = ChartBaseSortedReachOptAreaV(7, 1, j)
HSIReach7V(j, 6) = ChartBaseSortedReachOptAreaV(7, 2, j)
HSIReach7V(j, 7) = ChartSortedReachOptAreaV(7, 1, j)
HSIReach7V(j, 8) = ChartSortedReachOptAreaV(7, 2, j)
Next j
'Reach 8
For j = 1 To 20
    HSIReach8V(j, 1) = ChartBaseSortedReachAcceptAreaV(8, 1, j)
    HSIReach8V(j, 2) = ChartBaseSortedReachAcceptAreaV(8, 2, j)
    HSIReach8V(j, 3) = ChartSortedReachAcceptAreaV(8, 1, j)
    HSIReach8V(j, 4) = ChartSortedReachAcceptAreaV(8, 2, j)
    HSIReach8V(j, 5) = ChartBaseSortedReachOptAreaV(8, 1, j)
    HSIReach8V(j, 6) = ChartBaseSortedReachOptAreaV(8, 2, j)
    HSIReach8V(j, 7) = ChartSortedReachOptAreaV(8, 1, j)
    HSIReach8V(j, 8) = ChartSortedReachOptAreaV(8, 2, j)
Next j
'Reach 9
For j = 1 To 20
    HSIReach9V(j, 1) = ChartBaseSortedReachAcceptAreaV(9, 1, j)
    HSIReach9V(j, 2) = ChartBaseSortedReachAcceptAreaV(9, 2, j)
    HSIReach9V(j, 3) = ChartSortedReachAcceptAreaV(9, 1, j)
    HSIReach9V(j, 4) = ChartSortedReachAcceptAreaV(9, 2, j)
    HSIReach9V(j, 5) = ChartBaseSortedReachOptAreaV(9, 1, j)
    HSIReach9V(j, 6) = ChartBaseSortedReachOptAreaV(9, 2, j)
    HSIReach9V(j, 7) = ChartSortedReachOptAreaV(9, 1, j)
    HSIReach9V(j, 8) = ChartSortedReachOptAreaV(9, 2, j)
Next j
'Reach 10
For j = 1 To 20
    HSIReach10V(j, 1) = ChartBaseSortedReachAcceptAreaV(10, 1, j)
    HSIReach10V(j, 2) = ChartBaseSortedReachAcceptAreaV(10, 2, j)
    HSIReach10V(j, 3) = ChartSortedReachAcceptAreaV(10, 1, j)
    HSIReach10V(j, 4) = ChartSortedReachAcceptAreaV(10, 2, j)
    HSIReach10V(j, 5) = ChartBaseSortedReachOptAreaV(10, 1, j)
    HSIReach10V(j, 6) = ChartBaseSortedReachOptAreaV(10, 2, j)
    HSIReach10V(j, 7) = ChartSortedReachOptAreaV(10, 1, j)
    HSIReach10V(j, 8) = ChartSortedReachOptAreaV(10, 2, j)
Next j
'Depth
'Reach 1
For j = 1 To 20
    HSIReach1D(j, 1) = ChartBaseSortedReachAcceptAreaD(1, 1, j)
    HSIReach1D(j, 2) = ChartBaseSortedReachAcceptAreaD(1, 2, j)
    HSIReach1D(j, 3) = ChartSortedReachAcceptAreaD(1, 1, j)
    HSIReach1D(j, 4) = ChartSortedReachAcceptAreaD(1, 2, j)
    HSIReach1D(j, 5) = ChartBaseSortedReachOptAreaD(1, 1, j)
    HSIReach1D(j, 6) = ChartBaseSortedReachOptAreaD(1, 2, j)
    HSIReach1D(j, 7) = ChartSortedReachOptAreaD(1, 1, j)
    HSIReach1D(j, 8) = ChartSortedReachOptAreaD(1, 2, j)
Next j
'Reach 2
For j = 1 To 20
    HSIReach2D(j, 1) = ChartBaseSortedReachAcceptAreaD(2, 1, j)
    HSIReach2D(j, 2) = ChartBaseSortedReachAcceptAreaD(2, 2, j)
    HSIReach2D(j, 3) = ChartSortedReachAcceptAreaD(2, 1, j)
    HSIReach2D(j, 4) = ChartSortedReachAcceptAreaD(2, 2, j)
    HSIReach2D(j, 5) = ChartBaseSortedReachOptAreaD(2, 1, j)
    HSIReach2D(j, 6) = ChartBaseSortedReachOptAreaD(2, 2, j)
    HSIReach2D(j, 7) = ChartSortedReachOptAreaD(2, 1, j)
    HSIReach2D(j, 8) = ChartSortedReachOptAreaD(2, 2, j)
Next j
'Reach 3
For j = 1 To 20
    HSIReach3D(j, 1) = ChartBaseSortedReachAcceptAreaD(3, 1, j)
    HSIReach3D(j, 2) = ChartBaseSortedReachAcceptAreaD(3, 2, j)
    HSIReach3D(j, 3) = ChartSortedReachAcceptAreaD(3, 1, j)
    HSIReach3D(j, 4) = ChartSortedReachAcceptAreaD(3, 2, j)
    HSIReach3D(j, 5) = ChartBaseSortedReachOptAreaD(3, 1, j)
    HSIReach3D(j, 6) = ChartBaseSortedReachOptAreaD(3, 2, j)
    HSIReach3D(j, 7) = ChartSortedReachOptAreaD(3, 1, j)
    HSIReach3D(j, 8) = ChartSortedReachOptAreaD(3, 2, j)
Next j
'Reach 4
For j = 1 To 20
    HSIReach4D(j, 1) = ChartBaseSortedReachAcceptAreaD(4, 1, j)
    HSIReach4D(j, 2) = ChartBaseSortedReachAcceptAreaD(4, 2, j)
    HSIReach4D(j, 3) = ChartSortedReachAcceptAreaD(4, 1, j)
    HSIReach4D(j, 4) = ChartSortedReachAcceptAreaD(4, 2, j)
    HSIReach4D(j, 5) = ChartBaseSortedReachOptAreaD(4, 1, j)
    HSIReach4D(j, 6) = ChartBaseSortedReachOptAreaD(4, 2, j)
    HSIReach4D(j, 7) = ChartSortedReachOptAreaD(4, 1, j)
    HSIReach4D(j, 8) = ChartSortedReachOptAreaD(4, 2, j)
HSIREach4D(j, 2) = ChartBaseSortedReachAcceptAreaD(4, 2, j)
HSIREach4D(j, 3) = ChartSortedReachAcceptAreaD(4, 1, j)
HSIREach4D(j, 4) = ChartSortedReachAcceptAreaD(4, 2, j)
HSIREach4D(j, 5) = ChartBaseSortedReachOptAreaD(4, 1, j)
HSIREach4D(j, 6) = ChartBaseSortedReachOptAreaD(4, 2, j)
HSIREach4D(j, 7) = ChartSortedReachOptAreaD(4, 1, j)
HSIREach4D(j, 8) = ChartSortedReachOptAreaD(4, 2, j)
Next j
'Reach 5
For j = 1 To 20
    HSIREach5D(j, 1) = ChartBaseSortedReachAcceptAreaD(5, 1, j)
    HSIREach5D(j, 2) = ChartBaseSortedReachAcceptAreaD(5, 2, j)
    HSIREach5D(j, 3) = ChartSortedReachAcceptAreaD(5, 1, j)
    HSIREach5D(j, 4) = ChartSortedReachAcceptAreaD(5, 2, j)
    HSIREach5D(j, 5) = ChartBaseSortedReachOptAreaD(5, 1, j)
    HSIREach5D(j, 6) = ChartBaseSortedReachOptAreaD(5, 2, j)
    HSIREach5D(j, 7) = ChartSortedReachOptAreaD(5, 1, j)
    HSIREach5D(j, 8) = ChartSortedReachOptAreaD(5, 2, j)
Next j
'Reach 6
For j = 1 To 20
    HSIREach6D(j, 1) = ChartBaseSortedReachAcceptAreaD(6, 1, j)
    HSIREach6D(j, 2) = ChartBaseSortedReachAcceptAreaD(6, 2, j)
    HSIREach6D(j, 3) = ChartSortedReachAcceptAreaD(6, 1, j)
    HSIREach6D(j, 4) = ChartSortedReachAcceptAreaD(6, 2, j)
    HSIREach6D(j, 5) = ChartBaseSortedReachOptAreaD(6, 1, j)
    HSIREach6D(j, 6) = ChartBaseSortedReachOptAreaD(6, 2, j)
    HSIREach6D(j, 7) = ChartSortedReachOptAreaD(6, 1, j)
    HSIREach6D(j, 8) = ChartSortedReachOptAreaD(6, 2, j)
Next j
'Reach 7
For j = 1 To 20
    HSIREach7D(j, 1) = ChartBaseSortedReachAcceptAreaD(7, 1, j)
    HSIREach7D(j, 2) = ChartBaseSortedReachAcceptAreaD(7, 2, j)
    HSIREach7D(j, 3) = ChartSortedReachAcceptAreaD(7, 1, j)
    HSIREach7D(j, 4) = ChartSortedReachAcceptAreaD(7, 2, j)
    HSIREach7D(j, 5) = ChartBaseSortedReachOptAreaD(7, 1, j)
    HSIREach7D(j, 6) = ChartBaseSortedReachOptAreaD(7, 2, j)
HSIReach7D(j, 7) = ChartSortedReachOptAreaD(7, 1, j)
HSIReach7D(j, 8) = ChartSortedReachOptAreaD(7, 2, j)
Next j
'Reach 8
For j = 1 To 20
  HSIReach8D(j, 1) = ChartBaseSortedReachAcceptAreaD(8, 1, j)
  HSIReach8D(j, 2) = ChartBaseSortedReachAcceptAreaD(8, 2, j)
  HSIReach8D(j, 3) = ChartSortedReachAcceptAreaD(8, 1, j)
  HSIReach8D(j, 4) = ChartSortedReachAcceptAreaD(8, 2, j)
  HSIReach8D(j, 5) = ChartBaseSortedReachOptAreaD(8, 1, j)
  HSIReach8D(j, 6) = ChartBaseSortedReachOptAreaD(8, 2, j)
  HSIReach8D(j, 7) = ChartSortedReachOptAreaD(8, 1, j)
  HSIReach8D(j, 8) = ChartSortedReachOptAreaD(8, 2, j)
Next j
'Reach 9
For j = 1 To 20
  HSIReach9D(j, 1) = ChartBaseSortedReachAcceptAreaD(9, 1, j)
  HSIReach9D(j, 2) = ChartBaseSortedReachAcceptAreaD(9, 2, j)
  HSIReach9D(j, 3) = ChartSortedReachAcceptAreaD(9, 1, j)
  HSIReach9D(j, 4) = ChartSortedReachAcceptAreaD(9, 2, j)
  HSIReach9D(j, 5) = ChartBaseSortedReachOptAreaD(9, 1, j)
  HSIReach9D(j, 6) = ChartBaseSortedReachOptAreaD(9, 2, j)
  HSIReach9D(j, 7) = ChartSortedReachOptAreaD(9, 1, j)
  HSIReach9D(j, 8) = ChartSortedReachOptAreaD(9, 2, j)
Next j
'Reach 10
For j = 1 To 20
  HSIReach10D(j, 1) = ChartBaseSortedReachAcceptAreaD(10, 1, j)
  HSIReach10D(j, 2) = ChartBaseSortedReachAcceptAreaD(10, 2, j)
  HSIReach10D(j, 3) = ChartSortedReachAcceptAreaD(10, 1, j)
  HSIReach10D(j, 4) = ChartSortedReachAcceptAreaD(10, 2, j)
  HSIReach10D(j, 5) = ChartBaseSortedReachOptAreaD(10, 1, j)
  HSIReach10D(j, 6) = ChartBaseSortedReachOptAreaD(10, 2, j)
  HSIReach10D(j, 7) = ChartSortedReachOptAreaD(10, 1, j)
  HSIReach10D(j, 8) = ChartSortedReachOptAreaD(10, 2, j)
Next j
'Depth and Velocity
'Reach 1

256
For \( j = 1 \) To 20
\[
\begin{align*}
\text{HSIReach1DV}(j, 1) & = \text{ChartBaseSortedReachAcceptAreaDV}(1, 1, j) \\
\text{HSIReach1DV}(j, 2) & = \text{ChartBaseSortedReachAcceptAreaDV}(1, 2, j) \\
\text{HSIReach1DV}(j, 3) & = \text{ChartSortedReachAcceptAreaDV}(1, 1, j) \\
\text{HSIReach1DV}(j, 4) & = \text{ChartSortedReachAcceptAreaDV}(1, 2, j) \\
\text{HSIReach1DV}(j, 5) & = \text{ChartBaseSortedReachOptAreaDV}(1, 1, j) \\
\text{HSIReach1DV}(j, 6) & = \text{ChartBaseSortedReachOptAreaDV}(1, 2, j) \\
\text{HSIReach1DV}(j, 7) & = \text{ChartSortedReachOptAreaDV}(1, 1, j) \\
\text{HSIReach1DV}(j, 8) & = \text{ChartSortedReachOptAreaDV}(1, 2, j)
\end{align*}
\]
Next \( j \)

'Reach 2
For \( j = 1 \) To 20
\[
\begin{align*}
\text{HSIReach2DV}(j, 1) & = \text{ChartBaseSortedReachAcceptAreaDV}(2, 1, j) \\
\text{HSIReach2DV}(j, 2) & = \text{ChartBaseSortedReachAcceptAreaDV}(2, 2, j) \\
\text{HSIReach2DV}(j, 3) & = \text{ChartSortedReachAcceptAreaDV}(2, 1, j) \\
\text{HSIReach2DV}(j, 4) & = \text{ChartSortedReachAcceptAreaDV}(2, 2, j) \\
\text{HSIReach2DV}(j, 5) & = \text{ChartBaseSortedReachOptAreaDV}(2, 1, j) \\
\text{HSIReach2DV}(j, 6) & = \text{ChartBaseSortedReachOptAreaDV}(2, 2, j) \\
\text{HSIReach2DV}(j, 7) & = \text{ChartSortedReachOptAreaDV}(2, 1, j) \\
\text{HSIReach2DV}(j, 8) & = \text{ChartSortedReachOptAreaDV}(2, 2, j)
\end{align*}
\]
Next \( j \)

'Reach 3
For \( j = 1 \) To 20
\[
\begin{align*}
\text{HSIReach3DV}(j, 1) & = \text{ChartBaseSortedReachAcceptAreaDV}(3, 1, j) \\
\text{HSIReach3DV}(j, 2) & = \text{ChartBaseSortedReachAcceptAreaDV}(3, 2, j) \\
\text{HSIReach3DV}(j, 3) & = \text{ChartSortedReachAcceptAreaDV}(3, 1, j) \\
\text{HSIReach3DV}(j, 4) & = \text{ChartSortedReachAcceptAreaDV}(3, 2, j) \\
\text{HSIReach3DV}(j, 5) & = \text{ChartBaseSortedReachOptAreaDV}(3, 1, j) \\
\text{HSIReach3DV}(j, 6) & = \text{ChartBaseSortedReachOptAreaDV}(3, 2, j) \\
\text{HSIReach3DV}(j, 7) & = \text{ChartSortedReachOptAreaDV}(3, 1, j) \\
\text{HSIReach3DV}(j, 8) & = \text{ChartSortedReachOptAreaDV}(3, 2, j)
\end{align*}
\]
Next \( j \)

'Reach 4
For \( j = 1 \) To 20
\[
\begin{align*}
\text{HSIReach4DV}(j, 1) & = \text{ChartBaseSortedReachAcceptAreaDV}(4, 1, j) \\
\text{HSIReach4DV}(j, 2) & = \text{ChartBaseSortedReachAcceptAreaDV}(4, 2, j) \\
\text{HSIReach4DV}(j, 3) & = \text{ChartSortedReachAcceptAreaDV}(4, 1, j) \\
\text{HSIReach4DV}(j, 4) & = \text{ChartSortedReachAcceptAreaDV}(4, 2, j)
\end{align*}
\]
HSIReach4DV(j, 5) = ChartBaseSortedReachOptAreaDV(4, 1, j)
HSIReach4DV(j, 6) = ChartBaseSortedReachOptAreaDV(4, 2, j)
HSIReach4DV(j, 7) = ChartSortedReachOptAreaDV(4, 1, j)
HSIReach4DV(j, 8) = ChartSortedReachOptAreaDV(4, 2, j)
Next j
' Reach 5
For j = 1 To 20
    HSIReach5DV(j, 1) = ChartBaseSortedReachAcceptAreaDV(5, 1, j)
    HSIReach5DV(j, 2) = ChartBaseSortedReachAcceptAreaDV(5, 2, j)
    HSIReach5DV(j, 3) = ChartSortedReachAcceptAreaDV(5, 1, j)
    HSIReach5DV(j, 4) = ChartSortedReachAcceptAreaDV(5, 2, j)
    HSIReach5DV(j, 5) = ChartBaseSortedReachOptAreaDV(5, 1, j)
    HSIReach5DV(j, 6) = ChartBaseSortedReachOptAreaDV(5, 2, j)
    HSIReach5DV(j, 7) = ChartSortedReachOptAreaDV(5, 1, j)
    HSIReach5DV(j, 8) = ChartSortedReachOptAreaDV(5, 2, j)
Next j
' Reach 6
For j = 1 To 20
    HSIReach6DV(j, 1) = ChartBaseSortedReachAcceptAreaDV(6, 1, j)
    HSIReach6DV(j, 2) = ChartBaseSortedReachAcceptAreaDV(6, 2, j)
    HSIReach6DV(j, 3) = ChartSortedReachAcceptAreaDV(6, 1, j)
    HSIReach6DV(j, 4) = ChartSortedReachAcceptAreaDV(6, 2, j)
    HSIReach6DV(j, 5) = ChartBaseSortedReachOptAreaDV(6, 1, j)
    HSIReach6DV(j, 6) = ChartBaseSortedReachOptAreaDV(6, 2, j)
    HSIReach6DV(j, 7) = ChartSortedReachOptAreaDV(6, 1, j)
    HSIReach6DV(j, 8) = ChartSortedReachOptAreaDV(6, 2, j)
Next j
' Reach 7
For j = 1 To 20
    HSIReach7DV(j, 1) = ChartBaseSortedReachAcceptAreaDV(7, 1, j)
    HSIReach7DV(j, 2) = ChartBaseSortedReachAcceptAreaDV(7, 2, j)
    HSIReach7DV(j, 3) = ChartSortedReachAcceptAreaDV(7, 1, j)
    HSIReach7DV(j, 4) = ChartSortedReachAcceptAreaDV(7, 2, j)
    HSIReach7DV(j, 5) = ChartBaseSortedReachOptAreaDV(7, 1, j)
    HSIReach7DV(j, 6) = ChartBaseSortedReachOptAreaDV(7, 2, j)
    HSIReach7DV(j, 7) = ChartSortedReachOptAreaDV(7, 1, j)
    HSIReach7DV(j, 8) = ChartSortedReachOptAreaDV(7, 2, j)
Next j
'Reach 8
For j = 1 To 20
    HSIReach8DV(j, 1) = ChartBaseSortedReachAcceptAreaDV(8, 1, j)
    HSIReach8DV(j, 2) = ChartBaseSortedReachAcceptAreaDV(8, 2, j)
    HSIReach8DV(j, 3) = ChartSortedReachAcceptAreaDV(8, 1, j)
    HSIReach8DV(j, 4) = ChartSortedReachAcceptAreaDV(8, 2, j)
    HSIReach8DV(j, 5) = ChartBaseSortedReachOptAreaDV(8, 1, j)
    HSIReach8DV(j, 6) = ChartBaseSortedReachOptAreaDV(8, 2, j)
    HSIReach8DV(j, 7) = ChartSortedReachOptAreaDV(8, 1, j)
    HSIReach8DV(j, 8) = ChartSortedReachOptAreaDV(8, 2, j)
Next j
'Reach 9
For j = 1 To 20
    HSIReach9DV(j, 1) = ChartBaseSortedReachAcceptAreaDV(9, 1, j)
    HSIReach9DV(j, 2) = ChartBaseSortedReachAcceptAreaDV(9, 2, j)
    HSIReach9DV(j, 3) = ChartSortedReachAcceptAreaDV(9, 1, j)
    HSIReach9DV(j, 4) = ChartSortedReachAcceptAreaDV(9, 2, j)
    HSIReach9DV(j, 5) = ChartBaseSortedReachOptAreaDV(9, 1, j)
    HSIReach9DV(j, 6) = ChartBaseSortedReachOptAreaDV(9, 2, j)
    HSIReach9DV(j, 7) = ChartSortedReachOptAreaDV(9, 1, j)
    HSIReach9DV(j, 8) = ChartSortedReachOptAreaDV(9, 2, j)
Next j
'Reach 10
For j = 1 To 20
    HSIReach10DV(j, 1) = ChartBaseSortedReachAcceptAreaDV(10, 1, j)
    HSIReach10DV(j, 2) = ChartBaseSortedReachAcceptAreaDV(10, 2, j)
    HSIReach10DV(j, 3) = ChartSortedReachAcceptAreaDV(10, 1, j)
    HSIReach10DV(j, 4) = ChartSortedReachAcceptAreaDV(10, 2, j)
    HSIReach10DV(j, 5) = ChartBaseSortedReachOptAreaDV(10, 1, j)
    HSIReach10DV(j, 6) = ChartBaseSortedReachOptAreaDV(10, 2, j)
    HSIReach10DV(j, 7) = ChartSortedReachOptAreaDV(10, 1, j)
    HSIReach10DV(j, 8) = ChartSortedReachOptAreaDV(10, 2, j)
Next j
frmHSIVelocity.chrtVelocity1.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIVelocity.chrtVelocity1.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity1/chartData = HSIReach1V
frmHSIVelocity.chrtVelocity2.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity2.ChartData = HSIReach2V

frmHSIVelocity.chrtVelocity3.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity3.ChartData = HSIReach3V

frmHSIVelocity.chrtVelocity4.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity4.ChartData = HSIReach4V

frmHSIVelocity.chrtVelocity5.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIVelocity.chrtVelocity5.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity5.ChartData = HSIReach5V

frmHSIVelocity.chrtVelocity6.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity6.ChartData = HSIReach6V

frmHSIVelocity.chrtVelocity7.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity7.ChartData = HSIReach7V

frmHSIVelocity.chrtVelocity8.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity8.ChartData = HSIReach8V

frmHSIVelocity.chrtVelocity9.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIVelocity.chrtVelocity9.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity9.ChartData = HSIReach9V
frmHSIVelocity.chrtVelocity9.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"

frmHSIVelocity.chrtVelocity10.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocity10.ChartData = HSIReach10V
frmHSIVelocity.chrtVelocity10.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"

frmHSIVelocity.chrtVelocityStream.Plot.UniformAxis = False
frmHSIVelocity.chrtVelocityStream.ChartData = ChartSortedStreamAreaV
frmHSIVelocity.chrtVelocityStream.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"
frmHSIVelocity.chrtVelocityStream.Plot.SeriesCollection(1).LegendText = "Baseline Acceptable"
frmHSIVelocity.chrtVelocityStream.Plot.SeriesCollection(3).LegendText = "Scenario Acceptable"
frmHSIVelocity.chrtVelocityStream.Plot.SeriesCollection(5).LegendText = "Baseline Optimum"
frmHSIVelocity.chrtVelocityStream.Plot.SeriesCollection(7).LegendText = "Scenario Optimum"
frmHSIDepth.chrtDepth1.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIDepth.chrtDepth1.Plot.UniformAxis = False
frmHSIDepth.chrtDepth1.ChartData = HSIReach1D

frmHSIDepth.chrtDepth2.Plot.UniformAxis = False
frmHSIDepth.chrtDepth2.ChartData = HSIReach2D

frmHSIDepth.chrtDepth3.Plot.UniformAxis = False
frmHSIDepth.chrtDepth3.ChartData = HSIReach3D

frmHSIDepth.chrtDepth4.Plot.UniformAxis = False
frmHSIDepth.chrtDepth4.ChartData = HSIReach4D

frmHSIDepth.chrtDepth5.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIDepth.chrtDepth5.Plot.UniformAxis = False
frmHSIDepth.chrtDepth5.ChartData = HSIReach5D

frmHSIDepth.chrtDepth6.Plot.UniformAxis = False
frmHSIDepth.chrtDepth6.ChartData = HSIReach6D

frmHSIDepth.chrtDepth7.Plot.UniformAxis = False
frmHSIDepth.chrtDepth7.ChartData = HSIReach7D

frmHSIDepth.chrtDepth8.Plot.UniformAxis = False
frmHSIDepth.chrtDepth8.ChartData = HSIReach8D

frmHSIDepth.chrtDepth9.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIDepth.chrtDepth9.Plot.UniformAxis = False
frmHSIDepth.chrtDepth9.ChartData = HSIReach9D
frmHSIDepth.chrtDepth9.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"

frmHSIDepth.chrtDepth10.Plot.UniformAxis = False
frmHSIDepth.chrtDepth10.ChartData = HSIReach10D
frmHSIDepth.chrtDepth10.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"

frmHSIDepth.chrtDepthStream.Plot.UniformAxis = False
frmHSIDepth.chrtDepthStream.ChartData = ChartSortedStreamAreaD
frmHSIDepth.chrtDepthStream.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"
frmHSIDepth.chrtDepthStream.Plot.SeriesCollection(1).LegendText = "Baseline Acceptable"
frmHSIDepth.chrtDepthStream.Plot.SeriesCollection(3).LegendText = "Scenario Acceptable"
frmHSIDepth.chrtDepthStream.Plot.SeriesCollection(5).LegendText = "Baseline Optimum"
frmHSIDepth.chrtDepthStream.Plot.SeriesCollection(7).LegendText = "Scenario Optimum"

frmHSIDV.chrtDV1.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIDV.chrtDV1.Plot.UniformAxis = False
frmHSIDV.chrtDV1.ChartData = HSIReach1DV

frmHSIDV.chrtDV2.Plot.UniformAxis = False
frmHSIDV.chrtDV2.ChartData = HSIReach2DV

frmHSIDV.chrtDV3.Plot.UniformAxis = False
frmHSIDV.chrtDV3.ChartData = HSIReach3DV

frmHSIDV.chrtDV4.Plot.UniformAxis = False
frmHSIDV.chrtDV4.ChartData = HSIReach4DV

frmHSIDV.chrtDV5.Plot.Axis(1).AxisTitle = "Quantile"
frmHSIDV.chrtDV5.Plot.UniformAxis = False
frmHSIDV.chrtDV5.ChartData = HSIReach5DV

frmHSIDV.chrtDV6.Plot.UniformAxis = False
frmHSIDV.chrtDV6.ChartData = HSIReach6DV

frmHSIDV.chrtDV7.Plot.UniformAxis = False
frmHSIDV.chrtDV7.ChartData = HSIReach7DV

frmHSIDV.chrtDV8.Plot.UniformAxis = False
frmHSIDV.chrtDV8.ChartData = HSIReach8DV

frmHSIDV.chrtDV9.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"
frmHSIDV.chrtDV9.Plot.UniformAxis = False
frmHSIDV.chrtDV9.ChartData = HSIReach9DV
frmHSIDV.chrtDV9.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"
frmHSIDV.chrtDV10.Plot.UniformAxis = False
frmHSIDV.chrtDV10.ChartData = HSIReach10DV
frmHSIDV.chrtDV10.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"

frmHSIDV.chrtDVStream.Plot.UniformAxis = False
frmHSIDV.chrtDVStream.ChartData = ChartSortedStreamAreaDV
frmHSIDV.chrtDVStream.Plot.Axis(VtChAxisIdX).AxisTitle = "Fraction of Available"
frmHSIDV.chrtDVStream.Plot.SeriesCollection(1).LegendText = "Baseline Acceptable"
frmHSIDV.chrtDVStream.Plot.SeriesCollection(3).LegendText = "Scenario Acceptable"
frmHSIDV.chrtDVStream.Plot.SeriesCollection(5).LegendText = "Baseline Optimum"
frmHSIDV.chrtDVStream.Plot.SeriesCollection(7).LegendText = "Scenario Optimum"

CommonDialog1.FilterIndex = 3
CommonDialog1.FileName = ""
CommonDialog1.ShowSave
If CommonDialog1.FileName <> "" Then
    CompareFile = CommonDialog1.FileName
    Open CompareFile For Output As #2
    LineString = "Run Date" & "," & Date & vbCrLf
    LineString = LineString & "Run Time" & "," & Time & vbCrLf
    LineString = LineString & "Base File" & "," & BaseFile & vbCrLf
    LineString = LineString & "Scenario File" & "," & ScenarioFile & vbCrLf
    For i = 1 To 3
        LineString = LineString & ";" & vbCrLf
    Next i
    LineString = LineString & "BaseFile" & "," & "G" & "," & Skew(2) & "," & "S" & "," & StdDev(2) & "," & "Cv" & "," & ChartArrayStats(1) & vbCrLf
    LineString = LineString & "ScenarioFile" & "," & "G" & "," & Skew(1) & "," & "S" & "," & StdDev(1) & "," & "Cv" & "," & ChartArrayStats(2) & vbCrLf
    LineString = LineString & ";" & vbCrLf
    LineString = LineString & "Return Period" & "," & "Base File Output" & "," & "Scenario File Output" & vbCrLf
    For i = 1 To 6
        LineString = LineString & ChartArray(i, 1) & "," & ChartArray(i, 2) & "," & ChartArray(i, 4) & vbCrLf
    Next i
End If
LineString = LineString & "Base File Hydrographs" & vbCrLf
For i = 1 To 6
    LineString = LineString & ChartArray(i, 1) & "Yr Storm" & vbCrLf
    If BaseHydrographCount(i) <> 0 Then
        For k = 1 To BaseHydrographCount(i)
            If BaseHr(StartEnd(i, 2) + k - 1) = 24 Then
                LineString = LineString & BaseMo(StartEnd(i, 2) + k - 1) & "/" & BaseDy(StartEnd(i, 2) + k) & "/" & BaseYr(StartEnd(i, 2) + k - 1) & "" & "0" & ":" & BaseMn(StartEnd(i, 2) + k - 1) & "," & BaseqStormHydrograph(i, k) & vbCrLf
            Else
                LineString = LineString & BaseMo(StartEnd(i, 2) + k - 1) & "/" & BaseDy(StartEnd(i, 2) + k - 1) & "/" & BaseYr(StartEnd(i, 2) + k - 1) & BaseHr(StartEnd(i, 2) + k - 1) & ":" & BaseMn(StartEnd(i, 2) + k - 1) & "," & BaseqStormHydrograph(i, k) & vbCrLf
            End If
        Next k
    Else
        LineString = LineString & "None" & vbCrLf
    End If
Next i
LineString = LineString & "Scenario File Hydrographs" & vbCrLf
For i = 1 To 6
    LineString = LineString & ChartArray(i, 1) & "Yr Storm" & vbCrLf
    If HydrographCount(i) <> 0 Then
        For k = 1 To HydrographCount(i)
            If Hr(StartEnd(i, 1) + k - 1) = 24 Then
                LineString = LineString & Mo(StartEnd(i, 1) + k - 1) & "/" & Dy(StartEnd(i, 1) + k) & "/" & Yr(StartEnd(i, 1) + k - 1) & ":" & Mn(StartEnd(i, 1) + k - 1) & "," & qStormHydrograph(i, k) & vbCrLf
            Else
                LineString = LineString & Mo(StartEnd(i, 1) + k - 1) & "/" & Dy(StartEnd(i, 1) + k - 1) & "/" & Yr(StartEnd(i, 1) + k - 1) & ":" & Mn(StartEnd(i, 1) + k - 1) & "," & qStormHydrograph(i, k) & vbCrLf
            End If
        Next k
    Else
        LineString = LineString & "None" & vbCrLf
    End If
Next i
Next

LineString = LineString & "" & vbCrLf
LineString = LineString & "Base File Low Flow" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "7 Day" & vbCrLf
LineString = LineString & "15 Day" & vbCrLf
LineString = LineString & "30 Day" & vbCrLf
LineString = LineString & "60 Day" & vbCrLf
LineString = LineString & "90 Day" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "Scenario File Low Flow" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "7 Day" & vbCrLf
LineString = LineString & "15 Day" & vbCrLf
LineString = LineString & "30 Day" & vbCrLf
LineString = LineString & "60 Day" & vbCrLf
LineString = LineString & "90 Day" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "Scenario File Log Pearson Type 3 Density" & vbCrLf
LineString = LineString & "Ln Q" & vbCrLf
LineString = LineString & "LP3 Density" & vbCrLf
For i = 1 To 20
    LineString = LineString & ChartArrayFreq(i, 3) & vbCrLf
    LineString = LineString & ChartArrayFreq(i, 4) & vbCrLf
Next i
Next i

LineString = LineString & "" & vbCrLf
LineString = LineString & "Base File Log Pearson Type 3 Density" & vbCrLf
LineString = LineString & "" & vbCrLf
LineString = LineString & "Ln Q" & "," & "LP3 Density" & vbCrLf
For i = 1 To 20
    LineString = LineString & ChartArrayFreq(i, 1) & "," & ChartArrayFreq(i, 2) & vbCrLf
Next i
Print #2, LineString
Close #2

End If

'Create Stream Health File
CommonDialog1.FilterIndex = 3
CommonDialog1.FileName = ""
CommonDialog1.ShowSave
If CommonDialog1.FileName <> "" Then
    CompareFile = CommonDialog1.FileName
    Open CompareFile For Output As #32
    LineString2 = "Run Date" & "," & Date & vbCrLf
    LineString2 = LineString2 & "Run Time" & "," & Time & vbCrLf
    LineString2 = LineString2 & "Base File" & "," & BaseFile & vbCrLf
    LineString2 = LineString2 & "Scenario File" & "," & ScenarioFile & vbCrLf
    LineString2 = LineString2 & "HSI File" & "," & HSIFile & vbCrLf

'Variability
    LineString2 = LineString2 & "Variability" & vbCrLf
    LineString2 = LineString2 & "" & vbCrLf
    LineString2 = LineString2 & "Coefficient of Variation" & "," & VariabilityArray(1, 1) & "," & "Index"
    LineString2 = LineString2 & "," & Index(1, 1) & vbCrLf
    LineString2 = LineString2 & "90-10 Range" & "," & VariabilityArray(1, 2) & "," & "Index"
    LineString2 = LineString2 & "," & Index(1, 2) & vbCrLf
LineString2 = LineString2 & "Number of Storms 3x Median" & "," & VariabilityArray(1, 3) & "," & "Index" & "," & Index(1, 3) & vbCrLf
LineString2 = LineString2 & "Number of High Flow Events" & "," & VariabilityArray(1, 4) & "," & "Index" & "," & Index(1, 4) & vbCrLf
LineString2 = LineString2 & "High Flow Duration (hrs)" & "," & VariabilityArray(1, 5) & "," & "Index" & "," & Index(1, 5) & vbCrLf
LineString2 = LineString2 & "Number of Low Flow Events" & "," & VariabilityArray(1, 6) & "," & "Index" & "," & Index(1, 6) & vbCrLf
LineString2 = LineString2 & "Low Flow Duration (hrs)" & "," & VariabilityArray(1, 7) & "," & "Index" & "," & Index(1, 7) & vbCrLf
LineString2 = LineString2 & "Overall Index" & "," & AvgIndex(1) & vbCrLf
LineString2 = LineString2 & vbCrLf
LineString2 = LineString2 & "Scenario" & vbCrLf
LineString2 = LineString2 & "Coefficient of Variation" & "," & VariabilityArray(2, 1) & "," & "Index" & "," & Index(2, 1) & vbCrLf
LineString2 = LineString2 & "90-10 Range" & "," & VariabilityArray(2, 2) & "," & "Index" & "," & Index(2, 2) & vbCrLf
LineString2 = LineString2 & "Number of Storms 3x Median" & "," & VariabilityArray(2, 3) & "," & "Index" & "," & Index(2, 3) & vbCrLf
LineString2 = LineString2 & "Number of High Flow Events" & "," & VariabilityArray(2, 4) & "," & "Index" & "," & Index(2, 4) & vbCrLf
LineString2 = LineString2 & "High Flow Duration (hrs)" & "," & VariabilityArray(2, 5) & "," & "Index" & "," & Index(2, 5) & vbCrLf
LineString2 = LineString2 & "Number of Low Flow Events" & "," & VariabilityArray(2, 6) & "," & "Index" & "," & Index(2, 6) & vbCrLf
LineString2 = LineString2 & "Low Flow Duration (hrs)" & "," & VariabilityArray(2, 7) & "," & "Index" & "," & Index(2, 7) & vbCrLf
LineString2 = LineString2 & "Overall Index" & "," & AvgIndex(2) & vbCrLf

'HSI
LineString2 = LineString2 & "HSI" & vbCrLf
LineString2 = LineString2 & vbCrLf
LineString2 = LineString2 & "Base" & vbCrLf
For i = 1 To 3
  LineString2 = LineString2 & "" & vbCrLf
Next i
LineString2 = LineString2 & "Accept Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartBaseSortedReachAcceptAreaV(i, k, j) & ","
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartBaseSortedReachAcceptAreaD(i, k, j) & ","
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
LineString2 = LineString2 & "Accept Depth and Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartBaseSortedReachAcceptAreaV(i, k, j) & ","
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
Next j
LineString2 = LineString2 & "Accept Depth" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartBaseSortedReachAcceptAreaD(i, k, j) & ","
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
Next j
LineString2 = LineString2 & "Accept Depth and Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
"Quantile6" & "," & "Quantile7" & "," & "Quantile8" & "," & "Quantile9" & "," & "Quantile10" & vbCrLf
For j = 1 To 20
    For i = 1 To 10
        For k = 1 To 2
            LineString2 = LineString2 & ChartBaseSortedReachAcceptAreaDV(i, k, j) & ","
            If i = 10 And k = 2 Then
                LineString2 = LineString2 & vbCrLf
            End If
        Next k
    Next i
Next j
LineString2 = LineString2 & "Opt Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Quantile2" & "," & "Quantile3" & "," & "Quantile4" & "," & "Quantile5" & "," & "Quantile6" & "," & "Quantile7" & "," & "Quantile8" & "," & "Quantile9" & "," & "Quantile10" & vbCrLf
For j = 1 To 20
    For i = 1 To 10
        For k = 1 To 2
            LineString2 = LineString2 & ChartBaseSortedReachOptAreaV(i, k, j) & ","
            If i = 10 And k = 2 Then
                LineString2 = LineString2 & vbCrLf
            End If
        Next k
    Next i
Next j
LineString2 = LineString2 & "Opt Depth" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Quantile2" & "," & "Quantile3" & "," & "Quantile4" & "," & "Quantile5" & "," & "Quantile6" & "," & "Quantile7" & "," & "Quantile8" & "," & "Quantile9" & "," & "Quantile10" & vbCrLf
For j = 1 To 20
    For i = 1 To 10
        For k = 1 To 2
            LineString2 = LineString2 & ChartBaseSortedReachOptAreaD(i, k, j) & ","
            If i = 10 And k = 2 Then
                LineString2 = LineString2 & vbCrLf
            End If
        Next k
    Next i
Next j

End If
Next k
Next i
Next j
LineString2 = LineString2 & "Opt Depth and Velocity" & vbCrLf
LineString2 = LineString2 & "Reach1" & "," & "Quantile1" & "," & "Reach1" & "" & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach9" & "," & "Quantile9" & "," & "Reach10" & "," & "Quantile10" & "," & vbCrLf
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartBaseSortedReachOptAreaDV(i, k, j) & ""
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
Next j
LineString2 = LineString2 & vbCrLf
LineString2 = LineString2 & "Stream" & vbCrLf
LineString2 = LineString2 & vbCrLf
LineString2 = LineString2 & "Velocity" & "," & "," & "," & "," & "," & "Depth" & "," & "," & "," & "," & vbCrLf
For j = 1 To 20
  LineString2 = LineString2 & ChartSortedStreamAreaV(j, 1) & "," & ChartSortedStreamAreaV(j, 2) & "," & ChartSortedStreamAreaV(j, 5) & "," & ChartSortedStreamAreaV(j, 6) & "," & ChartSortedStreamAreaD(j, 1) & "," & ChartSortedStreamAreaD(j, 2) & "," & ChartSortedStreamAreaD(j, 5) & "," & ChartSortedStreamAreaD(j, 6) & "," & ChartSortedStreamAreaDV(j, 1) & "," & ChartSortedStreamAreaDV(j, 2) & "," & ChartSortedStreamAreaDV(j, 5) & "," & ChartSortedStreamAreaDV(j, 6) & vbCrLf
Next j
'Scenario
LineString2 = LineString2 & "" & vbCrLf
LineString2 = LineString2 & "Scenario" & vbCrLf
LineString2 = LineString2 & "" & vbCrLf
LineString2 = LineString2 & "Accept Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
    For i = 1 To 10
        For k = 1 To 2
            LineString2 = LineString2 & ChartSortedReachAcceptAreaV(i, k, j) & ","
        Next k
        If i = 10 And k = 2 Then
            LineString2 = LineString2 & vbCrLf
        End If
    Next i
Next j

LineString2 = LineString2 & "Accept Depth" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
    For i = 1 To 10
        For k = 1 To 2
            LineString2 = LineString2 & ChartSortedReachAcceptAreaD(i, k, j) & ","
        Next k
        If i = 10 And k = 2 Then
            LineString2 = LineString2 & vbCrLf
        End If
    Next i
Next j

LineString2 = LineString2 & "Accept Depth and Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
    For i = 1 To 10
        For k = 1 To 2
            LineString2 = LineString2 & ChartSortedReachAcceptAreaV(i, k, j) & ","
        Next k
        If i = 10 And k = 2 Then
            LineString2 = LineString2 & vbCrLf
        End If
    Next i
Next j
Next i
Next j
"" & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartSortedReachAcceptAreaDV(i, k, j) & ","
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
Next j
LineString2 = LineString2 & "Opt Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartSortedReachOptAreaV(i, k, j) & ","
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
Next j
LineString2 = LineString2 & "Opt Depth" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
  For i = 1 To 10
    For k = 1 To 2
      LineString2 = LineString2 & ChartSortedReachOptAreaD(i, k, j) & ","
      If i = 10 And k = 2 Then
        LineString2 = LineString2 & vbCrLf
      End If
    Next k
  Next i
Next j
LineString2 = LineString2 & vbCrLf

272
End If
Next k
Next i
Next j
LineString2 = LineString2 & "Opt Depth and Velocity" & vbCrLf
LineString2 = LineString2 & "Quantile1" & "," & "Reach1" & "," & "Quantile2" & "," & "Reach2" & "," & "Quantile3" & "," & "Reach3" & "," & "Quantile4" & "," & "Reach4" & "," & "Quantile5" & "," & "Reach5" & "," & "Quantile6" & "," & "Reach6" & "," & "Quantile7" & "," & "Reach7" & "," & "Quantile8" & "," & "Reach8" & "," & "Quantile9" & "," & "Reach9" & "," & "Quantile10" & "," & "Reach10" & vbCrLf
For j = 1 To 20
    For i = 1 To 10
        For k = 1 To 2
            LineString2 = LineString2 & ChartSortedReachOptAreaDV(i, k, j) & ","
            If i = 10 And k = 2 Then
                LineString2 = LineString2 & vbCrLf
            End If
        Next k
    Next i
Next j
LineString2 = LineString2 & "Stream" & vbCrLf
LineString2 = LineString2 & vbCrLf
LineString2 = LineString2 & "Velocity" & "," & "," & "," & "," & "," & "," & "," & "Depth" & "," & "," & "," & "," & vbCrLf
For j = 1 To 20
    LineString2 = LineString2 & ChartSortedStreamAreaV(j, 3) & "," & ChartSortedStreamAreaV(j, 4) & "," & ChartSortedStreamAreaV(j, 7) & "," & ChartSortedStreamAreaV(j, 8) & "," & ChartSortedStreamAreaD(j, 3) & "," & ChartSortedStreamAreaD(j, 4) & "," & ChartSortedStreamAreaD(j, 7) & "," & ChartSortedStreamAreaD(j, 8) & "," & ChartSortedStreamAreaDV(j, 3) & "," & ChartSortedStreamAreaDV(j, 4) & "," & ChartSortedStreamAreaDV(j, 7) & "," & ChartSortedStreamAreaDV(j, 8) & vbCrLf
Next j
LineString2 = LineString2 & vbCrLf
LineString2 = LineString2 & "Time %" & vbCrLf

273
LineString2 = LineString2 & "Basefile" & vbCrLf
LineString2 = LineString2 & "Percent of time that the following are available:" & vbCrLf
For k = 1 To 10
    LineString2 = LineString2 & "Optimum Velocity in Reach" & "," & k & "," & vbCrLf
    BaseReachPercentOptTimeV(k) * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Acceptable Velocity in Reach" & "," & k & "," & vbCrLf
    BaseReachPercentAcceptTimeV(k) * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Optimum Depth in Reach" & "," & k & "," & vbCrLf
    BaseReachPercentOptTimeD(k) * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Acceptable Depth in Reach" & "," & k & "," & vbCrLf
    BaseReachPercentAcceptTimeD(k) * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Optimum Depth & Velocity in Reach" & "," & k & "," & vbCrLf
    BaseReachPercentOptTimeDV(k) * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Acceptable Depth & Velocity in Reach" & "," & k & "," & vbCrLf
Next k

LineString2 = LineString2 & "Scenario" & vbCrLf
LineString2 = LineString2 & "Percent of time that the following are available:" & vbCrLf
For k = 1 To 10
    LineString2 = LineString2 & "Optimum Velocity in Stream" & "," & vbCrLf
    BaseStreamPercentOptTimeV * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Acceptable Velocity in Stream" & "," & vbCrLf
    BaseStreamPercentAcceptTimeV * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Optimum Depth in Stream" & "," & vbCrLf
    BaseStreamPercentOptTimeD * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Acceptable Depth in Stream" & "," & vbCrLf
    BaseStreamPercentAcceptTimeD * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Optimum Depth & Velocity in Stream" & "," & vbCrLf
    BaseStreamPercentOptTimeDV * 100 & "," & vbCrLf
    LineString2 = LineString2 & "Acceptable Depth & Velocity in Stream" & "," & vbCrLf
Next k
LineString2 = LineString2 & "Acceptable Depth in Reach" & "," & k & "," & ReachPercentAcceptTimeD(k) * 100 & "," & "%" & vbCrLf
LineString2 = LineString2 & "Optimum Depth & Velocity in Reach" & "," & k & "," & ReachPercentOptTimeDV(k) * 100 & "," & "%" & vbCrLf
Next k
LineString2 = LineString2 & "Optimum Velocity in Stream" & "," & StreamPercentOptTimeV * 100 & "," & "%" & vbCrLf
LineString2 = LineString2 & "Acceptable Velocity in Stream" & "," & StreamPercentAcceptTimeV * 100 & "," & "%" & vbCrLf
LineString2 = LineString2 & "Optimum Depth in Stream" & "," & StreamPercentOptTimeD * 100 & "," & "%" & vbCrLf
LineString2 = LineString2 & "Acceptable Depth in Stream" & "," & StreamPercentAcceptTimeD * 100 & "," & "%" & vbCrLf
LineString2 = LineString2 & "Optimum Depth & Velocity in Stream" & "," & StreamPercentOptTimeDV * 100 & "," & "%" & vbCrLf
LineString2 = LineString2 & "Acceptable Depth & Velocity in Stream" & "," & StreamPercentAcceptTimeDV * 100 & "," & "%" & vbCrLf
Print #32, LineString2
Close #32
End If
frmOutput.Show
frmHSIVelocity.Show
frmHSIDepth.Show
frmHSIDV.Show
ProgressBar1.Value = 0
End Sub

Private Sub mnuDefaults_Click()
frmOptions.Show (1)
End Sub

Private Sub mnuExit_Click()
Unload Me
End Sub
Private Sub mnuHSI_Click()
Dim i As Long
Dim j As Long
Dim trash As String

CommonDialog1.Filter = "All Files (*.*)|*.|CSV Files (*.CSV)|*.CSV"
CommonDialog1.FileName = ""
CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then
    CurrentFile = CommonDialog1.FileName
    HSIFile = CurrentFile
    txtHSIFile(1).Text = CommonDialog1.FileName
    Open CurrentFile For Input As #7

    ReDim OptimumV(2), AcceptableV(2) 'Resize arrays to specified size
    ReDim OptimumD(2), AcceptableD(2) 'Resize arrays to specified size

    Input #7, trash, OptimumV(1), OptimumV(2)
    Input #7, trash, AcceptableV(1), AcceptableV(2)
    Input #7, trash, OptimumD(1), OptimumD(2)
    Input #7, trash, AcceptableD(1), AcceptableD(2)

    Close #7
End If
End Sub

Private Sub mnuScenario_Click()
Dim i As Long
Dim j As Long
Dim Fileline As String

'CommonDialog1.Filter = "All Files (*.*)|*.|HSPF Files (*.PLT)|*.PLT"
CommonDialog1.FilterIndex = 2
CommonDialog1.FileName = ""
CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then
    CurrentFile = CommonDialog1.FileName
    ScenarioFile = CurrentFile
    txtScenarioFile(0).Text = CommonDialog1.FileName
    Open CurrentFile For Input As #1
    ProgressBar1.Value = 1 / 10
    For i = 1 To 26
        Line Input #1, Fileline 'Skip Heading Line
    Next i
    j = 0
    Do While Not EOF(1)
        Line Input #1, Fileline
        j = j + 1
    Loop
    Close #1
ReDim Yr(j), Mo(j), Dy(j), Hr(j), Mn(j), q(10, j) 'Resize Array to Unknown Size
Open CurrentFile For Input As #1
For i = 1 To 26
    Line Input #1, Fileline 'Skip Heading Line
Next i
For i = 1 To j
    ProgressBar1.Value = (i / j * 100)
    Line Input #1, Fileline
    'Read Variables From File
    Yr(i) = val(Trim(Mid(Fileline, 6, 5)))
    Mo(i) = val(Trim(Mid(Fileline, 12, 2)))
    Dy(i) = val(Trim(Mid(Fileline, 15, 2)))
    Hr(i) = val(Trim(Mid(Fileline, 18, 2)))
    Mn(i) = val(Trim(Mid(Fileline, 21, 2)))
    q(1, i) = val(Trim(Mid(Fileline, 26, 7)))
    q(2, i) = val(Trim(Mid(Fileline, 40, 7)))
    q(3, i) = val(Trim(Mid(Fileline, 54, 7)))
q(4, i) = val(Trim(Mid(Fileline, 68, 7)))
q(5, i) = val(Trim(Mid(Fileline, 82, 7)))
q(6, i) = val(Trim(Mid(Fileline, 96, 7)))
q(7, i) = val(Trim(Mid(Fileline, 110, 7)))
q(8, i) = val(Trim(Mid(Fileline, 124, 7)))
q(9, i) = val(Trim(Mid(Fileline, 138, 7)))
q(10, i) = val(Trim(Mid(Fileline, 152, 7)))
Next i
TotalLines = j
Close #1
ProgressBar1.Value = 0

Call ScenarioFlood
End If
End Sub
Sub SelectStorms()

Dim qStorm As Single
Dim BaseqStorm As Single
Dim i As Long
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim qStormPeak() As Single
Dim qStormPeakLocation() As Single
Dim qStormStatus() As Integer
Dim BaseqStormPeak() As Single
Dim BaseqStormPeakLocation() As Single
Dim BaseqStormStatus() As Integer
Dim BackCount As Integer
Dim ForCount As Integer
Dim BackCountCheck As Integer
Dim ForCountCheck As Integer
Dim PercentDifference As Single
Dim StormHydrographStart As Long
Dim HydrographPeak As Single
Dim BaseHydrographPeak As Single
Dim AbsolutePeak As Integer
Dim StormCountArray() As Integer
Dim BaseStormCountArray() As Integer
Dim BankfullPeak() As Single
Dim BaseBankfullPeak() As Single

ReDim BaseqStormHydrograph(6, 1000)
ReDim qStormHydrograph(6, 1000)
ReDim StormCountArray(7)
ReDim BaseStormCountArray(7)
ReDim StartEnd(6, 2)

For j = 1 To 6
    qStorm = 10000000000#
    ReDim Preserve qStormPeak(j), qStormPeakLocation(j), qStormStatus(j)

    For i = 2 To TotalLines - 1
        AbsolutePeak = 0
        'Check if Local Peak
        If q(10, i - 1) < q(10, i) And q(10, i + 1) < q(10, i) Then
            'Check if Absolute Peak
            'Move backward in time
            BackCountCheck = 0
            HydrographPeak = i
            Do While BackCountCheck < TimeLimit
                If HydrographPeak <> 1 Then
                    PercentDifference = Abs((q(10, HydrographPeak) - q(10, HydrographPeak - 1)) / q(10, HydrographPeak))
                    If q(10, HydrographPeak) > q(10, i) Then
                        AbsolutePeak = 1
                        End If
                End If
                BackCountCheck = BackCountCheck + 1
                HydrographPeak = HydrographPeak - 1
            Loop
        End If
    Next i
Next j

AbsolutePeak = If PercentDifference >= Percent Then 0 Else 1
BackCountCheck = TimeLimit
End If
Loop 'Move forward in time
ForCountCheck = 0
HydrographPeak = i
Do While ForCountCheck < TimeLimit
    PercentDifference = Abs((q(10, HydrographPeak) - q(10, HydrographPeak + 1)) / q(10, HydrographPeak))
    If q(10, HydrographPeak) > q(10, i) Then
        AbsolutePeak = 1
    End If
    ForCountCheck = ForCountCheck + 1
    HydrographPeak = HydrographPeak + 1
    If PercentDifference >= Percent Then ForCountCheck = 0 'resets count for Percent
    Loop
End If 'AbsolutePeak <> 1 Then
'Count Storms
'High flow variability data based on baseline bankfull
If q(10, i) > Bankfull Then
    HighFlowCount = HighFlowCount + 1
    ReDim Preserve BankfullPeak(2, HighFlowCount)
    BankfullPeak(1, HighFlowCount) = i
End If

'FRE3 data based on baseline bankfull
If q(10, i) > FRE3 Then
    FRE3Count = FRE3Count + 1
End If

Select Case q(10, i)
    Case Is < ChartArray(6, 2) ' less than 2 yrs
        StormCountArray(1) = StormCountArray(1) + 1
    Case ChartArray(6, 2) To ChartArray(5, 2) '2 to 5 yrs
        StormCountArray(2) = StormCountArray(2) + 1
    Case ChartArray(5, 2) To ChartArray(4, 2) '5 to 10 yrs
        StormCountArray(3) = StormCountArray(3) + 1
End Select
StormCountArray(3) = StormCountArray(3) + 1
Case ChartArray(4, 2) To ChartArray(3, 2) '10 to 25 yrs
  StormCountArray(4) = StormCountArray(4) + 1
Case ChartArray(3, 2) To ChartArray(2, 2) '25 to 50 yrs
  StormCountArray(5) = StormCountArray(5) + 1
Case ChartArray(2, 2) To ChartArray(1, 2) '50 to 100 yrs
  StormCountArray(6) = StormCountArray(6) + 1
Case Is > ChartArray(1, 2) ' more than 100 yrs
  StormCountArray(7) = StormCountArray(7) + 1
End Select
'Check for Peak Percent Violation
If Abs(q(10, i) - ChartArray(j, 4)) < Percent * ChartArray(j, 4) Then
  'Check for closest peak
  If Abs(q(10, i) - ChartArray(j, 4)) < qStorm Then
    qStorm = Abs(q(10, i) - ChartArray(j, 4))
    qStormPeak(j) = q(10, i)
    qStormPeakLocation(j) = i
    qStormStatus(j) = 1 '1= "j" yr return period storm found  0= storm Not found
  End If
End If
End If
Next i
Next j

'Base Return Period Storms
For j = 1 To 6
  BaseqStorm = 10000000000#
  ReDim Preserve BaseqStormPeak(j), BaseqStormPeakLocation(j), BaseqStormStatus(j)
  For i = 2 To BaseTotalLines - 1
    AbsolutePeak = 0
    'Check if Local Peak
    If Baseq(10, i - 1) < Baseq(10, i) And Baseq(10, i + 1) < Baseq(10, i) Then
      'Check if Absolute Peak
      'Move backward in time
      BackCountCheck = 0
      BaseHydrographPeak = i
    End If
  Next i
Next j
Do While BackCountCheck < TimeLimit
    If BaseHydrographPeak <> 1 Then
        PercentDifference = Abs((Baseq(10, BaseHydrographPeak) - Baseq(10, 
                                BaseHydrographPeak - 1)) / Baseq(10, BaseHydrographPeak))
        If Baseq(10, BaseHydrographPeak) > Baseq(10, i) Then
            AbsolutePeak = 1
        End If
        BackCountCheck = BackCountCheck + 1
        BaseHydrographPeak = BaseHydrographPeak - 1
        If PercentDifference >= Percent Then BackCountCheck = 0 'resets count for Percent Violation
    Else
        AbsolutePeak = 1
        BackCountCheck = TimeLimit
    End If
Loop
'Move forward in time
ForCountCheck = 0
BaseHydrographPeak = i
Do While ForCountCheck < TimeLimit
    PercentDifference = Abs((Baseq(10, BaseHydrographPeak) - Baseq(10, BaseHydrographPeak + 
                            1)) / Baseq(10, BaseHydrographPeak))
    If Baseq(10, BaseHydrographPeak) > Baseq(10, i) Then
        AbsolutePeak = 1
    End If
    ForCountCheck = ForCountCheck + 1
    BaseHydrographPeak = BaseHydrographPeak + 1
    If PercentDifference >= Percent Then ForCountCheck = 0 'resets count for Percent Violation
Loop

If AbsolutePeak <> 1 Then
    'Count Storms
    'High flow variability data based on baseline bankfull
    If Baseq(10, i) > Bankfull Then
        BaseHighFlowCount = BaseHighFlowCount + 1
        BankfullPeak(2, BaseHighFlowCount) = i
    End If
End If
End If
'FRE3 data based on baseline bankfull
If Baseq(10, i) > FRE3 Then
    BaseFRE3Count = BaseFRE3Count + 1
End If

Select Case Baseq(10, i)
    Case Is < ChartArray(6, 2) ' less than 2 yrs
        BaseStormCountArray(1) = BaseStormCountArray(1) + 1
    Case ChartArray(6, 2) To ChartArray(5, 2) ' 2 to 5 yrs
        BaseStormCountArray(2) = BaseStormCountArray(2) + 1
    Case ChartArray(5, 2) To ChartArray(4, 2) ' 5 to 10 yrs
        BaseStormCountArray(3) = BaseStormCountArray(3) + 1
    Case ChartArray(4, 2) To ChartArray(3, 2) ' 10 to 25 yrs
        BaseStormCountArray(4) = BaseStormCountArray(4) + 1
    Case ChartArray(3, 2) To ChartArray(2, 2) ' 25 to 50 yrs
        BaseStormCountArray(5) = BaseStormCountArray(5) + 1
    Case ChartArray(2, 2) To ChartArray(1, 2) ' 50 to 100 yrs
        BaseStormCountArray(6) = BaseStormCountArray(6) + 1
    Case Is > ChartArray(1, 2) ' more than 100 yrs
        BaseStormCountArray(7) = BaseStormCountArray(7) + 1
End Select

'Check for Peak Percent Violation
If Abs(Baseq(10, i) - ChartArray(j, 2)) < (Percent * ChartArray(j, 2)) Then
    'Check for closest peak
    If Abs(Baseq(10, i) - ChartArray(j, 2)) < BaseqStorm Then
        BaseqStorm = Abs(Baseq(10, i) - ChartArray(j, 2))
        BaseqStormPeak(j) = Baseq(10, i)
        BaseqStormPeakLocation(j) = i
        BaseqStormStatus(j) = 1 '1= "j" yr return period storm found 0= storm Not found
    End If
End If
Next i
Next j
'Scenario Hydrographs
For j = 1 To 6
    ReDim Preserve HydrographCount(j)
    If qStormStatus(j) = 1 Then
        'Move backward in time
        BackCount = 0
        BackCountCheck = 0
        HydrographPeak = qStormPeakLocation(j)
        Do While BackCountCheck < TimeLimit
            PercentDifference = Abs((q(10, HydrographPeak) - q(10, HydrographPeak - 1)) / q(10,
                                    HydrographPeak - 1))
            BackCount = BackCount + 1
            BackCountCheck = BackCountCheck + 1
            HydrographPeak = HydrographPeak - 1
            If PercentDifference >= Percent Then BackCountCheck = 0 'resets count for Percent Violation
        Loop
        'Move forward in time
        ForCount = 0
        ForCountCheck = 0
        HydrographPeak = qStormPeakLocation(j)
        Do While ForCountCheck < TimeLimit
            PercentDifference = Abs((q(10, HydrographPeak) - q(10, HydrographPeak + 1)) / q(10,
                                    HydrographPeak))
            ForCount = ForCount + 1
            ForCountCheck = ForCountCheck + 1
            HydrographPeak = HydrographPeak + 1
            If PercentDifference >= Percent Then ForCountCheck = 0 'resets count for Percent Violation
        Loop
        StormHydrographStart = qStormPeakLocation(j) - BackCount
        For l = 0 To ForCount + BackCount
            'ReDim Preserve qStormHydrograph(j, l + 1)
            qStormHydrograph(j, l + 1) = q(10, StormHydrographStart + l)
        Next l
        HydrographCount(j) = l
        StartEnd(j, 1) = StormHydrographStart
    Else
        'ReDim Preserve qStormHydrograph(j, 1)
'qStormHydrograph(j, 1) = 0
Next j

'Base Hydrographs
For j = 1 To 6
    ReDim Preserve BaseHydrographCount(j)
    If BaseqStormStatus(j) = 1 Then
        'Move backward in time
        BackCount = 0
        BackCountCheck = 0
        BaseHydrographPeak = BaseqStormPeakLocation(j)
        Do While BackCountCheck < TimeLimit
            PercentDifference = Abs((Baseq(10, BaseHydrographPeak) - Baseq(10, BaseHydrographPeak - 1)) / Baseq(10, BaseHydrographPeak - 1))
            BackCount = BackCount + 1
            BackCountCheck = BackCountCheck + 1
            BaseHydrographPeak = BaseHydrographPeak - 1
            If PercentDifference >= Percent Then BackCountCheck = 0 'resets count for Percent Violation
        Loop
        'Move forward in time
        ForCount = 0
        ForCountCheck = 0
        BaseHydrographPeak = BaseqStormPeakLocation(j)
        Do While ForCountCheck < TimeLimit
            PercentDifference = Abs((Baseq(10, BaseHydrographPeak) - Baseq(10, BaseHydrographPeak + 1)) / Baseq(10, BaseHydrographPeak))
            ForCount = ForCount + 1
            ForCountCheck = ForCountCheck + 1
            BaseHydrographPeak = BaseHydrographPeak + 1
            If PercentDifference >= Percent Then ForCountCheck = 0 'resets count for Percent Violation
        Loop
        StormHydrographStart = BaseqStormPeakLocation(j) - BackCount
        For l = 0 To ForCount + BackCount
            'ReDim Preserve BaseqStormHydrograph(j, l + 1)
            BaseqStormHydrograph(j, l + 1) = Baseq(10, StormHydrographStart + l)
        Next l
    BaseHydrographCount(j) = l
StartEnd(j, 2) = StormHydrographStart
End If

Next j

lstOutput.AddItem "Number of Storms in Return Period Classes"
lstOutput.AddItem "Scenario File"
lstOutput.AddItem "Less Than 2 Yr Peak Storms : " & StormCountArray(1) / 6 'Each peak was counted 5 times with 6 Tr
lstOutput.AddItem "2 to 5 Yr Peak Storms : " & StormCountArray(2) / 6
lstOutput.AddItem "5 to 10 Yr Peak Storms : " & StormCountArray(3) / 6
lstOutput.AddItem "10 to 25 Yr Peak Storms : " & StormCountArray(4) / 6
lstOutput.AddItem "25 to 50 Yr Peak Storms : " & StormCountArray(5) / 6
lstOutput.AddItem "50 to 100 Yr Peak Storms : " & StormCountArray(6) / 6
lstOutput.AddItem "More Than 100 Yr Peak Storms : " & StormCountArray(7) / 6
lstOutput.AddItem "Base File"
lstOutput.AddItem "Less Than 2 Yr Peak Storms : " & BaseStormCountArray(1) / 6
lstOutput.AddItem "2 to 5 Yr Peak Storms : " & BaseStormCountArray(2) / 6
lstOutput.AddItem "5 to 10 Yr Peak Storms : " & BaseStormCountArray(3) / 6
lstOutput.AddItem "10 to 25 Yr Peak Storms : " & BaseStormCountArray(4) / 6
lstOutput.AddItem "25 to 50 Yr Peak Storms : " & BaseStormCountArray(5) / 6
lstOutput.AddItem "50 to 100 Yr Peak Storms : " & BaseStormCountArray(6) / 6
lstOutput.AddItem "More Than 100 Yr Peak Storms : " & BaseStormCountArray(7) / 6

'High flow(Bankfull) duration calculations
'Scenario Duration
For j = 1 To HighFlowCount
    'Move backward in time
    BackCount = 0
    BackCountCheck = 0
    HydrographPeak = BankfullPeak(1, j)
    Do While BackCountCheck < TimeLimit
        PercentDifference = Abs((q(10, HydrographPeak) - q(10, HydrographPeak - 1)) / q(10, HydrographPeak - 1))
        BackCount = BackCount + 1
        BackCountCheck = BackCountCheck + 1
    Loop
Next j
HydrographPeak = HydrographPeak - 1
If PercentDifference >= Percent Then BackCountCheck = 0 'resets count for Percent Violation
Loop
'Move forward in time
ForCount = 0
ForCountCheck = 0
HydrographPeak = BankfullPeak(1, j)
Do While ForCountCheck < TimeLimit
PercentDifference = Abs((q(10, HydrographPeak) - q(10, HydrographPeak + 1)) / q(10, HydrographPeak))
ForCount = ForCount + 1
ForCountCheck = ForCountCheck + 1
HydrographPeak = HydrographPeak + 1
If PercentDifference >= Percent Then ForCountCheck = 0 'resets count for Percent Violation
Loop
StormHydrographStart = BankfullPeak(1, j) - BackCount
HighFlowDuration = ForCount + BackCount + HighFlowDuration

Next j

'Base Duration
For j = 1 To BaseHighFlowCount
'Move backward in time
BackCount = 0
BackCountCheck = 0
HydrographPeak = BankfullPeak(2, j)
Do While BackCountCheck < TimeLimit
PercentDifference = Abs((Baseq(10, HydrographPeak) - Baseq(10, HydrographPeak - 1)) / q(10, HydrographPeak - 1))
BackCount = BackCount + 1
BackCountCheck = BackCountCheck + 1
HydrographPeak = HydrographPeak - 1
If PercentDifference >= Percent Then BackCountCheck = 0 'resets count for Percent Violation
Loop
'Move forward in time
ForCount = 0
ForCountCheck = 0
HydrographPeak = BankfullPeak(2, j)
Do While ForCountCheck < TimeLimit

287
PercentDifference = Abs((Baseq(10, HydrographPeak) - Baseq(10, HydrographPeak + 1)) / q(10, HydrographPeak))
ForCount = ForCount + 1
ForCountCheck = ForCountCheck + 1
HydrographPeak = HydrographPeak + 1
If PercentDifference >= Percent Then ForCountCheck = 0 'resets count for percent violation
Loop
BaseHighFlowDuration = ForCount + BackCount + BaseHighFlowDuration
Next j

'Fill rest of variability array
VariabilityArray(1, 3) = BaseFRE3Count
VariabilityArray(1, 4) = BaseHighFlowCount
VariabilityArray(1, 5) = BaseHighFlowDuration

VariabilityArray(2, 3) = FRE3Count
VariabilityArray(2, 4) = HighFlowCount
VariabilityArray(2, 5) = HighFlowDuration
End Sub

Sub Low_Flow()
Dim TL() As Single
Dim T0 As Single
Dim n As Single
Dim N0 As Single
Dim qL As Double
Dim ML As Double
Dim SL As Double
Dim GL As Double
Dim T As Double
Dim KL As Double
Dim DayAverage As Single
Dim i As Long
Dim j As Integer
Dim LowFlow() As Single
Dim LowFlowDate() As Integer
Dim HourCount As Double
Dim HourSum As Single
Dim WaterYr As Integer

288
Dim DayArray() As Single
Dim k As Integer
Dim DayArraySize As Integer
Dim CurrentYear As Integer
Dim LowFlowCounter As Integer
Dim Average As Single
Dim SumTemp As Single
Dim AverageTemp As Single
Dim ZeroLowFlowCounter As Integer
Dim ZeroLowFlow() As Single
Dim SumQ As Double
Dim LnZeroLowFlow() As Single
Dim SumDiffSquared As Double
Dim SumDiffCubed As Double
Dim z As Double
Dim KT As Double
Dim QT() As Double
Dim l As Integer
Dim Days() As Single
Dim Complete As Integer

ReDim TL(5, 3), QT(5, 3), PrintLowFlow(5, 6)
ReDim Days(5)

'Define Days Array
Days(1) = 1
Days(2) = 7
Days(3) = 15
Days(4) = 30
Days(5) = 60

For l = 1 To 5

'Daily Averages
j = 0 'Annual Peaks Array Counter

289
i = 1 'Total Dataset Array Counter
Do While i < BaseTotalLines + 1

'Check for new water year
If BaseMo(i) = 10 And BaseDy(i) = 1 And BaseHr(i) = 1 And BaseMn(i) = 0 Then
    WaterYr = BaseYr(i) + 1
End If

'Check for new day
If BaseHr(i) = 1 Then 'If at a new day
    j = j + 1
    ReDim Preserve DayArray(5, j) 'Resize Array to Unknown Size
    HourSum = 0

    'Sum all flows over the day
    For k = 0 To 23
        HourSum = HourSum + Baseq(10, i + k)
    Next k

    DayArray(1, j) = WaterYr 'Record Water Yr
    DayArray(2, j) = BaseYr(i) 'Record Julian Yr
    DayArray(3, j) = BaseMo(i) 'Record Julian Mo
    DayArray(4, j) = BaseDy(i) 'Record Julian Dy
    DayArray(5, j) = HourSum / 24 'Record Daily average flow
    DayArraySize = j
End If
i = i + 1
Loop

'Get Days(l) day averages for each year
CurrentYear = 0
LowFlowCounter = 0
For i = 1 To DayArraySize
    If DayArray(1, i) <> CurrentYear Then 'Check if new water year has begun

290
LowFlowCounter = LowFlowCounter + 1
ReDim Preserve LowFlow(3, LowFlowCounter)
Average = 10000000000# 'Set the lowest average to very high number for the upcoming year
End If
SumTemp = 0 'Set the previous Days(l) day sum to zero

'Eliminate first Day(l)-1 averages as incomplete
Select Case Days(l)
    Case Is < 32 '31 Days in October
        If DayArray(3, i) = 10 And DayArray(4, i) < Days(l) Then
            Complete = 0 'Part of Incomplete Data Set at year start
        Else
            Complete = 1 'Part of complete Data Set
        End If
    Case 32 To 61 '31 + 30 days in November
        If DayArray(3, i) = 10 Or DayArray(3, i) = 11 And DayArray(4, i) < Days(l) - 31 Then
            Complete = 0 'Part of Incomplete Data Set at year start
        Else
            Complete = 1 'Part of complete Data Set
        End If
    Case 62 To 92 '61 + 31 days in December
        If DayArray(3, i) = 10 Or DayArray(3, i) = 11 Or DayArray(3, i) = 12 And DayArray(4, i) < Days(l) - 61 Then
            Complete = 0 'Part of Incomplete Data Set at year start
        Else
            Complete = 1 'Part of complete Data Set
        End If
End Select
If Complete = 0 Then
Else
    For j = i - (Days(l) - 1) To i
        SumTemp = SumTemp + DayArray(5, j) 'sum the Days(i) days of average flows
    Next j

AverageTemp = SumTemp / Days(l) 'calculate the Days(l) day average
If AverageTemp < Average Then 'check if current average is less than previous min
    Average = AverageTemp 'replace the previous min with the new min
End If

' Set LowFlow current array
LowFlow(1, LowFlowCounter) = DayArray(1, i)
LowFlow(2, LowFlowCounter) = DayArray(2, i)
LowFlow(3, LowFlowCounter) = Average
End If
CurrentYear = DayArray(1, i) ' Set the current water year
Next i

' Count zeroes and create a new array
N0 = 0
n = LowFlowCounter
ZeroLowFlowCounter = 0
For i = 1 To n
  If LowFlow(3, i) = 0 Then
    N0 = N0 + 1
  Else
    ZeroLowFlowCounter = ZeroLowFlowCounter + 1
    ReDim Preserve ZeroLowFlow(3, ZeroLowFlowCounter)
    ZeroLowFlow(1, ZeroLowFlowCounter) = LowFlow(1, i)
    ZeroLowFlow(2, ZeroLowFlowCounter) = LowFlow(2, i)
    ZeroLowFlow(3, ZeroLowFlowCounter) = LowFlow(3, i)
  End If
Next i

' Create ln transformed array
ReDim LnZeroLowFlow(3, ZeroLowFlowCounter)
SumQ = 0
For i = 1 To (n - N0)
  LnZeroLowFlow(1, i) = ZeroLowFlow(1, i)
  LnZeroLowFlow(2, i) = ZeroLowFlow(2, i)
  LnZeroLowFlow(3, i) = Log(ZeroLowFlow(3, i))
  SumQ = SumQ + LnZeroLowFlow(3, i) ' Get Sum
Next i
'Calculate Statistics

ML = SumQ / (n - N0)

SumDiffSquared = 0
SumDiffCubed = 0
For i = 1 To (n - N0)
   SumDiffSquared = (LnZeroLowFlow(3, i) - ML) ^ 2 + SumDiffSquared
   SumDiffCubed = (LnZeroLowFlow(3, i) - ML) ^ 3 + SumDiffCubed
Next i

SL = (SumDiffSquared / ((n - N0) - 1)) ^ 0.5
GL = (n - N0) / ((n - N0) - 1) / ((n - N0) - 2) * SumDiffCubed / SL ^ 3

'Define Return Periods and Calculate z
TL(l, 1) = 2
TL(l, 2) = 10
TL(l, 3) = 50
For i = 1 To 3
   T = 1 / ((n / (n - N0)) * (1 - 1 / TL(l, i)))
   Call StdNorm(z, T)
   KT = 2 / GL * (1 + GL * z / 6 - GL ^ 2 / 36) ^ 3 - 2 / GL
   QT(l, i) = Exp(ML + KT * SL)
   PrintLowFlow(l, i) = QT(l, i)
Next i
Next l

'Scenario Low Flows

For l = 1 To 5

'Daily Averages
j = 0 'Annual Peaks Array Counter

i = 1 'Total Dataset Array Counter
Do While i < TotalLines + 1

  'Check for new water year
  If Mo(i) = 10 And Dy(i) = 1 And Hr(i) = 1 And Mn(i) = 0 Then
    WaterYr = Yr(i) + 1
  End If

  'Check for new day
  If Hr(i) = 1 Then 'If at a new day
    j = j + 1
    ReDim Preserve DayArray(5, j) 'Resize Array to Unknown Size
    HourSum = 0

    'Sum all flows over the day
    For k = 0 To 23
      HourSum = HourSum + q(10, i + k)
    Next k

    DayArray(1, j) = WaterYr 'Record Water Yr
    DayArray(2, j) = Yr(i) 'Record Julian Yr
    DayArray(3, j) = Mo(i) 'Record Julian Mo
    DayArray(4, j) = Dy(i) 'Record Julian Dy
    DayArray(5, j) = HourSum / 24 'Record Daily average flow
    DayArraySize = j
  End If
  i = i + 1
Loop

'Get Days(l) day averages for each year
CurrentYear = 0
LowFlowCounter = 0
For i = 1 To DayArraySize
If DayArray(1, i) <> CurrentYear Then 'Check if new water year has begun
   LowFlowCounter = LowFlowCounter + 1
   ReDim Preserve LowFlow(3, LowFlowCounter)
   Average = 10000000000# 'Set the lowest average to very high number for the upcoming year
End If
SumTemp = 0 'Set the previous Days(l) day sum to zero

'Eliminate first Day(l)-1 averages as incomplete
Select Case Days(l)
   Case Is < 32 '31 Days in October
      If DayArray(3, i) = 10 And DayArray(4, i) < Days(l) Then
         Complete = 0 'Part of Incomplete Data Set at year start
      Else
         Complete = 1 'Part of complete Data Set
      End If
   Case 32 To 61 '31 + 30 days in November
      If DayArray(3, i) = 10 Or DayArray(3, i) = 11 And DayArray(4, i) < Days(l) - 31 Then
         Complete = 0 'Part of Incomplete Data Set at year start
      Else
         Complete = 1 'Part of complete Data Set
      End If
   Case 62 To 92 '61 + 31 days in December
      If DayArray(3, i) = 10 Or DayArray(3, i) = 11 Or DayArray(3, i) = 12 And DayArray(4, i) <
        Days(l) - 61 Then
         Complete = 0 'Part of Incomplete Data Set at year start
      Else
         Complete = 1 'Part of complete Data Set
      End If
End Select

If Complete = 0 Then
Else
   For j = i - (Days(l) - 1) To i
      SumTemp = SumTemp + DayArray(5, j) 'sum the Days(i) days of average flows
   Next j
   AverageTemp = SumTemp / Days(l) 'calculate the Days(l) day average
If AverageTemp < Average Then 'check if current average is less than previous min

Average = AverageTemp 'replace the previous min with the new min
End If

' Set LowFlow current array
LowFlow(1, LowFlowCounter) = DayArray(1, i)
LowFlow(2, LowFlowCounter) = DayArray(2, i)
LowFlow(3, LowFlowCounter) = Average
End If
CurrentYear = DayArray(1, i) ' Set the current water year
Next i

' Count zeroes and create a new array
N0 = 0
n = LowFlowCounter
ZeroLowFlowCounter = 0
For i = 1 To n
   If LowFlow(3, i) = 0 Then
      N0 = N0 + 1
   Else
      ZeroLowFlowCounter = ZeroLowFlowCounter + 1
      ReDim Preserve ZeroLowFlow(3, ZeroLowFlowCounter)
      ZeroLowFlow(1, ZeroLowFlowCounter) = LowFlow(1, i)
      ZeroLowFlow(2, ZeroLowFlowCounter) = LowFlow(2, i)
      ZeroLowFlow(3, ZeroLowFlowCounter) = LowFlow(3, i)
   End If
Next i

' Create ln transformed array
ReDim LnZeroLowFlow(3, ZeroLowFlowCounter)
SumQ = 0
For i = 1 To (n - N0)
   LnZeroLowFlow(1, i) = ZeroLowFlow(1, i)
   LnZeroLowFlow(2, i) = ZeroLowFlow(2, i)
   LnZeroLowFlow(3, i) = Log(ZeroLowFlow(3, i))
   SumQ = SumQ + LnZeroLowFlow(3, i) ' Get Sum
Next i
'Calculate Statistics

ML = SumQ / (n - N0)

SumDiffSquared = 0
SumDiffCubed = 0
For i = 1 To (n - N0)
    SumDiffSquared = (LnZeroLowFlow(3, i) - ML) ^ 2 + SumDiffSquared
    SumDiffCubed = (LnZeroLowFlow(3, i) - ML) ^ 3 + SumDiffCubed
Next i

SL = (SumDiffSquared / ((n - N0) - 1)) ^ 0.5
GL = (n - N0) / ((n - N0) - 1) / ((n - N0) - 2) * SumDiffCubed / SL ^ 3

'Define Return Periods and Calculate z
TL(l, 1) = 2
TL(l, 2) = 10
TL(l, 3) = 50
For i = 1 To 3
    T = 1 / ((n / (n - N0)) * (1 - 1 / TL(l, i)))
    Call StdNorm(z, T)
    KT = 2 / GL * (1 + GL * z / 6 - GL ^ 2 / 36) ^ 3 - 2 / GL
    QT(l, i) = Exp(ML + KT * SL)
    PrintLowFlow(l, i + 3) = QT(l, i)
Next i

Next l
'Get low flow values for variability
BaseLowFlowValue = PrintLowFlow(1, 1)
End Sub

Sub Statistics()
Dim LogQ() As Single
Dim SumLogQ As Single
Dim i As Long
Dim Mean As Single 'Mean of log flows
Dim SumDiff As Single
Dim SumDiffSquared As Single
Dim SumDiffCubed As Single
Dim S As Single
Dim G As Single

Dim b As Single
Dim a As Single
Dim c As Single
Dim MaxLogQ As Single
Dim MinLogQ As Single
Dim BinNumber As Single
Dim BinSize As Single
Dim PDFPoint() As Single
Dim MidPoint() As Single
Dim gammab As Double
Dim Pi As Single
Dim LnPDFPoint As Double
Dim lngammab As Double

ReDim ChartArrayStats(2)
ReDim LogQ(BaseTotalLines)

SumLogQ = 0
MaxLogQ = 0
MinLogQ = 0

For i = 1 To BaseTotalLines
    LogQ(i) = Log(Baseq(10, i))
    SumLogQ = SumLogQ + LogQ(i)
    If LogQ(i) > MaxLogQ Then
        MaxLogQ = LogQ(i)
    End If
End For
End If
    If LogQ(i) < MinLogQ Then
        MinLogQ = LogQ(i)
    End If
Next i

Mean = SumLogQ / BaseTotalLines

SumDiff = 0
SumDiffSquared = 0
SumDiffCubed = 0

For i = 1 To BaseTotalLines
    SumDiff = SumDiff + (LogQ(i) - Mean)
    SumDiffSquared = SumDiffSquared + (LogQ(i) - Mean) ^ 2
    SumDiffCubed = SumDiffCubed + (LogQ(i) - Mean) ^ 3
Next i

S = (SumDiffSquared / (BaseTotalLines - 1)) ^ 0.5
G = (CSng(BaseTotalLines) * SumDiffCubed) / (CSng((BaseTotalLines - 1)) * CSng((BaseTotalLines - 2)) * S ^ 3)

b = 4 / G ^ 2
a = G * b ^ 0.5 / (Abs(G) * S)
c = Mean - (b / a)

'Create Bins
Pi = 3.14159
BinNumber = 20
BinSize = (MaxLogQ - MinLogQ) / BinNumber

ReDim PDFPoint(BinNumber), MidPoint(BinNumber), ChartArrayFreq(BinNumber, 4),
ChartArrayFreqShift(BinNumber, 4)
For i = 1 To BinNumber
    MidPoint(i) = BinSize * i - BinSize / 2
    'Estimate for Gamma function
lngammab = (b - 0.5) * Log(b) - b + 0.5 * Log(2 * Pi) + 1 / (12 * b) - 1 / (360 * b ^ 3) + 1 / (1260 * b ^ 5) - 1 / (1680 * b ^ 7)

LnPDFPoint = Log(Abs(a)) + (b - 1) * Log(a * (MidPoint(i) - c)) - (a * (MidPoint(i) - c)) - lngammab

PDFPoint(i) = Exp(LnPDFPoint)
ChartArrayFreq(i, 1) = MidPoint(i)
ChartArrayFreq(i, 2) = PDFPoint(i)
ChartArrayFreqShift(i, 1) = MidPoint(i) - Mean
ChartArrayFreqShift(i, 2) = PDFPoint(i)

Next i

'Scenario File
ReDim LogQ(TotalLines)
SumLogQ = 0
MaxLogQ = 0
MinLogQ = 0

For i = 1 To TotalLines
    LogQ(i) = Log(q(10, i))
    SumLogQ = SumLogQ + LogQ(i)
    If LogQ(i) > MaxLogQ Then
        MaxLogQ = LogQ(i)
    End If
    If LogQ(i) < MinLogQ Then
        MinLogQ = LogQ(i)
    End If
Next i

Mean = SumLogQ / TotalLines

SumDiff = 0
SumDiffSquared = 0
SumDiffCubed = 0

For i = 1 To TotalLines
    SumDiff = SumDiff + (LogQ(i) - Mean)
    SumDiffSquared = SumDiffSquared + (LogQ(i) - Mean) ^ 2
    SumDiffCubed = SumDiffCubed + (LogQ(i) - Mean) ^ 3
Next i
$S = \left( \frac{\text{SumDiffSquared}}{\text{TotalLines} - 1} \right)^{0.5}$

$G = \frac{\text{CSng(TotalLines)} \times \text{SumDiffCubed}}{(\text{CSng(TotalLines} - 1)) \times \text{CSng(TotalLines} - 2)) \times S^3}$

$b = \frac{4}{G^2}$

$a = G \times b^{0.5} / (\text{Abs}(G) \times S)$

$c = \text{Mean} - (b / a)$

'Create Bins
$\pi = 3.14159$

$\text{BinNumber} = 20$

$\text{BinSize} = \frac{\text{MaxLogQ} - \text{MinLogQ}}{\text{BinNumber}}$

ReDim PDFPoint(BinNumber), MidPoint(BinNumber)

For $i = 1$ To $\text{BinNumber}$
    $\text{MidPoint}(i) = \text{BinSize} \times i - \text{BinSize} / 2$
    'Estimate for Gamma function
    lngammab = $\left( b - 0.5 \right) \times \log(b) - b + 0.5 \times \log(2 \times \pi) + 1 / (12 \times b) - 1 / (360 \times b^3) + 1 / (1260 \times b^5) - 1 / (1680 \times b^7)$
    $\text{LnPDFPoint} = \log(\text{Abs}(a)) + (b - 1) \times \log(a \times (\text{MidPoint}(i) - c)) - (a \times (\text{MidPoint}(i) - c)) - \text{lngammab}$
    $\text{PDFPoint}(i) = \exp(\text{LnPDFPoint})$
    $\text{ChartArrayFreq}(i, 3) = \text{MidPoint}(i)$
    $\text{ChartArrayFreq}(i, 4) = \text{PDFPoint}(i)$
    $\text{ChartArrayFreqShift}(i, 3) = \text{MidPoint}(i) - \text{Mean}$
    $\text{ChartArrayFreqShift}(i, 4) = \text{PDFPoint}(i)$

Next $i$

End Sub

Sub Variability()
    Dim BaseCV As Single
    Dim CV As Single
    Dim StoredVal As Single
    Dim j As Long
    Dim sorted As Boolean

Sub Variability()
Dim BaseSum As Single
Dim BaseMean As Single
Dim BaseS As Single
Dim BaseSumDiffSquared As Single
Dim i As Long
Dim BaseSortedLines() As Long
Dim Sum As Single
Dim Mean As Single
Dim S As Single
Dim SumDiffSquared As Single
Dim SortedLines() As Long
Dim k As Long
Dim k2 As Integer
Dim ArrayCount As Integer
Dim Fileline As String
Dim PreSortedArray() As Single
Dim SortedArray() As Single
Dim MedianPosition As Long
Dim Median As Single
Dim BaseQuartile() As Single
Dim BaseQuartileRange As Single
Dim Quartile() As Single
Dim QuartileRange As Single
Dim BaseQuantile() As Single
Dim temp As Boolean
Dim BaseqMax As Long
Dim BaseLowFlowEventCount As Integer
Dim BaseContinue As Integer
Dim BaseLowFlowDuration As Long
Dim qMax As Long
Dim LowFlowEventCount As Integer
Dim Continue As Integer
Dim LowFlowDuration As Long
'Calculate regular flow mean
BaseSum = 0
For i = 1 To BaseTotalLines
    BaseSum = BaseSum + Baseq(10, i)
Next i
BaseMean = BaseSum / BaseTotalLines

For i = 1 To BaseTotalLines
    BaseSumDiffSquared = BaseSumDiffSquared + (Baseq(10, i) - BaseMean) ^ 2
Next i
BaseS = (BaseSumDiffSquared / (BaseTotalLines - 1)) ^ 0.5
BaseCV = BaseS / BaseMean

'Quantiles and Median
'Prepare a sorted q array
Open "c:\HSPFPost\BaseQvalues.dat" For Output As #3
For i = 1 To BaseTotalLines
    Print #3, Baseq(10, i)
Next i
Close #3

'temp = Shell("c:\HSPFPost\Qvalues.bat", vbHide) 'Text sorts the array (Presort)
temp = SyncShell("c:\HSPFPost\BaseQvalues.bat", vbNormal) 'Text sorts the array (Presort)

Open "c:\HSPFPost\BaseQvalues.out" For Input As #4

'Sorts the presorted data
ReDim SortedArray(10, BaseTotalLines)
ReDim PreSortedArray(10, BaseTotalLines)
For i = 1 To BaseTotalLines
    Line Input #4, Fileline
    Read Variables From File
PreSortedArray(10, i) = val(Trim(Mid(Fileline, 1, 15)))

Next i
Close #4
'ProgressBar1.Value = 0

j = 0
For k = 1 To 7
    For k2 = 1 To 9
        For i = 1 To BaseTotalLines
            If k = 1 Then
                If PreSortedArray(10, i) < 1 And k2 = 1 Then
                    j = j + 1
                    SortedArray(10, j) = PreSortedArray(10, i)
                End If
            Else
                If PreSortedArray(10, i) < ((k2 + 1) * 10 ^ (k - 2)) And PreSortedArray(10, i) >= (k2 * 10 ^ (k - 2)) Then
                    j = j + 1
                    SortedArray(10, j) = PreSortedArray(10, i)
                End If
            End If
        Next i
    Next k2
Next k

'Obtain FRE3
MedianPosition = Int(BaseTotalLines / 2)
Median = SortedArray(10, MedianPosition)
FRE3 = 3 * Median

'Obtain quartiles
ReDim BaseQuartile(2)
ReDim BaseQuantile(BaseTotalLines)

'Obtain Low Flow Duration and Count
BaseqMax = 0
For i = 1 To BaseTotalLines
    If i > BaseqMax Then 'Check to see if value has already been evaluated
        If Baseq(10, i) <= BaseLowFlowValue Then
            Baseq(10, i) = 1
            BaseqMax = i
        Else
            Baseq(10, i) = 0
        End If
    End If
Next i
BaseLowFlowEventCount = BaseLowFlowEventCount + 1
'Check for length of time low flow event occurs
BaseContinue = 1
k = i
Do While BaseContinue = 1
    Do While Baseq(10, k) <= BaseLowFlowValue And k <> BaseTotalLines
        BaseLowFlowDuration = BaseLowFlowDuration + 1
        k = k + 1
    Loop
    BaseqMax = k - 1
    BaseContinue = 0
    For j = 1 To 24
        If Baseq(10, k - 1 + j) <= BaseLowFlowValue And (k - 1 + j) <> BaseTotalLines Then
            BaseContinue = 1
        End If
        If (k - 1 + j) = BaseTotalLines Then j = 25 'exit loop if at end of array
    Next j
    If BaseContinue = 1 Then
        If Baseq(10, k) > BaseLowFlowValue Then
            BaseLowFlowDuration = BaseLowFlowDuration + 1 'Add to duration for values above
        End If
        k = k + 1 'Advance counter
    End If
Loop
End If
Next i
For i = 2 To BaseTotalLines
    BaseQuantile(i) = (i - 0.5) / BaseTotalLines
    If BaseQuantile(i) = 0.1 Then
        BaseQuartile(1) = SortedArray(10, i)
    Else
        If BaseQuantile(i) = 0.9 Then
            BaseQuartile(2) = SortedArray(10, i)
        Else
            If BaseQuantile(i - 1) < 0.1 And BaseQuantile(i) > 0.1 Then
                BaseQuartile(3) = SortedArray(10, i)
            Else
                BaseQuartile(4) = SortedArray(10, i)
            End If
        End If
    End If
Next i
BaseQuartile(1) = SortedArray(10, i) - (BaseQuantile(i) - 0.1) / (BaseQuantile(i) - 
BaseQuantile(i - 1)) * (SortedArray(10, i) - SortedArray(10, i - 1))
Else
    If BaseQuantile(i - 1) < 0.9 And BaseQuantile(i) > 0.1 Then
        BaseQuartile(2) = SortedArray(10, i) - (BaseQuantile(i) - 0.9) / (BaseQuantile(i) - 
        BaseQuantile(i - 1)) * (SortedArray(10, i) - SortedArray(10, i - 1))
    End If
End If
End If
Next i
BaseQuartileRange = BaseQuartile(2) - BaseQuartile(1)

'Scenario
'Calculate regular flow mean
Sum = 0
For i = 1 To TotalLines
    Sum = Sum + q(10, i)
Next i
Mean = Sum / TotalLines
For i = 1 To TotalLines
    SumDiffSquared = SumDiffSquared + (q(10, i) - Mean) ^ 2
Next i
S = (SumDiffSquared / (TotalLines - 1)) ^ 0.5
CV = S / Mean

'Quantiles and Median
'Prepare a sorted q array
Open "c:\HSPFPost\Qvalues.dat" For Output As #3
For i = 1 To TotalLines
    Print #3, q(10, i)
Next i
Close #3

temp = SyncShell("c:\HSPFPost\Qvalues.bat", vbHide) 'Text sorts the array (Presort)
Open "c:\HSPFPost\Qvalues.out" For Input As #4

'Sorts the presorted data
ReDim SortedArray(10, TotalLines)
ReDim PreSortedArray(10, TotalLines)
For i = 1 To TotalLines
 'ProgressBar1.Value = (i / j * 100)
Line Input #4, Fileline

 'Read Variables From File
 PreSortedArray(10, i) = val(Trim(Mid(Fileline, 1, 15)))
Next i
Close #4
 'ProgressBar1.Value = 0

j = 0
For k = 1 To 7
 For k2 = 1 To 9
  For i = 1 To TotalLines
   If k = 1 Then
    If PreSortedArray(10, i) < 1 And k2 = 1 Then
     j = j + 1
     SortedArray(10, j) = PreSortedArray(10, i)
    End If
   Else
    If PreSortedArray(10, i) < ((k2 + 1) * 10 ^ (k - 2)) And PreSortedArray(10, i) >= (k2 * 10 ^ (k - 2)) Then
     j = j + 1
     SortedArray(10, j) = PreSortedArray(10, i)
    End If
   End If
  Next i
 Next k2
Next k
'Obtain quartiles
ReDim Quartile(2)
ReDim Quantile(TotalLines)
'Obtain Low Flow Duration and Count
qMax = 0
For i = 1 To TotalLines
    If i > qMax Then 'Check to see if value has already been evaluated
        If q(10, i) <= BaseLowFlowValue Then
            LowFlowEventCount = LowFlowEventCount + 1
            'Check for length of time low flow event occurs
            Continue = 1
            k = i
            Do While Continue = 1
                Do While q(10, k) <= BaseLowFlowValue And k <> TotalLines
                    LowFlowDuration = LowFlowDuration + 1
                    k = k + 1
                Loop
                qMax = k - 1
                Continue = 0
                For j = 1 To 24
                    If q(10, k - 1 + j) <= BaseLowFlowValue And (k - 1 + j) <> TotalLines Then
                        Continue = 1
                    End If
                If (k - 1 + j) = TotalLines Then j = 25 'exit loop if at end of array
                Next j
                If Continue = 1 Then
                    If q(10, k) > BaseLowFlowValue Then
                        LowFlowDuration = LowFlowDuration + 1 'Add to duration for values above BaseLowFlowValue during event
                    End If
                k = k + 1 'Advance counter
            End If
            'Advance counter
        End If
    End If
Next i
For i = 2 To TotalLines
    Quantile(i) = (i - 0.5) / TotalLines
If Quantile(i) = 0.1 Then
  Quartile(1) = SortedArray(10, i)
Else
  If Quantile(i) = 0.9 Then
    Quartile(2) = SortedArray(10, i)
  Else
    If Quantile(i - 1) < 0.1 And Quantile(i) > 0.1 Then
      Quartile(1) = SortedArray(10, i) - (Quantile(i) - 0.1) / (Quantile(i) - Quantile(i - 1)) * (SortedArray(10, i) - SortedArray(10, i - 1))
    Else
      If Quantile(i - 1) < 0.9 And Quantile(i) > 0.9 Then
        Quartile(2) = SortedArray(10, i) - (Quantile(i) - 0.9) / (Quantile(i) - Quantile(i - 1)) * (SortedArray(10, i) - SortedArray(10, i - 1))
      End If
    End If
  End If
End If
Next i
QuartileRange = Quartile(2) - Quartile(1)

ReDim VariabilityArray(2, 7)
VariabilityArray(1, 1) = BaseCV
VariabilityArray(1, 2) = BaseQuartileRange
VariabilityArray(1, 6) = BaseLowFlowEventCount
VariabilityArray(1, 7) = BaseLowFlowDuration
VariabilityArray(2, 1) = CV
VariabilityArray(2, 2) = QuartileRange
VariabilityArray(2, 6) = LowFlowEventCount
VariabilityArray(2, 7) = LowFlowDuration

temp = SyncShell("c:\HSPFPost\remove1.bat", vbHide) 'Remove data files

'Shrink arrays to save memory
ReDim SortedArray(1, 1)
ReDim PreSortedArray(1, 1)
End Sub
Sub ScenarioHSI()
Dim i As Long
Dim j As Long
Dim j2() As Long
Dim FtableQ() As Single
Dim FtableD() As Single
Dim d As Single
Dim TW As Single
Dim IW As Single
Dim b() As Single
Dim z() As Single
Dim k As Integer
Dim n() As Single
Dim NumSlices As Integer
Dim SliceD() As Single
Dim SliceA() As Single
Dim SliceWP() As Single
Dim SliceR() As Single
Dim SliceK() As Single
Dim Ratio() As Single
Dim SliceV() As Single
Dim RectA As Single
Dim RectWP As Single
Dim RectR As Single
Dim RectK As Single
Dim RectV As Single
Dim TotalK As Single
Dim VelocityArray() As Single
Dim DepthArray() As Single
Dim AreaArray() As Single
Dim k2 As Integer
Dim k3 As Integer
Dim k4 As Integer
Dim Fileline As String
Dim FtableLines() As Integer
Dim AcceptTimeV() As Single
Dim OptTimeV() As Single
Dim AcceptTimeD() As Single
Dim OptTimeD() As Single
Dim temp As Boolean

Dim SumOptTimeV() As Single
Dim SumAcceptTimeV() As Single
Dim SumOptTimeD() As Single
Dim SumAcceptTimeD() As Single
Dim SumOptTimeDV() As Single
Dim SumAcceptTimeDV() As Single

Dim SumOptAreaV() As Single
Dim SumAcceptAreaV() As Single
Dim SumOptAreaD() As Single
Dim SumAcceptAreaD() As Single
Dim SumOptAreaDV() As Single
Dim SumAcceptAreaDV() As Single

Dim ReachSumOptTimeV() As Single
Dim ReachSumAcceptTimeV() As Single
Dim ReachSumOptTimeD() As Single
Dim ReachSumAcceptTimeD() As Single
Dim ReachSumOptTimeDV() As Single
Dim ReachSumAcceptTimeDV() As Single

Dim StreamOptAreaV() As Single
Dim StreamAcceptAreaV() As Single
Dim StreamOptAreaD() As Single
Dim StreamAcceptAreaD() As Single
Dim StreamOptAreaDV() As Single
Dim StreamAcceptAreaDV() As Single

Dim StreamSumOptTimeV As Single
Dim StreamSumAcceptTimeV As Single
Dim StreamSumOptTimeD As Single
Dim StreamSumAcceptTimeD As Single
Dim StreamSumOptTimeDV As Single
Dim StreamSumAcceptTimeDV As Single
Dim StreamTotalArea() As Single
Dim TotalArea() As Single

Dim SortedReachOptAreaV() As Single
Dim SortedReachAcceptAreaV() As Single

Dim SortedReachOptAreaD() As Single
Dim SortedReachAcceptAreaD() As Single

Dim SortedReachOptAreaDV() As Single
Dim SortedReachAcceptAreaDV() As Single

Dim SortedStreamOptAreaV() As Single
Dim SortedStreamAcceptAreaV() As Single

Dim SortedStreamOptAreaD() As Single
Dim SortedStreamAcceptAreaD() As Single

Dim SortedStreamOptAreaDV() As Single
Dim SortedStreamAcceptAreaDV() As Single

Dim Quantile() As Single

Dim SortedStreamOptAreaVZero As Long
Dim SortedStreamAcceptAreaVZero As Long
Dim SortedStreamOptAreaDZero As Long
Dim SortedStreamAcceptAreaDZero As Long
Dim SortedStreamOptAreaDVZero As Long
Dim SortedStreamAcceptAreaDVZero As Long

Dim SortedReachOptAreaVZero() As Long
Dim SortedReachAcceptAreaVZero() As Long
Dim SortedReachOptAreaDZero() As Long
Dim SortedReachAcceptAreaDZero() As Long
Dim SortedReachOptAreaDVZero() As Long
Dim SortedReachAcceptAreaDVZero() As Long
NumSlices = 4

ReDim StreamOptAreaV(TotalLines)
ReDim StreamAcceptAreaV(TotalLines)
ReDim StreamOptAreaD(TotalLines)
ReDim StreamAcceptAreaD(TotalLines)
ReDim StreamOptAreaDV(TotalLines)
ReDim StreamAcceptAreaDV(TotalLines)

ReDim AcceptTimeV(10, TotalLines, NumSlices + 1)
ReDim OptTimeV(10, TotalLines, NumSlices + 1)
ReDim AcceptTimeD(10, TotalLines, NumSlices + 1)
ReDim OptTimeD(10, TotalLines, NumSlices + 1)

ReDim SliceD(NumSlices), SliceA(NumSlices), SliceWP(NumSlices)
ReDim SliceR(NumSlices), SliceK(NumSlices), SliceV(NumSlices)
ReDim Ratio(NumSlices)
ReDim VelocityArray(10, TotalLines, NumSlices + 1)
ReDim DepthArray(10, TotalLines, NumSlices + 1)
ReDim AreaArray(10, TotalLines, NumSlices + 1)

ReDim SortedReachOptAreaV(10, TotalLines)
ReDim SortedReachAcceptAreaV(10, TotalLines)

ReDim SortedReachOptAreaD(10, TotalLines)
ReDim SortedReachAcceptAreaD(10, TotalLines)

ReDim SortedReachOptAreaDV(10, TotalLines)
ReDim SortedReachAcceptAreaDV(10, TotalLines)

ReDim SortedStreamOptAreaV(TotalLines)
ReDim SortedStreamAcceptAreaV(TotalLines)

ReDim SortedStreamOptAreaD(TotalLines)
ReDim SortedStreamAcceptAreaD(TotalLines)

ReDim SortedStreamOptAreaDV(TotalLines)
ReDim SortedStreamAcceptAreaDV(TotalLines)
ReDim FtableD(10, 20), FtableQ(10, 20), FtableLines(10)

ReDim Quantile(TotalLines)

ReDim StreamPercentOutput(6)
ReDim ReachPercentOutput(10, 6)

ReDim ReachPercentOptTimeV(10)
ReDim ReachPercentAcceptTimeV(10)
ReDim ReachPercentOptTimeD(10)
ReDim ReachPercentAcceptTimeD(10)
ReDim ReachPercentOptTimeDV(10)
ReDim ReachPercentAcceptTimeDV(10)

ReDim z(10), b(10), n(10)

Open "c:\HSPFPost\FTABLES.csv" For Input As #6

For j = 1 To 10
    Input #6, FtableLines(j)
    Input #6, z(j), b(j), n(j)
    For i = 1 To FtableLines(j)
        Input #6, FtableD(j, i), FtableQ(j, i)
    Next i
Next j
Close #6

For k2 = 1 To 10
    For i = 1 To TotalLines 'Use all data
        For j = 2 To FtableLines(k2) 'Points in each Ftable
            If q(k2, i) <= FtableQ(k2, j) And q(k2, i) > FtableQ(k2, j - 1) Then 'Find where Q falls on Ftable
                d = FtableD(k2, j) - ((FtableQ(k2, j) - q(k2, i)) * (FtableD(k2, j) - FtableD(k2, j - 1)) / (FtableQ(k2, j) - FtableQ(k2, j - 1)))
                TW = b(k2) + 2 * d * z(k2) 'Top width
IW = (TW - b(k2)) / (2 * NumSlices) 'Slice widths
For k = 1 To NumSlices
    SliceD(k) = d - ((IW / z(k2) * k) + (IW / z(k2) * (k - 1))) / 2
    SliceA(k) = ((d - k * IW / z(k2)) * IW) + (IW / 2) * ((d - (IW / z(k2) * (k - 1))) - (d - (IW / z(k2) * k)))
    SliceWP(k) = (IW ^ 2 + ((d - IW / z(k2) * (k - 1)) - (d - IW / z(k2) * k)) ^ 2) ^ 0.5
    SliceR(k) = SliceA(k) / SliceWP(k)
    SliceK(k) = 1.49 / n(k2) * SliceA(k) * SliceR(k) ^ (2 / 3)
Next k
RectA = b(k2) * d
RectWP = b(k2)
RectK = RectA / RectWP
RectR = RectK ^ (2 / 3)
TotalK = 0
For k = 1 To NumSlices
    TotalK = SliceK(k) * 2 + TotalK '2 channel sides
Next k
TotalK = TotalK + RectK
For k = 1 To NumSlices
    Ratio(k) = SliceK(k) / TotalK
    SliceV(k) = (Ratio(k) * q(k2, i)) / SliceA(k)
Next k
RectV = (RectK / TotalK * q(k2, i)) / RectA
'Store values in arrays
For k = 1 To NumSlices
    VelocityArray(k2, i, k) = SliceV(k)
    DepthArray(k2, i, k) = SliceD(k)
    AreaArray(k2, i, k) = SliceA(k) * 2 '2 channel sides
Next k
VelocityArray(k2, i, NumSlices + 1) = RectV
DepthArray(k2, i, NumSlices + 1) = d
AreaArray(k2, i, NumSlices + 1) = RectA
End If
Next j
Next i
Next k2
ReDim TotalArea(10, TotalLines), StreamTotalArea(TotalLines)
For k2 = 1 To 10
For i = 1 To TotalLines
    For k = 1 To NumSlices + 1
        TotalArea(k2, i) = AreaArray(k2, i, k) + TotalArea(k2, i)
    Next k
Next i
Next k2
For k2 = 1 To 10
    For i = 1 To TotalLines
        StreamTotalArea(i) = TotalArea(k2, i) + StreamTotalArea(i)
    Next i
Next k2
For i = 1 To TotalLines
    For k2 = 1 To 10
        For j = 1 To NumSlices + 1
            If VelocityArray(k2, i, j) > OptimumV(1) And VelocityArray(k2, i, j) < OptimumV(2) Then
                OptTimeV(k2, i, j) = 1  'If optimum velocity is available record a 1 at that time
            End If
            If VelocityArray(k2, i, j) > AcceptableV(1) And VelocityArray(k2, i, j) < AcceptableV(2) Then
                AcceptTimeV(k2, i, j) = 1  'If acceptable velocity is available record a 1 at that time
            End If
            If DepthArray(k2, i, j) > OptimumD(1) And DepthArray(k2, i, j) < OptimumD(2) Then
                OptTimeD(k2, i, j) = 1  'If optimum depth is available record a 1 at that time
            End If
            If DepthArray(k2, i, j) > AcceptableD(1) And DepthArray(k2, i, j) < AcceptableD(2) Then
                AcceptTimeD(k2, i, j) = 1  'If acceptable depth is available record a 1 at that time
            End If
        Next j
    Next k2
Next i
ReDim SumOptTimeV(10, TotalLines)
ReDim SumAcceptTimeV(10, TotalLines)
ReDim SumOptTimeD(10, TotalLines)
ReDim SumAcceptTimeD(10, TotalLines)
ReDim SumOptTimeDV(10, TotalLines)
ReDim SumAcceptTimeDV(10, TotalLines)

ReDim SumOptAreaV(10, TotalLines)
ReDim SumAcceptAreaV(10, TotalLines)
ReDim SumOptAreaD(10, TotalLines)
ReDim SumAcceptAreaD(10, TotalLines)
ReDim SumOptAreaDV(10, TotalLines)
ReDim SumAcceptAreaDV(10, TotalLines)

' % Time loop
For i = 1 To TotalLines
    For k2 = 1 To 10
        For j = 1 To NumSlices + 1
            If OptTimeV(k2, i, j) = 1 Then
                ' this will allow for values higher than 1
                SumOptTimeV(k2, i) = SumOptTimeV(k2, i) + 1
                SumOptAreaV(k2, i) = SumOptAreaV(k2, i) + AreaArray(k2, i, j) / TotalArea(k2, i)
            End If
            If AcceptTimeV(k2, i, j) = 1 Then
                SumAcceptTimeV(k2, i) = SumAcceptTimeV(k2, i) + 1
                SumAcceptAreaV(k2, i) = SumAcceptAreaV(k2, i) + AreaArray(k2, i, j) / TotalArea(k2, i)
            End If
            If OptTimeD(k2, i, j) = 1 Then
                SumOptTimeD(k2, i) = SumOptTimeD(k2, i) + 1
                SumOptAreaD(k2, i) = SumOptAreaD(k2, i) + AreaArray(k2, i, j) / TotalArea(k2, i)
            End If
            If AcceptTimeD(k2, i, j) = 1 Then
                SumAcceptTimeD(k2, i) = SumAcceptTimeD(k2, i) + 1
                SumAcceptAreaD(k2, i) = SumAcceptAreaD(k2, i) + AreaArray(k2, i, j) / TotalArea(k2, i)
            End If
            If OptTimeV(k2, i, j) = 1 And OptTimeD(k2, i, j) = 1 Then
                ' This condition is met
            End If
        Next j
    Next k2
Next i
SumOptTimeDV(k2, i) = SumOptTimeDV(k2, i) + 1
SumOptAreaDV(k2, i) = SumOptAreaDV(k2, i) + AreaArray(k2, i, j) / TotalArea(k2, i)

End If

If AcceptTimeV(k2, i, j) = 1 And OptTimeD(k2, i, j) = 1 Then
SumAcceptTimeDV(k2, i) = SumAcceptTimeDV(k2, i) + 1
SumAcceptAreaDV(k2, i) = SumAcceptAreaDV(k2, i) + AreaArray(k2, i, j) / TotalArea(k2, i)

End If

Next j

Next k2
Next i

ReDim ReachSumOptTimeV(10)
ReDim ReachSumAcceptTimeV(10)
ReDim ReachSumOptTimeD(10)
ReDim ReachSumAcceptTimeD(10)
ReDim ReachSumOptTimeDV(10)
ReDim ReachSumAcceptTimeDV(10)

For i = 1 To TotalLines
For k2 = 1 To 10
    If SumOptTimeV(k2, i) > 0 Then
        ReachSumOptTimeV(k2) = ReachSumOptTimeV(k2) + 1
    End If
    If SumAcceptTimeV(k2, i) > 0 Then
        ReachSumAcceptTimeV(k2) = ReachSumAcceptTimeV(k2) + 1
    End If
    If SumOptTimeD(k2, i) > 0 Then
        ReachSumOptTimeD(k2) = ReachSumOptTimeD(k2) + 1
    End If
    If SumAcceptTimeD(k2, i) > 0 Then
        ReachSumAcceptTimeD(k2) = ReachSumAcceptTimeD(k2) + 1

Next k2
Next i
End If
If SumOptTimeDV(k2, i) > 0 Then
    ReachSumOptTimeDV(k2) = ReachSumOptTimeDV(k2) + 1
End If
If SumAcceptTimeDV(k2, i) > 0 Then
    ReachSumAcceptTimeDV(k2) = ReachSumAcceptTimeDV(k2) + 1
End If
StreamOptAreaV(i) = SumOptAreaV(k2, i) * TotalArea(k2, i) / StreamTotalArea(i) + StreamOptAreaV(i)
StreamAcceptAreaV(i) = SumAcceptAreaV(k2, i) * TotalArea(k2, i) / StreamTotalArea(i) + StreamAcceptAreaV(i)
StreamOptAreaD(i) = SumOptAreaD(k2, i) * TotalArea(k2, i) / StreamTotalArea(i) + StreamOptAreaD(i)
StreamAcceptAreaD(i) = SumAcceptAreaD(k2, i) * TotalArea(k2, i) / StreamTotalArea(i) + StreamAcceptAreaD(i)
StreamOptAreaDV(i) = SumOptAreaDV(k2, i) * TotalArea(k2, i) / StreamTotalArea(i) + StreamOptAreaDV(i)
StreamAcceptAreaDV(i) = SumAcceptAreaDV(k2, i) * TotalArea(k2, i) / StreamTotalArea(i) + StreamAcceptAreaDV(i)

Next k2
Next i
ReDim AcceptTimeV(1, 1, 1)
ReDim OptTimeV(1, 1, 1)
ReDim AcceptTimeD(1, 1, 1)
ReDim OptTimeD(1, 1, 1)

ReDim SliceD(1), SliceA(1), SliceWP(1)
ReDim SliceR(1), SliceK(1), SliceV(1)
ReDim Ratio(1)
ReDim VelocityArray(1, 1, 1)
ReDim DepthArray(1, 1, 1)
ReDim AreaArray(1, 1, 1)

For k2 = 1 To 10
    StreamSumOptTimeV = ReachSumOptTimeV(k2) + StreamSumOptTimeV
StreamSumAcceptTimeV = ReachSumAcceptTimeV(k2) + StreamSumAcceptTimeV
StreamSumOptTimeD = ReachSumOptTimeD(k2) + StreamSumOptTimeD
StreamSumAcceptTimeD = ReachSumAcceptTimeD(k2) + StreamSumAcceptTimeD
StreamSumOptTimeDV = ReachSumOptTimeDV(k2) + StreamSumOptTimeDV
StreamSumAcceptTimeDV = ReachSumAcceptTimeDV(k2) + StreamSumAcceptTimeDV

ReachPercentOptTimeV(k2) = ReachSumOptTimeV(k2) / i
ReachPercentAcceptTimeV(k2) = ReachSumAcceptTimeV(k2) / i
ReachPercentOptTimeD(k2) = ReachSumOptTimeD(k2) / i
ReachPercentAcceptTimeD(k2) = ReachSumAcceptTimeD(k2) / i
ReachPercentOptTimeDV(k2) = ReachSumOptTimeDV(k2) / i
ReachPercentAcceptTimeDV(k2) = ReachSumAcceptTimeDV(k2) / i

Next k2

StreamPercentOptTimeV = StreamSumOptTimeV / (i * 10)
StreamPercentAcceptTimeV = StreamSumAcceptTimeV / (i * 10)
StreamPercentOptTimeD = StreamSumOptTimeD / (i * 10)
StreamPercentAcceptTimeD = StreamSumAcceptTimeD / (i * 10)
StreamPercentOptTimeDV = StreamSumOptTimeDV / (i * 10)
StreamPercentAcceptTimeDV = StreamSumAcceptTimeDV / (i * 10)

'Sort Arrays
ReDim SortedReachOptAreaVZero(10)
ReDim SortedReachAcceptAreaVZero(10)
ReDim SortedReachOptAreaDZero(10)
ReDim SortedReachAcceptAreaDZero(10)
ReDim SortedReachOptAreaDVZero(10)
ReDim SortedReachAcceptAreaDVZero(10)
ReDim j2(6)

For k2 = 1 To 10
    For k = 1 To 6
        j2(k) = 0
    Next k
Next \( k \)
For \( k = 0 \) To 100
    For \( i = 1 \) To TotalLines
        If Round(SumAcceptAreaV(k2, i), 2) = \( k \) / 100 Then
            \( j2(1) = j2(1) + 1 \)
            SortedReachAcceptAreaV(k2, j2(1)) = Round(SumAcceptAreaV(k2, i), 2)
        End If
        If Round(SumOptAreaV(k2, i), 2) = \( k \) / 100 Then
            \( j2(2) = j2(2) + 1 \)
            SortedReachOptAreaV(k2, j2(2)) = Round(SumOptAreaV(k2, i), 2)
        End If
        If Round(SumAcceptAreaD(k2, i), 2) = \( k \) / 100 Then
            \( j2(3) = j2(3) + 1 \)
            SortedReachAcceptAreaD(k2, j2(3)) = Round(SumAcceptAreaD(k2, i), 2)
        End If
        If Round(SumOptAreaD(k2, i), 2) = \( k \) / 100 Then
            \( j2(4) = j2(4) + 1 \)
            SortedReachOptAreaD(k2, j2(4)) = Round(SumOptAreaD(k2, i), 2)
        End If
        If Round(SumAcceptAreaDV(k2, i), 2) = \( k \) / 100 Then
            \( j2(5) = j2(5) + 1 \)
            SortedReachAcceptAreaDV(k2, j2(5)) = Round(SumAcceptAreaDV(k2, i), 2)
        End If
        If Round(SumOptAreaDV(k2, i), 2) = \( k \) / 100 Then
            \( j2(6) = j2(6) + 1 \)
            SortedReachOptAreaDV(k2, j2(6)) = Round(SumOptAreaDV(k2, i), 2)
        End If
    Next i
Next \( k \)
Next k2
For \( k = 1 \) To 6
    \( j2(k) = 0 \)
Next \( k \)
For \( k = 0 \) To 100
    For \( i = 1 \) To TotalLines
        If Round(StreamAcceptAreaV(i), 2) = \( k \) / 100 Then
            \( j2(1) = j2(1) + 1 \)
            SortedStreamAcceptAreaV(j2(1)) = Round(StreamAcceptAreaV(i), 2)
        End If
    Next i
Next \( k \)
End If
If Round(StreamOptAreaV(i), 2) = k / 100 Then
    j2(2) = j2(2) + 1
    SortedStreamOptAreaV(j2(2)) = Round(StreamOptAreaV(i), 2)
End If
If Round(StreamAcceptAreaD(i), 2) = k / 100 Then
    j2(3) = j2(3) + 1
    SortedStreamAcceptAreaD(j2(3)) = Round(StreamAcceptAreaD(i), 2)
End If
If Round(StreamOptAreaD(i), 2) = k / 100 Then
    j2(4) = j2(4) + 1
    SortedStreamOptAreaD(j2(4)) = Round(StreamOptAreaD(i), 2)
End If
If Round(StreamAcceptAreaDV(i), 2) = k / 100 Then
    j2(5) = j2(5) + 1
    SortedStreamAcceptAreaDV(j2(5)) = Round(StreamAcceptAreaDV(i), 2)
End If
If Round(StreamOptAreaDV(i), 2) = k / 100 Then
    j2(6) = j2(6) + 1
    SortedStreamAcceptAreaDV(j2(6)) = Round(StreamOptAreaDV(i), 2)
End If
Next i
Next k

'Set Up Quantiles
'Stream Arrays
For i = 1 To TotalLines
    Quantile(i) = (i - 0.5) / TotalLines
Next i

ReDim ChartSortedReachOptAreaV(10, 2, TotalLines)
ReDim ChartSortedReachAcceptAreaV(10, 2, TotalLines)
ReDim ChartSortedReachOptAreaD(10, 2, TotalLines)
ReDim ChartSortedReachAcceptAreaD(10, 2, TotalLines)
ReDim ChartSortedReachOptAreaDV(10, 2, TotalLines)
ReDim ChartSortedReachAcceptAreaDV(10, 2, TotalLines)
For i = 1 To TotalLines - 1
    If SortedStreamOptAreaV(i) = 0 And SortedStreamOptAreaV(i + 1) > 0 Then
        ChartSortedStreamAreaV(1, 8) = Quantile(i)
        ChartSortedStreamAreaV(1, 7) = SortedStreamOptAreaV(i)
        SortedStreamOptAreaVZero = i
    End If
Next i
For i = 1 To TotalLines - 1
    If SortedStreamAcceptAreaV(i) = 0 And SortedStreamAcceptAreaV(i + 1) > 0 Then
        ChartSortedStreamAreaV(1, 4) = Quantile(i)
        ChartSortedStreamAreaV(1, 3) = SortedStreamAcceptAreaV(i)
        SortedStreamAcceptAreaVZero = i
    End If
Next i
For i = 1 To TotalLines - 1
    If SortedStreamOptAreaD(i) = 0 And SortedStreamOptAreaD(i + 1) > 0 Then
        ChartSortedStreamAreaD(1, 8) = Quantile(i)
        ChartSortedStreamAreaD(1, 7) = SortedStreamOptAreaD(i)
        SortedStreamOptAreaDZero = i
    End If
Next i
For i = 1 To TotalLines - 1
    If SortedStreamAcceptAreaD(i) = 0 And SortedStreamAcceptAreaD(i + 1) > 0 Then
        ChartSortedStreamAreaD(1, 4) = Quantile(i)
        ChartSortedStreamAreaD(1, 3) = SortedStreamAcceptAreaD(i)
        SortedStreamAcceptAreaDZero = i
    End If
Next i
End If

Next i
For i = 1 To TotalLines - 1

If SortedStreamOptAreaDV(i) = 0 And SortedStreamOptAreaDV(i + 1) > 0 Then
    ChartSortedStreamAreaDV(1, 8) = Quantile(i)
    ChartSortedStreamAreaDV(1, 7) = SortedStreamOptAreaDV(i)
    SortedStreamOptAreaDVZero = i
End If

Next i
For i = 1 To TotalLines - 1

If SortedStreamAcceptAreaDV(i) = 0 And SortedStreamAcceptAreaDV(i + 1) > 0 Then
    ChartSortedStreamAreaDV(1, 4) = Quantile(i)
    ChartSortedStreamAreaDV(1, 3) = SortedStreamAcceptAreaDV(i)
    SortedStreamAcceptAreaDVZero = i
End If

Next i
'Reach Arrays
For k2 = 1 To 10
    For i = 1 To TotalLines - 1

        If SortedReachOptAreaV(k2, i) = 0 And SortedReachOptAreaV(k2, i + 1) > 0 Then
            ChartSortedReachOptAreaV(k2, 2, 1) = Quantile(i)
            ChartSortedReachOptAreaV(k2, 1, 1) = SortedReachOptAreaV(k2, i)
            SortedReachOptAreaVZero(k2) = i
        End If

        Next i

    Next k2
For k2 = 1 To 10
  For i = 1 To TotalLines - 1
    If SortedReachAcceptAreaV(k2, i) = 0 And SortedReachAcceptAreaV(k2, i + 1) > 0 Then
      ChartSortedReachAcceptAreaV(k2, 2, 1) = Quantile(i)
      ChartSortedReachAcceptAreaV(k2, 1, 1) = SortedReachAcceptAreaV(k2, i)
      SortedReachAcceptAreaVZero(k2) = i
    End If
  Next i
Next k2

For k2 = 1 To 10
  For i = 1 To TotalLines - 1
    If SortedReachOptAreaD(k2, i) = 0 And SortedReachOptAreaD(k2, i + 1) > 0 Then
      ChartSortedReachOptAreaD(k2, 2, 1) = Quantile(i)
      ChartSortedReachOptAreaD(k2, 1, 1) = SortedReachOptAreaD(k2, i)
      SortedReachOptAreaDZero(k2) = i
    End If
  Next i
Next k2

For k2 = 1 To 10
  For i = 1 To TotalLines - 1
    If SortedReachAcceptAreaD(k2, i) = 0 And SortedReachAcceptAreaD(k2, i + 1) > 0 Then
      ChartSortedReachAcceptAreaD(k2, 2, 1) = Quantile(i)
      ChartSortedReachAcceptAreaD(k2, 1, 1) = SortedReachAcceptAreaD(k2, i)
      SortedReachAcceptAreaDZero(k2) = i
    End If
  Next i
Next k2
Next k2
For k2 = 1 To 10
    For i = 1 To TotalLines - 1
        If SortedReachOptAreaDV(k2, i) = 0 And SortedReachOptAreaDV(k2, i + 1) > 0 Then
            ChartSortedReachOptAreaDV(k2, 2, 1) = Quantile(i)
            ChartSortedReachOptAreaDV(k2, 1, 1) = SortedReachOptAreaDV(k2, i)
            SortedReachOptAreaDVZero(k2) = i
        End If
    Next i
Next k2
For k2 = 1 To 10
    For i = 1 To TotalLines - 1
        If SortedReachAcceptAreaDV(k2, i) = 0 And SortedReachAcceptAreaDV(k2, i + 1) > 0 Then
            ChartSortedReachAcceptAreaDV(k2, 2, 1) = Quantile(i)
            ChartSortedReachAcceptAreaDV(k2, 1, 1) = SortedReachAcceptAreaDV(k2, i)
            SortedReachAcceptAreaDVZero(k2) = i
        End If
    Next i
Next k2

For j = 2 To 20
    ChartSortedStreamAreaV(j, 8) = Quantile(SortedStreamOptAreaVZero + (TotalLines - SortedStreamOptAreaVZero) / 20 * j)
    ChartSortedStreamAreaV(j, 7) = SortedStreamOptAreaV(SortedStreamOptAreaVZero + (TotalLines - SortedStreamOptAreaVZero) / 20 * j)
    ChartSortedStreamAreaV(j, 4) = Quantile(SortedStreamAcceptAreaVZero + (TotalLines - SortedStreamAcceptAreaVZero) / 20 * j)
    ChartSortedStreamAreaV(j, 3) = SortedStreamAcceptAreaV(SortedStreamAcceptAreaVZero + (TotalLines - SortedStreamAcceptAreaVZero) / 20 * j)

326
Let $j \geq 2$.

Next $j$

For $k_2 = 1$ To 10
For $j = 2$ To 20
  ChartSortedReachOptAreaV(k2, 2, j) = Quantile(SortedReachOptAreaVZero(k2) + (TotalLines - SortedReachOptAreaVZero(k2)) / 20 * j)
  ChartSortedReachOptAreaV(k2, 1, j) = SortedReachOptAreaV(k2, SortedReachOptAreaVZero(k2) + (TotalLines - SortedReachOptAreaVZero(k2)) / 20 * j)
  ChartSortedReachAcceptAreaV(k2, 2, j) = Quantile(SortedReachAcceptAreaVZero(k2) + (TotalLines - SortedReachAcceptAreaVZero(k2)) / 20 * j)
  ChartSortedReachAcceptAreaV(k2, 1, j) = SortedReachAcceptAreaV(k2, SortedReachAcceptAreaVZero(k2) + (TotalLines - SortedReachAcceptAreaVZero(k2)) / 20 * j)
  ChartSortedReachOptAreaD(k2, 2, j) = Quantile(SortedReachOptAreaDZero(k2) + (TotalLines - SortedReachOptAreaDZero(k2)) / 20 * j)
  ChartSortedReachOptAreaD(k2, 1, j) = SortedReachOptAreaD(k2, SortedReachOptAreaDZero(k2) + (TotalLines - SortedReachOptAreaDZero(k2)) / 20 * j)
  ChartSortedReachAcceptAreaD(k2, 2, j) = Quantile(SortedReachAcceptAreaDZero(k2) + (TotalLines - SortedReachAcceptAreaDZero(k2)) / 20 * j)
  ChartSortedReachAcceptAreaD(k2, 1, j) = SortedReachAcceptAreaD(k2, SortedReachAcceptAreaDZero(k2) + (TotalLines - SortedReachAcceptAreaDZero(k2)) / 20 * j)

327
ChartSortedReachAcceptAreaD(k2, 1, j) = SortedReachAcceptAreaD(k2, SortedReachAcceptAreaDZero(k2) + (TotalLines - SortedReachAcceptAreaDZero(k2)) / 20 * j)

ChartSortedReachOptAreaDV(k2, 2, j) = Quantile(SortedReachOptAreaDVZero(k2) + (TotalLines - SortedReachOptAreaDVZero(k2)) / 20 * j)

ChartSortedReachOptAreaDV(k2, 1, j) = SortedReachOptAreaDV(k2, SortedReachOptAreaDVZero(k2) + (TotalLines - SortedReachOptAreaDVZero(k2)) / 20 * j)

ChartSortedReachAcceptAreaDV(k2, 2, j) = Quantile(SortedReachAcceptAreaDVZero(k2) + (TotalLines - SortedReachAcceptAreaDVZero(k2)) / 20 * j)

ChartSortedReachAcceptAreaDV(k2, 1, j) = SortedReachAcceptAreaDV(k2, SortedReachAcceptAreaDVZero(k2) + (TotalLines - SortedReachAcceptAreaDVZero(k2)) / 20 * j)

Next j

Next k2

'Shrink arrays to save memory
ReDim AcceptTimeV(1, 1, 1, 1)
ReDim OptTimeV(1, 1, 1, 1)
ReDim AcceptTimeD(1, 1, 1, 1)
ReDim OptTimeD(1, 1, 1, 1)

ReDim SliceD(1), SliceA(1), SliceWP(1)
ReDim SliceR(1), SliceK(1), SliceV(1)
ReDim Ratio(1)
ReDim VelocityArray(1, 1, 1)
ReDim DepthArray(1, 1, 1)
ReDim AreaArray(1, 1, 1)

ReDim SumOptTimeV(1, 1, 1)
ReDim SumAcceptTimeV(1, 1, 1)
ReDim SumOptTimeD(1, 1, 1)
ReDim SumAcceptTimeD(1, 1, 1)
ReDim SumOptTimeDV(1, 1, 1)
ReDim SumAcceptTimeDV(1, 1, 1)

ReDim SumOptAreaV(1, 1, 1)
ReDim SumAcceptAreaV(1, 1, 1)
ReDim SumOptAreaD(1, 1, 1)
ReDim SumAcceptAreaD(1, 1, 1)
ReDim SumOptAreaDV(1, 1, 1)
ReDim SumAcceptAreaDV(1, 1, 1)
ReDim ReachSumOptTimeV(1, 1)
ReDim ReachSumAcceptTimeV(1, 1)
ReDim ReachSumOptTimeD(1, 1)
ReDim ReachSumAcceptTimeD(1, 1)
ReDim ReachSumOptTimeDV(1, 1)
ReDim ReachSumAcceptTimeDV(1, 1)
ReDim StreamOptAreaV(1, 1)
ReDim StreamAcceptAreaV(1, 1)
ReDim StreamOptAreaD(1, 1)
ReDim StreamAcceptAreaD(1, 1)
ReDim StreamOptAreaDV(1, 1)
ReDim StreamAcceptAreaDV(1, 1)
ReDim SortedReachOptAreaV(1, 1, 1)
ReDim SortedReachAcceptAreaV(1, 1, 1)
ReDim SortedReachOptAreaD(1, 1, 1)
ReDim SortedReachAcceptAreaD(1, 1, 1)
ReDim SortedReachOptAreaDV(1, 1, 1)
ReDim SortedReachAcceptAreaDV(1, 1, 1)
ReDim SortedStreamOptAreaV(1, 1)
ReDim SortedStreamAcceptAreaV(1, 1)
ReDim SortedStreamOptAreaD(1, 1)
ReDim SortedStreamAcceptAreaD(1, 1)
ReDim SortedStreamOptAreaDV(1, 1)
ReDim SortedStreamAcceptAreaDV(1, 1)
ReDim FtableD(1, 1), FtableQ(1, 1), FtableLines(1)
ReDim Quantile(1)
End Sub

Sub BaseHSI()
Dim i As Long
Dim j As Long
Dim j2() As Long
Dim FtableQ() As Single
Dim FtableD() As Single
Dim d As Single
Dim TW As Single
Dim IW As Single
Dim b() As Single
Dim z() As Single
Dim k As Integer
Dim n() As Single
Dim NumSlices As Integer
Dim SliceD() As Single
Dim SliceA() As Single
Dim SliceWP() As Single
Dim SliceR() As Single
Dim SliceK() As Single
Dim Ratio() As Single
Dim SliceV() As Single
Dim RectA As Single
Dim RectWP As Single
Dim RectR As Single
Dim RectK As Single
Dim RectV As Single
Dim TotalK As Single
Dim BaseVelocityArray() As Single
Dim BaseDepthArray() As Single
Dim BaseAreaArray() As Single
Dim k2 As Integer
Dim k3 As Integer
Dim k4 As Integer
Dim Fileline As String
Dim FtableLines() As Integer
Dim BaseAcceptTimeV() As Single
Dim BaseOptTimeV() As Single
Dim BaseAcceptTimeD() As Single
Dim BaseOptTimeD() As Single
Dim temp As Boolean

Dim BaseSumOptTimeV() As Single
Dim BaseSumAcceptTimeV() As Single
Dim BaseSumOptTimeD() As Single
Dim BaseSumAcceptTimeD() As Single
Dim BaseSumOptTimeDV() As Single
Dim BaseSumAcceptTimeDV() As Single

Dim BaseSumOptAreaV() As Single
Dim BaseSumAcceptAreaV() As Single
Dim BaseSumOptAreaD() As Single
Dim BaseSumAcceptAreaD() As Single
Dim BaseSumOptAreaDV() As Single
Dim BaseSumAcceptAreaDV() As Single

Dim BaseReachSumOptTimeV() As Single
Dim BaseReachSumAcceptTimeV() As Single
Dim BaseReachSumOptTimeD() As Single
Dim BaseReachSumAcceptTimeD() As Single
Dim BaseReachSumOptTimeDV() As Single
Dim BaseReachSumAcceptTimeDV() As Single

Dim BaseStreamOptAreaV() As Single
Dim BaseStreamAcceptAreaV() As Single
Dim BaseStreamOptAreaD() As Single
Dim BaseStreamAcceptAreaD() As Single
Dim BaseStreamOptAreaDV() As Single
Dim BaseStreamAcceptAreaDV() As Single

Dim BaseStreamSumOptTimeV As Single
Dim BaseStreamSumAcceptTimeV As Single
Dim BaseStreamSumOptTimeD As Single
Dim BaseStreamSumAcceptTimeD As Single
Dim BaseStreamSumOptTimeDV As Single
Dim BaseStreamSumAcceptTimeDV As Single
Dim BaseStreamTotalArea() As Single
Dim BaseTotalArea() As Single
Dim BaseSortedReachOptAreaV() As Single
Dim BaseSortedReachAcceptAreaV() As Single
Dim BaseSortedReachOptAreaD() As Single
Dim BaseSortedReachAcceptAreaD() As Single
Dim BaseSortedReachOptAreaDV() As Single
Dim BaseSortedReachAcceptAreaDV() As Single
Dim BaseSortedStreamOptAreaV() As Single
Dim BaseSortedStreamAcceptAreaV() As Single
Dim BaseSortedStreamOptAreaD() As Single
Dim BaseSortedStreamAcceptAreaD() As Single
Dim BaseSortedStreamOptAreaDV() As Single
Dim BaseSortedStreamAcceptAreaDV() As Single
Dim Quantile() As Single
Dim BaseSortedStreamOptAreaVZero As Long
Dim BaseSortedStreamAcceptAreaVZero As Long
Dim BaseSortedStreamOptAreaDZero As Long
Dim BaseSortedStreamAcceptAreaDZero As Long
Dim BaseSortedStreamOptAreaDVZero As Long
Dim BaseSortedStreamAcceptAreaDVZero As Long
Dim BaseSortedReachOptAreaVZero() As Long
Dim BaseSortedReachAcceptAreaVZero() As Long
Dim BaseSortedReachOptAreaDZero() As Long
Dim BaseSortedReachAcceptAreaDZero() As Long
Dim BaseSortedReachOptAreaDVZero() As Long
Dim BaseSortedReachAcceptAreaDVZero() As Long

NumSlices = 4

ReDim ChartSortedStreamAreaV(20, 8)
ReDim ChartSortedStreamAreaD(20, 8)
ReDim ChartSortedStreamAreaDV(20, 8)

ReDim BaseAcceptTimeV(10, BaseTotalLines, NumSlices + 1)
ReDim BaseOptTimeV(10, BaseTotalLines, NumSlices + 1)
ReDim BaseAcceptTimeD(10, BaseTotalLines, NumSlices + 1)
ReDim BaseOptTimeD(10, BaseTotalLines, NumSlices + 1)

ReDim SliceD(NumSlices), SliceA(NumSlices), SliceWP(NumSlices)
ReDim SliceR(NumSlices), SliceK(NumSlices), SliceV(NumSlices)
ReDim Ratio(NumSlices)
ReDim BaseVelocityArray(10, BaseTotalLines, NumSlices + 1)
ReDim BaseDepthArray(10, BaseTotalLines, NumSlices + 1)
ReDim BaseAreaArray(10, BaseTotalLines, NumSlices + 1)

ReDim BaseStreamOptAreaV(BaseTotalLines)
ReDim BaseStreamAcceptAreaV(BaseTotalLines)
ReDim BaseStreamOptAreaD(BaseTotalLines)
ReDim BaseStreamAcceptAreaD(BaseTotalLines)
ReDim BaseStreamOptAreaDV(BaseTotalLines)
ReDim BaseStreamAcceptAreaDV(BaseTotalLines)

ReDim BaseSortedReachOptAreaV(10, BaseTotalLines)
ReDim BaseSortedReachAcceptAreaV(10, BaseTotalLines)

ReDim BaseSortedReachOptAreaD(10, BaseTotalLines)
ReDim BaseSortedReachAcceptAreaD(10, BaseTotalLines)

ReDim BaseSortedReachOptAreaDV(10, BaseTotalLines)
ReDim BaseSortedReachAcceptAreaDV(10, BaseTotalLines)
ReDim BaseSortedStreamOptAreaV(BaseTotalLines)
ReDim BaseSortedStreamAcceptAreaV(BaseTotalLines)

ReDim BaseSortedStreamOptAreaD(BaseTotalLines)
ReDim BaseSortedStreamAcceptAreaD(BaseTotalLines)

ReDim BaseSortedStreamOptAreaDV(BaseTotalLines)
ReDim BaseSortedStreamAcceptAreaDV(BaseTotalLines)

ReDim FtableD(10, 20), FtableQ(10, 20), FtableLines(10)

ReDim Quantile(BaseTotalLines)

ReDim BaseStreamPercentOutput(6)
ReDim BaseReachPercentOutput(10, 6)

ReDim BaseReachPercentOptTimeV(10)
ReDim BaseReachPercentAcceptTimeV(10)
ReDim BaseReachPercentOptTimeD(10)
ReDim BaseReachPercentAcceptTimeD(10)
ReDim BaseReachPercentOptTimeDV(10)
ReDim BaseReachPercentAcceptTimeDV(10)

ReDim z(10), b(10), n(10)
Open "c:\HSPFPost\FTABLES.csv" For Input As #6

For J = 1 To 10
    Input #6, FtableLines(J)
    Input #6, z(J), b(J), n(J)
    For I = 1 To FtableLines(J)
        Input #6, FtableD(J, I), FtableQ(J, I)
    Next I
Next J
Close #6
For k2 = 1 To 10
    For i = 1 To BaseTotalLines 'Use all data
        For j = 2 To FtableLines(k2) 'Points in each Ftable
            If Baseq(k2, i) <= FtableQ(k2, j) And Baseq(k2, i) > FtableQ(k2, j - 1) Then 'Find where Q falls on Ftable
                d = FtableD(k2, j) - ((FtableQ(k2, j) - Baseq(k2, i)) * (FtableD(k2, j) - FtableD(k2, j - 1))) / (FtableQ(k2, j) - FtableQ(k2, j - 1))
                TW = b(k2) + 2 * d * z(k2) 'Top width
                IW = (TW - b(k2)) / (2 * NumSlices) 'Slice widths
                For k = 1 To NumSlices
                    SliceD(k) = d - ((IW / z(k2) * k) + (IW / z(k2) * (k - 1))) / 2
                    SliceA(k) = ((d - k * IW / z(k2)) * IW) + (IW / 2) * ((d - (IW / z(k2) * (k - 1))) - (d - (IW / z(k2) * k)))
                    SliceWP(k) = (IW ^ 2 + ((d - IW / z(k2) * (k - 1)) - (d - IW / z(k2) * k)) ^ 2) ^ 0.5
                    SliceR(k) = SliceA(k) / SliceWP(k)
                    SliceK(k) = 1.49 / n(k2) * SliceA(k) * SliceR(k) ^ (2 / 3)
                Next k
                RectA = b(k2) * d
                RectWP = b(k2)
                RectR = RectA / RectWP
                RectK = 1.49 / n(k2) * RectA * RectR ^ (2 / 3)
                TotalK = 0
                For k = 1 To NumSlices
                    TotalK = SliceK(k) * 2 + TotalK '2 channel sides
                Next k
                TotalK = TotalK + RectK
                For k = 1 To NumSlices
                    Ratio(k) = SliceK(k) / TotalK
                    SliceV(k) = (Ratio(k) * Baseq(k2, i)) / SliceA(k)
                Next k
                RectV = (RectK / TotalK * Baseq(k2, i)) / RectA
            'Store values in arrays
                For k = 1 To NumSlices
                    BaseVelocityArray(k2, i, k) = SliceV(k)
                    BaseDepthArray(k2, i, k) = SliceD(k)
                    BaseAreaArray(k2, i, k) = SliceA(k) * 2 '2 channel sides
                Next k
                BaseVelocityArray(k2, i, NumSlices + 1) = RectV
BaseDepthArray(k2, i, NumSlices + 1) = d
BaseAreaArray(k2, i, NumSlices + 1) = RectA

End If
Next j
Next i
Next k2

ReDim BaseTotalArea(10, BaseTotalLines), BaseStreamTotalArea(BaseTotalLines)
For k2 = 1 To 10
    For i = 1 To BaseTotalLines
        For k = 1 To NumSlices + 1
            BaseTotalArea(k2, i) = BaseAreaArray(k2, i, k) + BaseTotalArea(k2, i)
        Next k
    Next i
Next k2

For k2 = 1 To 10
    For i = 1 To BaseTotalLines
        BaseStreamTotalArea(i) = BaseTotalArea(k2, i) + BaseStreamTotalArea(i)
    Next i
Next k2

For i = 1 To BaseTotalLines
    For k2 = 1 To 10
        For j = 1 To NumSlices + 1
            If BaseVelocityArray(k2, i, j) > OptimumV(1) And BaseVelocityArray(k2, i, j) < OptimumV(2)
            Then
                BaseOptTimeV(k2, i, j) = 1  'If optimum velocity is available record a 1 at that time
            End If
            If BaseVelocityArray(k2, i, j) > AcceptableV(1) And BaseVelocityArray(k2, i, j) < AcceptableV(2)
            Then
                BaseAcceptTimeV(k2, i, j) = 1  'If acceptable velocity is available record a 1 at that time
            End If
            If BaseDepthArray(k2, i, j) > OptimumD(1) And BaseDepthArray(k2, i, j) < OptimumD(2)
            Then
                BaseOptTimeD(k2, i, j) = 1  'If optimum depth is available record a 1 at that time
            End If
            If BaseDepthArray(k2, i, j) > AcceptableD(1) And BaseDepthArray(k2, i, j) < AcceptableD(2)
            Then
                BaseAcceptTimeD(k2, i, j) = 1  'If acceptable depth is available record a 1 at that time
            End If
        Next j
    Next k2
Next i
BaseAcceptTimeD(k2, i, j) = 1 'If acceptable depth is available record a 1 at that
time
      End If
  Next j
 Next k2
Next i

ReDim BaseSumOptTimeV(10, BaseTotalLines)
ReDim BaseSumAcceptTimeV(10, BaseTotalLines)
ReDim BaseSumOptTimeD(10, BaseTotalLines)
ReDim BaseSumAcceptTimeD(10, BaseTotalLines)
ReDim BaseSumOptTimeDV(10, BaseTotalLines)
ReDim BaseSumAcceptTimeDV(10, BaseTotalLines)

ReDim BaseSumOptAreaV(10, BaseTotalLines)
ReDim BaseSumAcceptAreaV(10, BaseTotalLines)
ReDim BaseSumOptAreaD(10, BaseTotalLines)
ReDim BaseSumAcceptAreaD(10, BaseTotalLines)
ReDim BaseSumOptAreaDV(10, BaseTotalLines)
ReDim BaseSumAcceptAreaDV(10, BaseTotalLines)

' % Time loop
For i = 1 To BaseTotalLines
   For k2 = 1 To 10
      For j = 1 To NumSlices + 1
         If BaseOptTimeV(k2, i, j) = 1 Then
            'this will allow for values higher than 1
            BaseSumOptTimeV(k2, i) = BaseSumOptTimeV(k2, i) + 1
            BaseSumOptAreaV(k2, i) = BaseSumOptAreaV(k2, i) + BaseAreaArray(k2, i, j) / BaseTotalArea(k2, i)
         End If
      Next j
   Next k2
Next i
End If
If BaseOptTimeD(k2, i, j) = 1 Then
    BaseSumOptTimeD(k2, i) = BaseSumOptTimeD(k2, i) + 1
    BaseSumOptAreaD(k2, i) = BaseSumOptAreaD(k2, i) + BaseAreaArray(k2, i, j) / BaseTotalArea(k2, i)
End If

End If
If BaseAcceptTimeD(k2, i, j) = 1 Then
    BaseSumAcceptTimeD(k2, i) = BaseSumAcceptTimeD(k2, i) + 1
    BaseSumAcceptAreaD(k2, i) = BaseSumAcceptAreaD(k2, i) + BaseAreaArray(k2, i, j) / BaseTotalArea(k2, i)
End If

End If
If BaseOptTimeV(k2, i, j) = 1 And BaseOptTimeD(k2, i, j) = 1 Then
    BaseSumOptTimeDV(k2, i) = BaseSumOptTimeDV(k2, i) + 1
    BaseSumOptAreaDV(k2, i) = BaseSumOptAreaDV(k2, i) + BaseAreaArray(k2, i, j) / BaseTotalArea(k2, i)
End If

End If
If BaseAcceptTimeV(k2, i, j) = 1 And BaseOptTimeD(k2, i, j) = 1 Then
    BaseSumAcceptTimeDV(k2, i) = BaseSumAcceptTimeDV(k2, i) + 1
    BaseSumAcceptAreaDV(k2, i) = BaseSumAcceptAreaDV(k2, i) + BaseAreaArray(k2, i, j) / BaseTotalArea(k2, i)
End If

Next j
Next k2
Next i

ReDim BaseReachSumOptTimeV(10)
ReDim BaseReachSumAcceptTimeV(10)
ReDim BaseReachSumOptTimeD(10)
ReDim BaseReachSumAcceptTimeD(10)
ReDim BaseReachSumOptTimeDV(10)
ReDim BaseReachSumAcceptTimeDV(10)

For i = 1 To BaseTotalLines
    For k2 = 1 To 10
        If BaseSumOptTimeV(k2, i) > 0 Then
            BaseReachSumOptTimeV(k2) = BaseReachSumOptTimeV(k2) + 1
        End If
        If BaseSumAcceptTimeV(k2, i) > 0 Then
            BaseReachSumAcceptTimeV(k2) = BaseReachSumAcceptTimeV(k2) + 1
        End If
        If BaseSumOptTimeD(k2, i) > 0 Then
            BaseReachSumOptTimeD(k2) = BaseReachSumOptTimeD(k2) + 1
        End If
        If BaseSumAcceptTimeD(k2, i) > 0 Then
            BaseReachSumAcceptTimeD(k2) = BaseReachSumAcceptTimeD(k2) + 1
        End If
        If BaseSumOptTimeDV(k2, i) > 0 Then
            BaseReachSumOptTimeDV(k2) = BaseReachSumOptTimeDV(k2) + 1
        End If
        If BaseSumAcceptTimeDV(k2, i) > 0 Then
            BaseReachSumAcceptTimeDV(k2) = BaseReachSumAcceptTimeDV(k2) + 1
        End If
        BaseStreamOptAreaV(i) = BaseSumOptAreaV(k2, i) * BaseTotalArea(k2, i) / BaseStreamTotalArea(i) + BaseStreamOptAreaV(i)
        BaseStreamAcceptAreaV(i) = BaseSumAcceptAreaV(k2, i) * BaseTotalArea(k2, i) / BaseStreamTotalArea(i) + BaseStreamAcceptAreaV(i)
        BaseStreamOptAreaD(i) = BaseSumOptAreaD(k2, i) * BaseTotalArea(k2, i) / BaseStreamTotalArea(i) + BaseStreamOptAreaD(i)
        BaseStreamAcceptAreaD(i) = BaseSumAcceptAreaD(k2, i) * BaseTotalArea(k2, i) / BaseStreamTotalArea(i) + BaseStreamAcceptAreaD(i)
        BaseStreamOptAreaDV(i) = BaseSumOptAreaDV(k2, i) * BaseTotalArea(k2, i) / BaseStreamTotalArea(i) + BaseStreamOptAreaDV(i)
        BaseStreamAcceptAreaDV(i) = BaseSumAcceptAreaDV(k2, i) * BaseTotalArea(k2, i) / BaseStreamTotalArea(i) + BaseStreamAcceptAreaDV(i)
Next i
'Shrink arrays to save memory
ReDim BaseAcceptTimeV(1, 1, 1, 1)
ReDim BaseOptTimeV(1, 1, 1, 1)
ReDim BaseAcceptTimeD(1, 1, 1, 1)
ReDim BaseOptTimeD(1, 1, 1, 1)
ReDim BaseVelocityArray(1, 1, 1)
ReDim BaseDepthArray(1, 1, 1)
ReDim BaseAreaArray(1, 1, 1)
For k2 = 1 To 10
    BaseStreamSumOptTimeV = BaseReachSumOptTimeV(k2) + BaseStreamSumOptTimeV
    BaseStreamSumAcceptTimeV = BaseReachSumAcceptTimeV(k2) + BaseStreamSumAcceptTimeV
    BaseStreamSumOptTimeD = BaseReachSumOptTimeD(k2) + BaseStreamSumOptTimeD
    BaseStreamSumAcceptTimeD = BaseReachSumAcceptTimeD(k2) + BaseStreamSumAcceptTimeD
    BaseStreamSumOptTimeDV = BaseReachSumOptTimeDV(k2) + BaseStreamSumOptTimeDV
    BaseStreamSumAcceptTimeDV = BaseReachSumAcceptTimeDV(k2) + BaseStreamSumAcceptTimeDV
    BaseReachPercentOptTimeV(k2) = BaseReachSumOptTimeV(k2) / i
    BaseReachPercentAcceptTimeV(k2) = BaseReachSumAcceptTimeV(k2) / i
    BaseReachPercentOptTimeD(k2) = BaseReachSumOptTimeD(k2) / i
    BaseReachPercentAcceptTimeD(k2) = BaseReachSumAcceptTimeD(k2) / i
    BaseReachPercentOptTimeDV(k2) = BaseReachSumOptTimeDV(k2) / i
    BaseReachPercentAcceptTimeDV(k2) = BaseReachSumAcceptTimeDV(k2) / i
Next k2

BaseStreamPercentOptTimeV = BaseStreamSumOptTimeV / (i * 10)
BaseStreamPercentAcceptTimeV = BaseStreamSumAcceptTimeV / (i * 10)
BaseStreamPercentOptTimeD = BaseStreamSumOptTimeD / (i * 10)
BaseStreamPercentAcceptTimeD = BaseStreamSumAcceptTimeD / (i * 10)
BaseStreamPercentOptTimeDV = BaseStreamSumOptTimeDV / (i * 10)
BaseStreamPercentAcceptTimeDV = BaseStreamSumAcceptTimeDV / (i * 10)

'Sort Arrays
ReDim BaseSortedReachOptAreaVZero(10)
ReDim BaseSortedReachAcceptAreaVZero(10)
ReDim BaseSortedReachOptAreaDZero(10)
ReDim BaseSortedReachAcceptAreaDZero(10)
ReDim BaseSortedReachOptAreaDVZero(10)
ReDim BaseSortedReachAcceptAreaDVZero(10)
ReDim j2(6)

For k2 = 1 To 10
  For k = 1 To 6
    j2(k) = 0
  Next k
For k = 0 To 100
  For i = 1 To BaseTotalLines
    If Round(BaseSumAcceptAreaV(k2, i), 2) = k / 100 Then
      j2(1) = j2(1) + 1
      BaseSortedReachAcceptAreaV(k2, j2(1)) = Round(BaseSumAcceptAreaV(k2, i), 2)
    End If
    If Round(BaseSumOptAreaV(k2, i), 2) = k / 100 Then
      j2(2) = j2(2) + 1
      BaseSortedReachOptAreaV(k2, j2(2)) = Round(BaseSumOptAreaV(k2, i), 2)
    End If
    If Round(BaseSumAcceptAreaD(k2, i), 2) = k / 100 Then
      j2(3) = j2(3) + 1
      BaseSortedReachAcceptAreaD(k2, j2(3)) = Round(BaseSumAcceptAreaD(k2, i), 2)
    End If
    If Round(BaseSumOptAreaD(k2, i), 2) = k / 100 Then
      j2(4) = j2(4) + 1
      BaseSortedReachOptAreaD(k2, j2(4)) = Round(BaseSumOptAreaD(k2, i), 2)
    End If
    If Round(BaseSumAcceptAreaDV(k2, i), 2) = k / 100 Then
      j2(5) = j2(5) + 1
    End If
Next i
Next k2
BaseSortedReachAcceptAreaDV(k2, j2(5)) = Round(BaseSumAcceptAreaDV(k2, i), 2)
End If
If Round(BaseSumOptAreaDV(k2, i), 2) = k / 100 Then
    j2(6) = j2(6) + 1
    BaseSortedReachAcceptAreaDV(k2, j2(6)) = Round(BaseSumOptAreaDV(k2, i), 2)
End If
Next i
Next k
Next k2

For k = 1 To 6
    j2(k) = 0
Next k
For k = 0 To 100
    For i = 1 To BaseTotalLines
        If Round(BaseStreamAcceptAreaV(i), 2) = k / 100 Then
            j2(1) = j2(1) + 1
            BaseSortedStreamAcceptAreaV(j2(1)) = Round(BaseStreamAcceptAreaV(i), 2)
        End If
        If Round(BaseStreamOptAreaV(i), 2) = k / 100 Then
            j2(2) = j2(2) + 1
            BaseSortedStreamOptAreaV(j2(2)) = Round(BaseStreamOptAreaV(i), 2)
        End If
        If Round(BaseStreamAcceptAreaD(i), 2) = k / 100 Then
            j2(3) = j2(3) + 1
            BaseSortedStreamAcceptAreaD(j2(3)) = Round(BaseStreamAcceptAreaD(i), 2)
        End If
        If Round(BaseStreamOptAreaD(i), 2) = k / 100 Then
            j2(4) = j2(4) + 1
            BaseSortedStreamOptAreaD(j2(4)) = Round(BaseStreamOptAreaD(i), 2)
        End If
        If Round(BaseStreamAcceptAreaDV(i), 2) = k / 100 Then
            j2(5) = j2(5) + 1
            BaseSortedStreamAcceptAreaDV(j2(5)) = Round(BaseStreamAcceptAreaDV(i), 2)
        End If
        If Round(BaseStreamOptAreaDV(i), 2) = k / 100 Then
            j2(6) = j2(6) + 1
            BaseSortedStreamOptAreaDV(j2(6)) = Round(BaseStreamOptAreaDV(i), 2)
        End If
    Next i
Next k
Next k2
End If
Next i
Next k

'Set Up Quantiles
'Stream Arrays
For i = 1 To BaseTotalLines
    Quantile(i) = (i - 0.5) / BaseTotalLines
Next i
ReDim ChartSortedStreamAreaV(20, 8)
ReDim ChartSortedStreamAreaD(20, 8)
ReDim ChartSortedStreamAreaDV(20, 8)
ReDim ChartBaseSortedReachOptAreaV(10, 2, BaseTotalLines)
ReDim ChartBaseSortedReachAcceptAreaV(10, 2, BaseTotalLines)
ReDim ChartBaseSortedReachOptAreaD(10, 2, BaseTotalLines)
ReDim ChartBaseSortedReachAcceptAreaD(10, 2, BaseTotalLines)
ReDim ChartBaseSortedReachOptAreaDV(10, 2, BaseTotalLines)
ReDim ChartBaseSortedReachAcceptAreaDV(10, 2, BaseTotalLines)
For i = 1 To BaseTotalLines - 1
    If BaseSortedStreamOptAreaV(i) = 0 And BaseSortedStreamOptAreaV(i + 1) > 0 Then
        ChartSortedStreamAreaV(1, 6) = Quantile(i)
        ChartSortedStreamAreaV(1, 5) = BaseSortedStreamOptAreaV(i)
        BaseSortedStreamOptAreaVZero = i
    End If
Next i
Next i
For i = 1 To BaseTotalLines - 1
    If BaseSortedStreamAcceptAreaV(i) = 0 And BaseSortedStreamAcceptAreaV(i + 1) > 0 Then
        ChartSortedStreamAreaV(1, 2) = Quantile(i)
        ChartSortedStreamAreaV(1, 1) = BaseSortedStreamAcceptAreaV(i)
        BaseSortedStreamAcceptAreaVZero = i
    End If
Next i
Next i
For i = 1 To BaseTotalLines - 1
    If BaseSortedStreamOptAreaD(i) = 0 And BaseSortedStreamOptAreaD(i + 1) > 0 Then
        ChartSortedStreamAreaD(1, 6) = Quantile(i)
        ChartSortedStreamAreaD(1, 5) = BaseSortedStreamOptAreaD(i)
        BaseSortedStreamOptAreaDZero = i
    End If
Next i
For i = 1 To BaseTotalLines - 1
    If BaseSortedStreamAcceptAreaD(i) = 0 And BaseSortedStreamAcceptAreaD(i + 1) > 0 Then
        ChartSortedStreamAreaD(1, 2) = Quantile(i)
        ChartSortedStreamAreaD(1, 1) = BaseSortedStreamAcceptAreaD(i)
        BaseSortedStreamAcceptAreaDZero = i
    End If
Next i
For i = 1 To BaseTotalLines - 1
    If BaseSortedStreamOptAreaDV(i) = 0 And BaseSortedStreamOptAreaDV(i + 1) > 0 Then
        ChartSortedStreamAreaDV(1, 6) = Quantile(i)
        ChartSortedStreamAreaDV(1, 5) = BaseSortedStreamOptAreaDV(i)
        BaseSortedStreamOptAreaDVZero = i
    End If
Next i
For i = 1 To BaseTotalLines - 1
    If BaseSortedStreamAcceptAreaDV(i) = 0 And BaseSortedStreamAcceptAreaDV(i + 1) > 0 Then
        ChartSortedStreamAreaDV(1, 2) = Quantile(i)
        ChartSortedStreamAreaDV(1, 1) = BaseSortedStreamAcceptAreaDV(i)
End If
BaseSortedStreamAcceptAreaDVZero = i
End If

Next i
'Reach Arrays
For k2 = 1 To 10
   For i = 1 To BaseTotalLines - 1
      If BaseSortedReachOptAreaV(k2, i) = 0 And BaseSortedReachOptAreaV(k2, i + 1) > 0 Then
         ChartBaseSortedReachOptAreaV(k2, 2, 1) = Quantile(i)
         ChartBaseSortedReachOptAreaV(k2, 1, 1) = BaseSortedReachOptAreaV(k2, i)
         BaseSortedReachOptAreaVZero(k2) = i
      End If
   Next i
Next k2
For k2 = 1 To 10
   For i = 1 To BaseTotalLines - 1
      If BaseSortedReachAcceptAreaV(k2, i) = 0 And BaseSortedReachAcceptAreaV(k2, i + 1) > 0 Then
         ChartBaseSortedReachAcceptAreaV(k2, 2, 1) = Quantile(i)
         ChartBaseSortedReachAcceptAreaV(k2, 1, 1) = BaseSortedReachAcceptAreaV(k2, i)
         BaseSortedReachAcceptAreaVZero(k2) = i
      End If
   Next i
Next k2
For k2 = 1 To 10
   For i = 1 To BaseTotalLines - 1
      If BaseSortedReachOptAreaD(k2, i) = 0 And BaseSortedReachOptAreaD(k2, i + 1) > 0 Then
         ChartBaseSortedReachOptAreaD(k2, 2, 1) = Quantile(i)
      End If
   Next i
Next k2

ChartBaseSortedReachOptAreaD(k2, 1, 1) = BaseSortedReachOptAreaD(k2, i)

BaseSortedReachOptAreaDZero(k2) = i
End If

Next i
Next k2
For k2 = 1 To 10
    For i = 1 To BaseTotalLines - 1
        If BaseSortedReachAcceptAreaD(k2, i) = 0 And BaseSortedReachAcceptAreaD(k2, i + 1) > 0 Then
            ChartBaseSortedReachAcceptAreaD(k2, 2, 1) = Quantile(i)
            ChartBaseSortedReachAcceptAreaD(k2, 1, 1) = BaseSortedReachAcceptAreaD(k2, i)
            BaseSortedReachAcceptAreaDZero(k2) = i
        End If
    Next i
    Next k2
For k2 = 1 To 10
    For i = 1 To BaseTotalLines - 1
        If BaseSortedReachOptAreaDV(k2, i) = 0 And BaseSortedReachOptAreaDV(k2, i + 1) > 0 Then
            ChartBaseSortedReachOptAreaDV(k2, 2, 1) = Quantile(i)
            ChartBaseSortedReachOptAreaDV(k2, 1, 1) = BaseSortedReachOptAreaDV(k2, i)
            BaseSortedReachOptAreaDVZero(k2) = i
        End If
    Next i
    Next k2
For k2 = 1 To 10
    For i = 1 To BaseTotalLines - 1
        If BaseSortedReachAcceptAreaDV(k2, i) = 0 And BaseSortedReachAcceptAreaDV(k2, i + 1) > 0 Then
            ChartBaseSortedReachAcceptAreaDV(k2, 2, 1) = Quantile(i)
            ChartBaseSortedReachAcceptAreaDV(k2, 1, 1) = BaseSortedReachAcceptAreaDV(k2, i)
            BaseSortedReachAcceptAreaDVZero(k2) = i
        End If
    Next i
    Next k2

If BaseSortedReachAcceptAreaDV(k2, i) = 0 And BaseSortedReachAcceptAreaDV(k2, i + 1) > 0 Then
ChartBaseSortedReachAcceptAreaDV(k2, 2, 1) = Quantile(i)
ChartBaseSortedReachAcceptAreaDV(k2, 1, 1) = BaseSortedReachAcceptAreaDV(k2, i)
BaseSortedReachAcceptAreaDVZero(k2) = i
End If
Next i
Next k2

For j = 2 To 20
  ChartSortedStreamAreaV(j, 6) = Quantile(BaseSortedStreamOptAreaVZero + (BaseTotalLines -
  BaseSortedStreamOptAreaVZero) / 20 * j)
  ChartSortedStreamAreaV(j, 5) = BaseSortedStreamOptAreaV(BaseSortedStreamOptAreaVZero + (BaseTotalLines -
  BaseSortedStreamOptAreaVZero) / 20 * j)
  ChartSortedStreamAreaV(j, 2) = Quantile(BaseSortedStreamAcceptAreaVZero + (BaseTotalLines -
  BaseSortedStreamAcceptAreaVZero) / 20 * j)
  ChartSortedStreamAreaV(j, 1) = BaseSortedStreamAcceptAreaV(BaseSortedStreamAcceptAreaVZero +
  (BaseTotalLines - BaseSortedStreamAcceptAreaVZero) / 20 * j)
  ChartSortedStreamAreaD(j, 6) = Quantile(BaseSortedStreamOptAreaDZero + (BaseTotalLines -
  BaseSortedStreamOptAreaDZero) / 20 * j)
  ChartSortedStreamAreaD(j, 5) = BaseSortedStreamOptAreaD(BaseSortedStreamOptAreaDZero + (BaseTotalLines -
  BaseSortedStreamOptAreaDZero) / 20 * j)
  ChartSortedStreamAreaD(j, 2) = Quantile(BaseSortedStreamAcceptAreaDZero + (BaseTotalLines -
  BaseSortedStreamAcceptAreaDZero) / 20 * j)
  ChartSortedStreamAreaD(j, 1) = BaseSortedStreamAcceptAreaD(BaseSortedStreamAcceptAreaDZero +
  (BaseTotalLines - BaseSortedStreamAcceptAreaDZero) / 20 * j)
  ChartSortedStreamAreaDV(j, 6) = Quantile(BaseSortedStreamOptAreaDVZero + (BaseTotalLines -
  BaseSortedStreamOptAreaDVZero) / 20 * j)
  ChartSortedStreamAreaDV(j, 5) = BaseSortedStreamOptAreaDV(BaseSortedStreamOptAreaDVZero +
  (BaseTotalLines - BaseSortedStreamOptAreaDVZero) / 20 * j)
  ChartSortedStreamAreaDV(j, 2) = Quantile(BaseSortedStreamAcceptAreaDVZero + (BaseTotalLines -
  BaseSortedStreamAcceptAreaDVZero) / 20 * j)
  ChartSortedStreamAreaDV(j, 1) = BaseSortedStreamAcceptAreaDV(BaseSortedStreamAcceptAreaDVZero +
  (BaseTotalLines - BaseSortedStreamAcceptAreaDVZero) / 20 * j)
Next j
For k2 = 1 To 10
    For j = 2 To 20
        ChartBaseSortedReachOptAreaV(k2, 2, j) = Quantile(BaseSortedReachOptAreaVZero(k2) + (BaseTotalLines - BaseSortedReachOptAreaVZero(k2)) / 20 * j)
        ChartBaseSortedReachOptAreaV(k2, 1, j) = BaseSortedReachOptAreaV(k2, BaseSortedReachOptAreaVZero(k2) + (BaseTotalLines - BaseSortedReachOptAreaVZero(k2)) / 20 * j)
        ChartBaseSortedReachAcceptAreaV(k2, 2, j) = Quantile(BaseSortedReachAcceptAreaVZero(k2) + (BaseTotalLines - BaseSortedReachAcceptAreaVZero(k2)) / 20 * j)
        ChartBaseSortedReachAcceptAreaV(k2, 1, j) = BaseSortedReachAcceptAreaV(k2, BaseSortedReachAcceptAreaVZero(k2) + (BaseTotalLines - BaseSortedReachAcceptAreaVZero(k2)) / 20 * j)
    Next j
Next k2

For j = 2 To 20
    k2 = 5
    'Debug.Print ChartBaseSortedReachOptAreaV(k2, 1, j), ChartBaseSortedReachOptAreaV(k2, 2, j)
Next j

'Compile Output Arrays
'Shrink arrays to save memory

ReDim BaseSumOptTimeV(1, 1, 1)
ReDim BaseSumAcceptTimeV(1, 1, 1)
ReDim BaseSumOptTimeD(1, 1, 1)
ReDim BaseSumAcceptTimeD(1, 1, 1)
ReDim BaseSumOptTimeDV(1, 1, 1)
ReDim BaseSumAcceptTimeDV(1, 1, 1)

ReDim BaseSumOptAreaV(1, 1, 1)
ReDim BaseSumAcceptAreaV(1, 1, 1)
ReDim BaseSumOptAreaD(1, 1, 1)
ReDim BaseSumAcceptAreaD(1, 1, 1)
ReDim BaseSumOptAreaDV(1, 1, 1)
ReDim BaseSumAcceptAreaDV(1, 1, 1)

ReDim BaseReachSumOptTimeV(1, 1)
ReDim BaseReachSumAcceptTimeV(1, 1)
ReDim BaseReachSumOptTimeD(1, 1)
ReDim BaseReachSumAcceptTimeD(1, 1)
ReDim BaseReachSumOptTimeDV(1, 1)
ReDim BaseReachSumAcceptTimeDV(1, 1)

ReDim BaseStreamOptAreaV(1, 1)
ReDim BaseStreamAcceptAreaV(1, 1)
ReDim BaseStreamOptAreaD(1, 1)
ReDim BaseStreamAcceptAreaD(1, 1)
ReDim BaseStreamOptAreaDV(1, 1)
ReDim BaseStreamAcceptAreaDV(1, 1)

ReDim BaseSortedReachOptAreaV(1, 1, 1)
ReDim BaseSortedReachAcceptAreaV(1, 1, 1)

ReDim BaseSortedReachOptAreaD(1, 1, 1)
ReDim BaseSortedReachAcceptAreaD(1, 1, 1)

ReDim BaseSortedReachOptAreaDV(1, 1, 1)
ReDim BaseSortedReachAcceptAreaDV(1, 1, 1)
ReDim BaseSortedStreamOptAreaV(1, 1)
ReDim BaseSortedStreamAcceptAreaV(1, 1)
ReDim BaseSortedStreamOptAreaD(1, 1)
ReDim BaseSortedStreamAcceptAreaD(1, 1)
ReDim BaseSortedStreamOptAreaDV(1, 1)
ReDim BaseSortedStreamAcceptAreaDV(1, 1)
ReDim FtableD(1, 1), FtableQ(1, 1), FtableLines(1)
ReDim Quantile(1)
End Sub
Module (Module1)

Option Explicit
Option Base 1
Public Percent As Single
Public TimeLimit As Integer
Public Gm As Single

'Code supplied By Eduardo Mendez
Declare Function FindWindow Lib "User32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long
Declare Function GetWindowTextLength Lib "User32" Alias "GetWindowTextLengthA" (ByVal hWnd As Long) As Long
Declare Function GetWindowText Lib "User32" Alias "GetWindowTextA" (ByVal hWnd As Long, ByVal lpString As String, ByVal nMaxCount As Long) As Long 'cch
Declare Function GetClassName Lib "User32" Alias "GetClassNameA" (ByVal hWnd As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As Long
Declare Function DestroyWindow Lib "User32" (ByVal hWnd As Long) As Long
Declare Function OpenProcess Lib "kernel32" (ByVal dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal dwProcessId As Long) As Long
Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As Long, ByVal dwMilliseconds As Long) As Long
Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
Const SYNCHRONIZE = &H100000
Const INFINITE = &HFFFFFFFF
Const WAIT_TIMEOUT = &H102
Public ProcessID As Long
Public ProcessHandle As Long

Function SyncShell(ByVal pathname As String, ByVal windowstyle As Integer) As Boolean
Dim winhand As Long
Dim val As Long
Dim clsname As Long
Dim wtitlelen, nMaxCount, lpClassName, hWnd, lresult, length As Long
Dim wtitle, wintxt, fromsim, sInput, T As String
In VB4, an error occurs if Shell fails to start the program
On Error GoTo SyncShell_Error3
' Shell the program, get its handle, ' and wait for it to terminate
ProcessID = Shell(pathname, windowstyle)
If ProcessID <> 0 Then
    ProcessHandle = OpenProcess(SYNCHRONIZE, True, ProcessID)
    rerunit3:
        val = WaitForSingleObject(ProcessHandle, 1000) '4000) ' INFINITE)'eduardo
        Select Case val
            Case WAIT_TIMEOUT
                GoTo rerunit3
        End Select
    allset3:
        SyncShell = True
        Exit Function
Else
    MsgBox "Simulation executable was not able to run. ", vbOKOnly + vbCritical
End If
SyncShell_Error3:
    On Error GoTo 0
    SyncShell = False
    Exit Function
End Function
Options Form (frmOptions)

Private Sub Form_Load()
    txtPercent.Text = CStr(Percent)
    txtTime.Text = CStr(TimeLimit)
    txtGmap.Text = CStr(Gm)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Percent = val(txtPercent.Text)
    TimeLimit = val(txtTime.Text)
    Gm = val(txtGmap.Text)
End Sub