```
Sub OneDimWater()

Set Inputs = Worksheets("1d Wat").Range("b7")
Set Distance = Worksheets("1d Wat").Range("f7")
Set Characteristics = Worksheets("1d Wat").Range("s7")
Set PeakFlow = Worksheets("1d Wat").Range("x7")
Set MonthlyFlow = Worksheets("1d Wat").Range("ac7")
Set MonthlyHeights = Worksheets("1d Wat").Range("ag7")
Set Distance2 = Worksheets("1d Wat").Range("at7")
Set XYZ = Worksheets("1d Wat").Range("bf7")
Set Elevation = Worksheets("1d Sed").Range("AJ7")
Set GrainSize = Elevation.Offset(0, 8)
Set ChaCha = Worksheets("1d Sed").Range("n5")
Set Mannings = Worksheets("1d Wat").Range("g37")

tNext = 0

m = 1
Do Until m > 12
Q0 = MonthlyFlow.Cells(m, 2).Value
Peak = Inputs.Cells(10).Value

tLast = Inputs.Cells(3).Value
If Peak = True Then
m = 12
Q0 = Inputs.Cells(8).Value
Else
tLast = tLast / 4
End If

FLAG = False

'
'Steady state conditions
'

Y0 = Inputs.Cells(6).Value

i = 1
Do Until Distance.Cells(i) = ""
Characteristics.Cells(i, 1).Value = ChaCha.Cells(i, 1).Value
Characteristics.Cells(i, 2).Value = ChaCha.Cells(i, 2).Value
Characteristics.Cells(i, 3).Value = ChaCha.Cells(i, 3).Value

S(i) = Characteristics.Cells(i, 1).Value
B0(i) = Characteristics.Cells(i, 2).Value
S0(i) = Characteristics.Cells(i, 3).Value

i = i + 1
Loop

NP1 = i - 1

i = 2
```

```
Do While i <= NP1 - 1

t1 = Worksheets("1d Sed").Range("c18").Value
If t1 = 0 Then
CMN(i) = 0.029
Characteristics.Cells(i, 4).Value = CMN(i)
Mannings.Cells(i, t1 + 1).Value = CMN(i)
End If
i = i + 1
Loop

ar = (B0(1) + Y0 * S(1)) * Y0
HR = ar / (B0(1) + 2 * Y0 * (1 + S(1) ^ 2) ^ 0.5)
Top(1) = B0(1) + 2 * Y0 * S(1)
CENTR = Y0 ^ 2 * (B0(1) / 2 + Y0 * S(1) / 3)

c = (g * ar / Top(1)) ^ 0.5
V0 = Q0 / ar
dx = Distance.Cells(2) - Distance.Cells(1)
dt = dx / (Abs(V0) + c)

Inputs.Cells(2).Value = dt

ip = 1
n = 0

iprint = Inputs.Cells(12).Value
Ntoprint = Inputs.Cells(13).Value

ok = False

If t = tNext Then

i = 1
Do While i <= NP1

Distance.Cells(i, 2) = Q0

Do
CMN(i) = Characteristics.Cells(i, 4).Value
YNEW = (Q0 / (Abs(S0(i)) ^ 0.5 / CMN(i) * (B0(i) + Y0 * S(i)) * Y0 / (B0(i) + 2 * Y0 * (1 + S(i) ^ 2)
^ 0.5)) - Y0 ^ 2 * S(i)) / B0(i)
Y0 = Y0 + 0.1 * (YNEW - Y0)
Loop Until (YNEW - Y0) < 0.01 And (YNEW - Y0) > -0.01

Distance.Cells(i, 3) = YNEW
ARNEW = (B0(i) + YNEW * S(i)) * YNEW
Distance.Cells(i, 4) = Q0 / ARNEW

i = i + 1
Loop

End If
```

```
'
'Compute transient conditions
'

t = Inputs.Cells(2).Value

Do While t <= tLast

'
'Upstream end
'

Distance.Cells(1, 2) = Q0
Distance.Cells(2, 2) = Q0

Do

YNEW = (Q0 / (S0(1) ^ 0.5 / CMN(1) * (B0(1) + Y0 * S(1)) * Y0 / (B0(1) + 2 * Y0 * (1 + S(1) ^ 2) ^
0.5)) - Y0 ^ 2 * S(1)) / B0(1)
Y0 = Y0 + 0.1 * (YNEW - Y0)
Loop Until (YNEW - Y0) < 0.01 And (YNEW - Y0) > -0.01

Distance.Cells(1, 3).Value = Y0
Do
YNEW = (Q0 / (S0(2) ^ 0.5 / CMN(2) * (B0(i) + Y0 * S(2)) * Y0 / (B0(2) + 2 * Y0 * (1 + S(2) ^ 2) ^
0.5)) - Y0 ^ 2 * S(2)) / B0(2)
Y0 = Y0 + 0.1 * (YNEW - Y0)
Loop Until (YNEW - Y0) < 0.01 And (YNEW - Y0) > -0.01

Distance.Cells(2, 3).Value = Y0

Distance.Cells(1, 4).Value = Distance.Cells(2, 4).Value
Distance.Cells(2, 4).Value = Distance.Cells(3, 4).Value

Distance.Cells(2, 5).Value = Distance.Cells(3, 5).Value
Distance.Cells(2, 6).Value = Q0
Distance.Cells(3, 6).Value = Q0
Distance.Cells(2, 7).Value = Distance.Cells(3, 7).Value
Distance.Cells(2, 8).Value = Distance.Cells(3, 8).Value
Distance.Cells(2, 9).Value = Q0
Distance.Cells(2, 10).Value = Distance.Cells(3, 10).Value

'
'Downstream end
'

Distance.Cells(NP1, 2).Value = Distance.Cells(NP1 - 1, 2).Value
Distance.Cells(NP1 - 1, 2).Value = Distance.Cells(NP1 - 2, 2).Value
Distance.Cells(NP1, 3).Value = Distance.Cells(NP1 - 1, 3).Value
Distance.Cells(NP1 - 1, 3).Value = Distance.Cells(NP1 - 2, 3).Value
Distance.Cells(NP1, 4).Value = Distance.Cells(NP1 - 1, 4).Value
Distance.Cells(NP1 - 1, 4).Value = Distance.Cells(NP1 - 2, 4).Value
Distance.Cells(NP1 - 1, 5).Value = Distance.Cells(NP1 - 2, 5).Value
Distance.Cells(NP1, 6) = Distance.Cells(NP1 - 1, 6)
```

```
Distance.Cells(NP1 - 1, 6) = Distance.Cells(NP1 - 2, 6)

'FACTORP1
Distance.Cells(NP1, 7) = Distance.Cells(NP1 - 1, 7)
Distance.Cells(NP1 - 1, 7) = Distance.Cells(NP1 - 2, 7)
Distance.Cells(NP1 - 1, 8).Value = Distance.Cells(NP1 - 2, 8).Value
Distance.Cells(NP1 - 1, 9).Value = Distance.Cells(NP1 - 2, 9).Value
Distance.Cells(NP1 - 1, 10).Value = Distance.Cells(NP1 - 2, 10).Value

'
'INTERIOR NODES
'
'PREDICTOR STEP
'

FLAG = True
Do While FLAG = True

DTDX = dt / dx

i = 2
Do While i <= NP1 - 1

QI1 = Distance.Cells(i - 1, 2)
YI1 = Distance.Cells(i - 1, 3)
ARI1 = (B0(i) + YI1 * S(i)) * YI1
CENTRI1 = YI1 ^ 2 * (B0(i) / 2 + YI1 * S(i) / 3)
HRI1 = ARI1 / (B0(i) + 2 * YI1 * (1 + S(i) ^ 2) ^ 0.5)

QI2 = Distance.Cells(i, 2)
YI2 = Distance.Cells(i, 3)
ARI2 = (B0(i) + YI2 * S(i)) * YI2
CENTRI2 = YI2 ^ 2 * (B0(i) / 2 + YI2 * S(i) / 3)
HRI2 = ARI2 / (B0(i) + 2 * YI2 * (1 + S(i) ^ 2) ^ 0.5)

QVI = QI2 ^ 2 / ARI2 + g * CENTRI2
QVIM1 = QI1 ^ 2 / ARI1 + g * CENTRI1
SFI = CMN(i) ^ 2 * QI2 * Abs(QI2) / (ARI2 ^ 2 * HRI2 ^ (4 / 3))

S0SFI = g * ARI2 * (S0(i) - SFI)

QP1 = QI2 - DTDX * (QVI - QVIM1) + S0SFI * dt
Distance.Cells(i, 6) = QP1

AP1 = ARI2 - DTDX * (QI2 - QI1)
Distance.Cells(i, 5) = AP1

If S(i) <= 0 Then
yp1 = AP1 / B0(i)
Else
YBACK = (-B0(i) + (B0(i) ^ 2 + 4 * AP1 * S(i)) ^ 0.5) / (2 * S(i))
yp1 = YBACK
End If
```

```
HRP1 = (B0(i) + yp1 * S(i)) * yp1 / (B0(i) + 2 * yp1 * (1 + S(i) ^ 2) ^ 0.5)
SFP1 = CMN(i) ^ 2 * QP1 * Abs(QP1) / (AP1 ^ 2 * HRP1 ^ (4 / 3))
S0SFP1 = g * AP1 * (S0(i) - SFP1)
Distance.Cells(i, 8) = S0SFP1

CENTRP1 = yp1 ^ 2 * (B0(i) / 2 + yp1 * S(i) / 3)
FACTORI = QP1 ^ 2 / AP1 + g * CENTRP1
Distance.Cells(i, 7) = FACTORI

i = i + 1
Loop

'
'CORRECTOR STEP
'

i = 2
DTMIN = 1000
FLAG = False

Do While i <= NP1 - 1
FACTORI2 = Distance.Cells(i, 7)
FACTORI3 = Distance.Cells(i + 1, 7)
S0SFP1 = Distance.Cells(i, 8)

QI2 = Distance.Cells(i, 2)
QP2 = QI2 - DTDX * (FACTORI3 - FACTORI2) + S0SFP1 * dt

YI2 = Distance.Cells(i, 3)
ARI2 = (B0(i) + YI2 * S(i)) * YI2

QP1I2 = Distance.Cells(i, 6)
QP1I3 = Distance.Cells(i + 1, 6)
AP2 = ARI2 - DTDX * (QP1I3 - QP1I2)

QNEW = 0.5 * (QP1I2 + QP2)

Distance.Cells(i, 9).Value = QNEW

AP1I2 = Distance.Cells(i, 5)
ARNEW = 0.5 * (AP1I2 + AP2)

If S(i) <= 0 Then
YNEW = ARNEW / B0(i)
Else
YBACK = (-B0(i) + (B0(i) ^ 2 + 4 * ARNEW * S(i)) ^ 0.5) / (2 * S(i))
YNEW = YBACK
End If

Distance.Cells(i, 10).Value = YNEW

VNEW = QNEW / ARNEW

If ip = iprint And Peak = True And t1 > 0 And Inputs.Cells(15).Value = True Then
```

```
AggAggRoughness
Characteristics.Cells(i, 4).Value = CMN(i)
Mannings.Cells(i, t1 + 1).Value = CMN(i)
End If


Distance.Cells(i, 4).Value = VNEW
Distance2.Cells(i, 4).Value = VNEW

Distance2.Cells(i, 2).Value = Distance.Cells(i, 9).Value
Distance2.Cells(i, 3).Value = Distance.Cells(i, 10).Value

'
'Check for stability
'

Top(i) = B0(i) + 2 * YNEW * S(i)

c = (g * ARNEW / Top(i)) ^ 0.5
If dt > dx / (Abs(VNEW) + c) Then
FLAG = True
End If

DTCHECK = 0.9 * dx / (Abs(VNEW) + c)

If DTCHECK < DTMIN Then
DTMIN = DTCHECK
End If

i = i + 1
Loop


dt = DTMIN

Inputs.Cells(1).Value = dt

Loop

i = 2
Do While i <= NP1 - 1
Distance.Cells(i, 2).Value = Distance.Cells(i, 9).Value
Distance.Cells(i, 3).Value = Distance.Cells(i, 10).Value

i = i + 1
Loop

If ip = iprint And Peak = True Then
If t1 = 0 Then
Worksheets("1d Wat").Range("bf5:iv1000").ClearContents
End If

ok = False
Do
```

```
If XYZ.Cells(1, 3 * n + 1).Offset(-1).Value <> "" Then
n = n + 1
Else
ok = True
End If
Loop Until ok

End If

t = t + dt
Inputs.Cells(2).Value = t

If ip = iprint Then
ip = 0
End If
ip = ip + 1

Application.Wait (True)

Loop

t = tLast

If Peak = True Then

Top(1) = Top(2)
Top(NP1) = Top(NP1 - 1)
For i = 1 To NP1
PeakFlow.Cells(i, 1).Value = Distance.Cells(i, 2).Value
PeakFlow.Cells(i, 2).Value = Distance.Cells(i, 3).Value
PeakFlow.Cells(i, 3).Value = Distance.Cells(i, 4).Value
PeakFlow.Cells(i, 4).Value = Top(i)
Next i
Else
For i = 1 To NP1
MonthlyHeights.Cells(i, m).Value = Distance.Cells(i, 3).Value
Next i

End If

tNext = tLast

m = m + 1
Loop

End Sub

Sub AggAggRoughness()

Set Topography = Worksheets("2d Wat").Range("i3:ab22")
mmax = Topography.Rows.Count
nmax = Topography.Columns.Count
Set Roughness = Topography.Offset(0, nmax + 4)
Set Heights = Topography.Offset(mmax + 4, 0)
```

```
'Model cross over
Worksheets("2d Wat").Range("e3").Value = i
Worksheets("2d Sed").Range("c8").Value = i
Template
sumh = 0
sumks = 0
no = 0
For n = 1 To nmax
For m = 1 To mmax
Z(m, n) = Topography.Cells(m, n).Value
ks(m, n) = Roughness.Cells(m, n).Value
H(m, n) = Heights.Cells(m, n).Value

If H(m, n) > Z(m, n) Then
ks(m, n) = Roughness.Cells(m, n).Value
sumh = sumh + (H(m, n) - Z(m, n))
sumks = sumks + ks(m, n)
no = no + 1
End If
Next m
Next n

tg = rho * VNEW ^ 2 / (5.75 * WorksheetFunction.Log10(12 * sumh / sumks)) ^ 2

HR = ARNEW / (B0(i) + 2 * YNEW * (1 + S(i) ^ 2) ^ 0.5)

CMN(i) = (tg * ARNEW ^ 2 * HR ^ (4 / 3) / (rho * g * YNEW * QNEW ^ 2)) ^ 0.5

End Sub
```

```
Sub OneDimSediment()

Set Inputs = Worksheets("1d Sed").Range("c6")

j = Inputs.Cells(13).Value
set_strain_curves
Get_Data
Get_Auxiliary_Data
Setup_Initial_Bed_and_Time
Send_Output
If j = 0 Then
Worksheets("1d Sed").Range("be3:iv1000").ClearContents
Worksheets("2d Wat").Range("ca1:iv20000").ClearContents
Worksheets("2d Sed").Range("bf1:iv20000").ClearContents
Worksheets("2d Veg").Range("bi1:iv20000").ClearContents
End If

Do

'Model cross over
Worksheets("1d Wat").Range("b16") = True
OneDimWater
Worksheets("1d Wat").Range("b16") = False

W = 0
DoneInnerLoop = False
Do

Find_Slopes_and_Shields_Stresses
Find_Load
Find_New_eta

W = W + 1
If W = Mtoprint Then DoneInnerLoop = True

Loop Until DoneInnerLoop


j = j + 1
Inputs.Cells(13).Value = j
Send_Output


If Worksheets("1d Wat").Range("b21") = True Then

'Model cross over
Worksheets("1d Wat").Range("b16") = False
OneDimWater

'Model cross over
OneDimReeds
End If

If j = Mprint Then
```

```
Finished = True
Output_Distribution
End If

Loop Until Finished

End Sub

Sub set_strain_curves()
po(1) = 0.6684:   oo(1) = 1.011:    so(1) = 0.8157
po(2) = 0.7639:   oo(2) = 1.011:    so(2) = 0.8157
po(3) = 0.8601:   oo(3) = 1.01:     so(3) = 0.8182
po(4) = 0.9096:   oo(4) = 1.008:    so(4) = 0.8233
po(5) = 0.9615:   oo(5) = 1.004:    so(5) = 0.8333
po(6) = 1#:       oo(6) = 0.9997:   so(6) = 0.8439
po(7) = 1.055:    oo(7) = 0.9903:   so(7) = 0.8621
po(8) = 1.108:    oo(8) = 0.9789:   so(8) = 0.8825
po(9) = 1.197:    oo(9) = 0.9567:   so(9) = 0.9214
po(10) = 1.302:   oo(10) = 0.9273:  so(10) = 0.9723
po(11) = 1.407:   oo(11) = 0.8964:  so(11) = 1.025
po(12) = 1.529:   oo(12) = 0.8604:  so(12) = 1.083
po(13) = 1.641:   oo(13) = 0.8287:  so(13) = 1.13
po(14) = 1.702:   oo(14) = 0.8123:  so(14) = 1.153
po(15) = 1.832:   oo(15) = 0.7796:  so(15) = 1.196
po(16) = 1.937:   oo(16) = 0.7554:  so(16) = 1.225
po(17) = 2.044:   oo(17) = 0.7326:  so(17) = 1.25
po(18) = 2.261:   oo(18) = 0.6928:  so(18) = 1.287
po(19) = 2.499:   oo(19) = 0.6585:  so(19) = 1.313
po(20) = 2.732:   oo(20) = 0.6345:  so(20) = 1.333
po(21) = 2.993:   oo(21) = 0.615:   so(21) = 1.352
po(22) = 3.477:   oo(22) = 0.5877:  so(22) = 1.38
po(23) = 4.075:   oo(23) = 0.564:   so(23) = 1.403
po(24) = 4.469:   oo(24) = 0.5523:  so(24) = 1.414
po(25) = 5.016:   oo(25) = 0.5395:  so(25) = 1.426
po(26) = 6.158:   oo(26) = 0.5209:  so(26) = 1.444
po(27) = 7.821:   oo(27) = 0.5045:  so(27) = 1.458
po(28) = 10.06:   oo(28) = 0.4917:  so(28) = 1.469
po(29) = 14.38:   oo(29) = 0.479:   so(29) = 1.48
po(30) = 19.97:   oo(30) = 0.4712:  so(30) = 1.486
po(31) = 25.79:   oo(31) = 0.4668:  so(31) = 1.49
po(32) = 38.57:   oo(32) = 0.462:   so(32) = 1.493
po(33) = 68.74:   oo(33) = 0.4578:  so(33) = 1.497
po(34) = 91.95:   oo(34) = 0.4564:  so(34) = 1.498
po(35) = 231.2:   oo(35) = 0.4541:  so(35) = 1.499
po(36) = 2320#:   oo(36) = 0.4527:  so(36) = 1.5
End Sub

Sub Get_Data()

Set Inputs = Worksheets("1d Sed").Range("c6")
Set InputDistribution = Worksheets("1d Sed").Range("b23")
Set Peak = Worksheets("1d Wat").Range("y7")
Set ChaCha = Worksheets("1d Sed").Range("n5")
```

```
qbTf = Inputs.Cells(2).Value
Inter = Inputs.Cells(3).Value
etadI = Inputs.Cells(4).Value
Sfbi = Inputs.Cells(5).Value
L = Inputs.Cells(6).Value
dt = Inputs.Cells(7).Value

m = Inputs.Cells(8).Value
Mtoprint = Inputs.Cells(9).Value
Mprint = Inputs.Cells(10).Value


For jj = 1 To npp
di(jj) = InputDistribution.Cells(jj, 1).Value
plff(jj) = InputDistribution.Cells(jj, 2).Value
FfI(jj) = InputDistribution.Cells(jj, 3).Value
Ffs(jj) = InputDistribution.Cells(jj, 4).Value

Next jj
dx = L / m
dt = timeday * dt
For jj = 1 To np
ds(jj) = (di(jj) * di(jj + 1)) ^ 0.5

psi(jj) = Log(ds(jj)) / Log(2)

plf(jj) = (plff(jj) - plff(jj + 1)) / 100
FI(jj) = (FfI(jj) - FfI(jj + 1)) / 100
fs(jj) = (Ffs(jj) - Ffs(jj + 1)) / 100

If di(jj) > 2 And di(jj + 1) <= 2 Then
fracsandI = (FfI(jj + 1) + (FfI(jj) - FfI(jj + 1)) / (Log(di(jj)) - Log(di(jj + 1))) * (Log(2) - Log(di(jj +
1)))) / 100
End If

Next jj

rload = tausr50 / tausc50

If rload > 1 Then
Cexp = aload * (1 - 1 / rload) ^ nload
nexp = nload / rload / (1 - 1 / rload)
End If

For i = 1 To m + 1

H(i) = Peak.Cells(i).Value
Vel(i) = Peak.Cells(i, 2).Value
sdsl(i) = ChaCha.Cells(i, 1).Value
B0(i) = ChaCha.Cells(i, 2).Value

Next i

End Sub
```

```
Sub Get_Auxiliary_Data()

Set Parameters = Worksheets("1d Sed").Range("v5")

nactive = Parameters.Cells(2).Value
Rr = Parameters.Cells(4).Value
lps = Parameters.Cells(5).Value
au = Parameters.Cells(6).Value
atrans = Parameters.Cells(7).Value
End Sub

Sub Setup_Initial_Bed_and_Time()

Set ChaCha = Worksheets("1d Sed").Range("n5")
Set Elevation = Worksheets("1d Sed").Range("AJ7")

time = j * dt * Mtoprint

dsgsi = 0
For jj = 1 To np
dsgsi = dsgsi + FI(jj) * psi(jj)
Next jj
dsgsi = Exp(Log(2) * dsgsi)

FfI(1) = 100
For jj = 2 To npp
FfI(jj) = FfI(jj - 1) - FI(jj - 1) * 100
Next jj

Z = False
jj = 0
Do
jj = jj + 1
If (FfI(jj) >= 90) And (FfI(jj + 1) < 90) Then Z = True
Loop Until Z
ds90si = Exp(Log(di(jj + 1)) + (Log(di(jj)) - Log(di(jj + 1))) / _
(FfI(jj) - FfI(jj + 1)) * (90 - FfI(jj + 1)))


Z = False
jj = 0
Do
jj = jj + 1
If (FfI(jj) >= 50) And (FfI(jj + 1) < 50) Then Z = True
Loop Until Z
ds50si = Exp(Log(di(jj + 1)) + (Log(di(jj)) - Log(di(jj + 1))) / _
(FfI(jj) - FfI(jj + 1)) * (50 - FfI(jj + 1)))

For i = 1 To m + 1

xx(i) = dx * (i - 1)

If j = 0 Then
```

```
eta(i) = etadI + L * Sfbi - xx(i) * Sfbi
Else
eta(i) = Elevation.Cells(i, 4).Value
End If

ChaCha.Cells(i, 3).Value = Sfbi
If Worksheets("1d Sed").Range("c19").Value = True Then
etarock(i) = L * Sfbi - xx(i) * Sfbi
ChaCha.Cells(i, 4).Value = etarock(i)
Else
ChaCha.Cells(i, 4).Value = 0
End If

For jj = 1 To np
f(i, jj) = FI(jj)
Next jj

dsgs(i) = dsgsi
ds90s(i) = ds90si
ds50s(i) = ds50si
fracsand(i) = fracsandI

Next i

Finished = False

End Sub

Sub Find_Slopes_and_Shields_Stresses()
Set ChaCha = Worksheets("1d Sed").Range("n5")

S(1) = (eta(1) - eta(2)) / dx
Sf(1) = ((H(1) + Vel(1) ^ 2 / (2 * g)) - (H(2) + Vel(2) ^ 2 / (2 * g))) / dx + S(1)

S(m + 1) = (eta(m) - eta(m + 1)) / dx
Sf(m + 1) = ((H(m) + Vel(m) ^ 2 / (2 * g)) - (H(m + 1) + Vel(m + 1) ^ 2 / (2 * g))) / dx + S(m + 1)

For i = 2 To m
S(i) = (eta(i - 1) - eta(i + 1)) / (2 * dx)
Sf(i) = ((H(i - 1) + Vel(i - 1) ^ 2 / (2 * g)) - (H(i + 1) + Vel(i + 1) ^ 2 / (2 * g))) / (2 * dx) + S(i)
Next i

For i = 1 To m + 1

ChaCha.Cells(i, 3).Value = S(i)
ChaCha.Cells(i, 1).Value = sdsl(i)

R = (B0(i) + H(i) * sdsl(i)) * H(i) / (B0(i) + 2 * H(i) * (1 + (sdsl(i)) ^ 2) ^ 0.5)
If Sf(i) <= 0 Then
tausg(i) = 0
Else
tausg(i) = R * Sf(i) / (Rr * ds50s(i) / 1000)
End If
Next i
```

```
End Sub

Sub Find_Load()

Set EquationSelector = Worksheets("1d Sed").Range("i9")

If EquationSelector = 1 Then 'Parker bedload relation
For i = 1 To m + 1
If etarock(i) <= eta(i) Then

dsgt = 0
For jj = 1 To np

Ft(jj) = f(i, jj)

dsgt = dsgt + Ft(jj) * psi(jj)
Next jj
dsgt = Exp(Log(2) * dsgt)


sas = 0
xfm = Log(dsgt) / Log(2)
For jj = 1 To np
sas = sas + ((psi(jj) - xfm) ^ 2) * Ft(jj)

Next jj

sas = 2 ^ (sas ^ 0.5)
sas = Log(sas) / Log(2)

phisgo = tausg(i) / taursgo

If phisgo < lowphi Then
   qbT(i) = 0
   For jj = 1 To np
      pl(i, jj) = 1 / np
   Next jj
Else
   Gsum = 0
   find_omega
   For jj = 1 To np

   Zs = ds(jj) / dsgt

      arg = omega * phisgo * Exp(-beta * Log(Zs))

If arg < 1 Then
gg = Exp(Mo * Log(arg))
Else
If arg <= 1.59 Then
gg = Exp(14.2 * (arg - 1) - 9.28 * (arg - 1) ^ 2)
Else
gg = 5474 * Exp(4.5 * Log(1 - 0.853 / arg))
```

```
End If
End If
      plt(jj) = Ft(jj) * gg
      Gsum = Gsum + plt(jj)
   Next jj
   For jj = 1 To np
      pl(i, jj) = plt(jj) / Gsum
   Next jj
   ustar = (Rr * g * (dsgs(i) / 1000) * tausg(i)) ^ 0.5
   qbT(i) = Gsum * 0.00218 * ustar ^ 3 / Rr / g
End If
Else
qbT(i) = 0
End If

Next i

Else

For i = 1 To m + 1

If etarock(i) <= eta(i) Then

taussrg = 0.021 + 0.015 * Exp(-20 * fracsand(i))
phisgo = tausg(i) / taussrg
If phisgo < 0.01 Then
   qbT(i) = 0
   For jj = 1 To np
      pl(i, jj) = 1 / np
   Next jj
Else
   Gsum = 0
   For jj = 1 To np
      b = 0.67 / (1 + Exp(1.5 - ds(jj) / dsgs(i)))
      arg = phisgo * (ds(jj) / dsgs(i)) ^ (-b)

If arg < 1.35 Then
GGwc = 0.002 * arg ^ 7.5
Else
GGwc = 14 * (1 - 0.894 / (arg ^ 0.5)) ^ 4.5
End If
      plt(jj) = f(i, jj) * GGwc
      Gsum = Gsum + plt(jj)
   Next jj
      For jj = 1 To np
      pl(i, jj) = plt(jj) / Gsum
   Next jj
   ustar = (Rr * g * (dsgs(i) / 1000) * tausg(i)) ^ 0.5
   qbT(i) = Gsum * ustar ^ 3 / Rr / g
End If

Else
qbT(i) = 0
End If
```

```
Next i

End If

End Sub

Sub find_omega()
If phisgo > po(35) Then
omegaoak = oo(35) + (oo(36) - oo(35)) / (po(36) - po(35)) _
* (phisgo - po(35))
sigoak = so(35) + (so(36) - so(35)) / (po(36) - po(35)) _
    * (phisgo - po(35))
Else
If phisgo <= 0.7639 Then
omegaoak = 1.011
sigoak = 0.8157
Else
foundit = False
kk = 1
Do
   kk = kk + 1
   If (phisgo > po(kk)) And (phisgo <= po(kk + 1)) Then foundit = True
Loop Until foundit
xm1 = po(kk - 1): x = po(kk): xp1 = po(kk + 1): xp2 = po(kk + 2)
ym1 = oo(kk - 1): y = oo(kk): yp1 = oo(kk + 1): yp2 = oo(kk + 2)
cubic_interpolator omegaoak
ym1 = so(kk - 1): y = so(kk): yp1 = so(kk + 1): yp2 = so(kk + 2)
cubic_interpolator sigoak
End If
End If
omega = 1 + sas / sigoak * (omegaoak - 1)
End Sub

Sub cubic_interpolator(yout)
a1 = ym1
a2 = (y - a1) / (x - xm1)
a3 = (yp1 - a1 - a2 * (xp1 - xm1)) / _
(xp1 - xm1) / (xp1 - x)
a4 = yp2 - a1 - a2 * (xp2 - xm1) _
- a3 * (xp2 - xm1) * (xp2 - x)
a4 = a4 / (xp2 - xm1) / (xp2 - x) / (xp2 - xp1)
yout = a1 + a2 * (phisgo - xm1) + a3 * (phisgo - xm1) * (phisgo - x) _
+ a4 * (phisgo - xm1) * (phisgo - x) * (phisgo - xp1)
End Sub

Sub Find_New_eta()

    For i = 1 To m + 1
        La(i) = nactive * ds90s(i) / 1000
        If i = 1 Then
            qbTdev = au * (qbT(i) - qbTf) / dx + (1 - au) * (qbT(i + 1) - qbT(i)) / dx
            For jj = 1 To np
                If qbTdev > 0 Then
```

```
                FIexc(i, jj) = fs(jj)
            Else
                FIexc(i, jj) = atrans * f(i, jj) + (1 - atrans) * pl(i, jj)
            End If
            qjj1dev(jj) = au * (qbT(i) * pl(i, jj) - qbTf * plf(jj)) / dx + _
                (1 - au) * (qbT(i + 1) * pl(i + 1, jj) - qbT(i) * pl(i, jj)) / dx
            qjj2dev(jj) = FIexc(i, jj) * qbTdev
        Next jj
    Else
        If i = m + 1 Then
            qbTdev = (qbT(i) - qbT(i - 1)) / dx
            For jj = 1 To np
                If qbTdev > 0 Then
                    FIexc(i, jj) = fs(jj)
                Else
                    FIexc(i, jj) = atrans * f(i, jj) + (1 - atrans) * pl(i, jj)
                End If
                qjj1dev(jj) = (qbT(i) * pl(i, jj) - qbT(i - 1) * pl(i - 1, jj)) / dx
                qjj2dev(jj) = FIexc(i, jj) * qbTdev
            Next jj
        Else
            qbTdev = au * (qbT(i) - qbT(i - 1)) / dx + _
                (1 - au) * (qbT(i + 1) - qbT(i)) / dx
            For jj = 1 To np
                If qbTdev > 0 Then
                    FIexc(i, jj) = fs(jj)
                Else
                    FIexc(i, jj) = atrans * f(i, jj) + (1 - atrans) * pl(i, jj)
                End If
                qjj1dev(jj) = au * (qbT(i) * pl(i, jj) - qbT(i - 1) * pl(i - 1, jj)) / dx + _
                    (1 - au) * (qbT(i + 1) * pl(i + 1, jj) - qbT(i) * pl(i, jj)) / dx
                qjj2dev(jj) = FIexc(i, jj) * qbTdev
            Next jj
        End If
    End If
    etanew(i) = eta(i) + dt * (-qbTdev * Inter / (1 - lps))
    For jj = 1 To np
        Fnew(i, jj) = f(i, jj) + dt * Inter * (-qjj1dev(jj) + qjj2dev(jj)) / (1 - lps) / La(i)
        If time > 0 Then
            Fnew(i, jj) = Fnew(i, jj) + (FIexc(i, jj) - f(i, jj)) / La(i) * (La(i) - Laold(i))
        End If
    Next jj
  Next i
For i = 1 To m + 1
Laold(i) = La(i)
If i < m + 1 Then
eta(i) = etanew(i)
If eta(i) < etarock(i) Then
eta(i) = etarock(i)
End If
End If

    For jj = 1 To np
        Ft(jj) = Fnew(i, jj)
```

```
            If Ft(jj) < 0 Then Ft(jj) = 0
         Next jj

      Sum = 0
      For jj = 1 To np
         If Ft(jj) < 0 Then Ft(jj) = 0
         Sum = Sum + Ft(jj)
      Next jj

For jj = 1 To np
Ft(jj) = Ft(jj) / Sum
f(i, jj) = Ft(jj)
Next jj

dsgst = 0
For jj = 1 To np
dsgst = dsgst + Ft(jj) * psi(jj)

Next jj
dsgst = Exp(Log(2) * dsgst)

dsgs(i) = dsgst

Fft(1) = 100
For jj = 2 To npp
Fft(jj) = Fft(jj - 1) - Ft(jj - 1) * 100
Next jj

Z = False
jj = 0
Do
jj = jj + 1
If (Fft(jj) >= 90) And (Fft(jj + 1) < 90) Then Z = True
Loop Until Z
ds90st = Exp(Log(di(jj + 1)) + (Log(di(jj)) - Log(di(jj + 1))) / _
(Fft(jj) - Fft(jj + 1)) * (90 - Fft(jj + 1)))

Z = False
jj = 0
Do
jj = jj + 1
If (Fft(jj) >= 50) And (Fft(jj + 1) < 50) Then Z = True
Loop Until Z
ds50st = Exp(Log(di(jj + 1)) + (Log(di(jj)) - Log(di(jj + 1))) / _
(Fft(jj) - Fft(jj + 1)) * (50 - Fft(jj + 1)))


fracsandt = 0
For jj = 1 To np
If di(jj) > 2 And di(jj + 1) <= 2 Then
fracsandt = Fft(jj + 1) + (Fft(jj) - Fft(jj + 1)) / (Log(di(jj)) - Log(di(jj + 1))) * (Log(2) - Log(di(jj + 1)))
fracsandt = fracsandt / 100
End If
Next jj
```

```
ds90s(i) = ds90st
ds50s(i) = ds50st
fracsand(i) = fracsandt

Next i

time = time + dt
End Sub

Sub Send_Output()

Set ChaCha = Worksheets("1d Sed").Range("n5")

Set Inputs = Worksheets("1d Sed").Range("c6")
Inputs.Cells(13).Value = j

Set Elevation = Worksheets("1d Sed").Range("AJ7")
Set GrainSize = Elevation.Offset(0, 8)
Set Load = GrainSize.Offset(0, 8)
Set XYZ = Load.Offset(0, 8)

Elevation.Offset(-1) = "elev (m)"
GrainSize.Offset(-1) = "Dsg (mm)"
Load.Offset(-1) = "qbT/qbTf"

Elevation.Offset(-1, 2) = Elevation.Offset(-1, 3)
Load.Offset(-1, 2) = Elevation.Offset(-1, 3)
GrainSize.Offset(-1, 2) = Elevation.Offset(-1, 3)

Elevation.Offset(-1, 3) = "t = " + CStr(j) + "yr"
Load.Offset(-1, 3) = "t = " + CStr(j) + "yr"
GrainSize.Offset(-1, 3) = "t = " + CStr(j) + "yr"

For i = 1 To m + 1

If time = 0 Then

Elevation.Cells(i, 1) = xx(i)
GrainSize.Cells(i, 1) = xx(i)
Load.Cells(i, 1) = xx(i)

Elevation.Cells(i, 2).Value = eta(i)
GrainSize.Cells(i, 2) = dsgs(i)
Load.Cells(i, 2) = qbT(i) / qbTf

End If
Elevation.Cells(i, 3) = Elevation.Cells(i, 4)
GrainSize.Cells(i, 3) = GrainSize.Cells(i, 4)
Load.Cells(i, 3) = Load.Cells(i, 4)

Elevation.Cells(i, 4).Value = eta(i)
GrainSize.Cells(i, 4) = dsgs(i)
Load.Cells(i, 4) = qbT(i) / qbTf
```

```
Next i

R = 1
For i = 2 To m + 1

x1(i) = Elevation.Cells(i - 1, 1).Value
x2(i) = Elevation.Cells(i - 1, 1).Value
x3(i) = Elevation.Cells(i - 1, 1).Value
x4(i) = Elevation.Cells(i - 1, 1).Value
x5(i) = Elevation.Cells(i - 1, 1).Value
x6(i) = Elevation.Cells(i - 1, 1).Value
x7(i) = Elevation.Cells(i, 1).Value
x8(i) = Elevation.Cells(i - 1, 1).Value
x9(i) = Elevation.Cells(i - 1, 1).Value
x10(i) = Elevation.Cells(i, 1).Value

y1(i) = B0(i - 1) / 2
y2(i) = 50
y3(i) = B0(i - 1) / 2
y4(i) = -B0(i - 1) / 2
y5(i) = -50
y6(i) = -B0(i - 1) / 2
y7(i) = -B0(i) / 2
y8(i) = -B0(i - 1) / 2
y9(i) = B0(i - 1) / 2
y10(i) = B0(i) / 2

z1(i) = Elevation.Cells(i - 1, 4).Value
z2(i) = Elevation.Cells(i - 1, 4).Value + (50 - B0(i - 1) / 2) / sdsl(i - 1)
z3(i) = Elevation.Cells(i - 1, 4).Value
Z4(i) = Elevation.Cells(i - 1, 4).Value
z5(i) = Elevation.Cells(i - 1, 4).Value + (50 - B0(i - 1) / 2) / sdsl(i - 1)
z6(i) = Elevation.Cells(i - 1, 4).Value
z7(i) = Elevation.Cells(i, 4).Value
z8(i) = Elevation.Cells(i - 1, 4).Value
z9(i) = Elevation.Cells(i - 1, 4).Value
z10(i) = Elevation.Cells(i, 4).Value

XYZ.Cells(R, 3 * j + 1) = x1(i)
XYZ.Cells(R + 1, 3 * j + 1) = x2(i)
XYZ.Cells(R + 2, 3 * j + 1) = x3(i)
XYZ.Cells(R + 3, 3 * j + 1) = x4(i)
XYZ.Cells(R + 4, 3 * j + 1) = x5(i)
XYZ.Cells(R + 5, 3 * j + 1) = x6(i)
XYZ.Cells(R + 6, 3 * j + 1) = x7(i)
XYZ.Cells(R + 7, 3 * j + 1) = x8(i)
XYZ.Cells(R + 8, 3 * j + 1) = x9(i)
XYZ.Cells(R + 9, 3 * j + 1) = x10(i)

XYZ.Cells(R, 3 * j + 2) = y1(i)
XYZ.Cells(R + 1, 3 * j + 2) = y2(i)
XYZ.Cells(R + 2, 3 * j + 2) = y3(i)
XYZ.Cells(R + 3, 3 * j + 2) = y4(i)
XYZ.Cells(R + 4, 3 * j + 2) = y5(i)
```

```
XYZ.Cells(R + 5, 3 * j + 2) = y6(i)
XYZ.Cells(R + 6, 3 * j + 2) = y7(i)
XYZ.Cells(R + 7, 3 * j + 2) = y8(i)
XYZ.Cells(R + 8, 3 * j + 2) = y9(i)
XYZ.Cells(R + 9, 3 * j + 2) = y10(i)

XYZ.Cells(R, 3 * j + 3) = z1(i)
XYZ.Cells(R + 1, 3 * j + 3) = z2(i)
XYZ.Cells(R + 2, 3 * j + 3) = z3(i)
XYZ.Cells(R + 3, 3 * j + 3) = Z4(i)
XYZ.Cells(R + 4, 3 * j + 3) = z5(i)
XYZ.Cells(R + 5, 3 * j + 3) = z6(i)
XYZ.Cells(R + 6, 3 * j + 3) = z7(i)
XYZ.Cells(R + 7, 3 * j + 3) = z8(i)
XYZ.Cells(R + 8, 3 * j + 3) = z9(i)
XYZ.Cells(R + 9, 3 * j + 3) = z10(i)

If Worksheets("1d Sed").Range("c19").Value = True And j = 0 Then

x1(i) = Elevation.Cells(i - 1, 1).Value
x2(i) = Elevation.Cells(i - 1, 1).Value
x3(i) = Elevation.Cells(i - 1, 1).Value
x4(i) = Elevation.Cells(i - 1, 1).Value
x5(i) = Elevation.Cells(i - 1, 1).Value
x6(i) = Elevation.Cells(i - 1, 1).Value
x7(i) = Elevation.Cells(i, 1).Value
x8(i) = Elevation.Cells(i - 1, 1).Value
x9(i) = Elevation.Cells(i - 1, 1).Value
x10(i) = Elevation.Cells(i, 1).Value

y1(i) = B0(i - 1) / 2
y2(i) = 50
y3(i) = B0(i - 1) / 2
y4(i) = -B0(i - 1) / 2
y5(i) = -50
y6(i) = -B0(i - 1) / 2
y7(i) = -B0(i) / 2
y8(i) = -B0(i - 1) / 2
y9(i) = B0(i - 1) / 2
y10(i) = B0(i) / 2

z1(i) = ChaCha.Cells(i - 1, 4).Value
z2(i) = ChaCha.Cells(i - 1, 4).Value + (50 - B0(i - 1) / 2) / sdsl(i - 1)
z3(i) = ChaCha.Cells(i - 1, 4).Value
Z4(i) = ChaCha.Cells(i - 1, 4).Value
z5(i) = ChaCha.Cells(i - 1, 4).Value + (50 - B0(i - 1) / 2) / sdsl(i - 1)
z6(i) = ChaCha.Cells(i - 1, 4).Value
z7(i) = ChaCha.Cells(i, 4).Value
z8(i) = ChaCha.Cells(i - 1, 4).Value
z9(i) = ChaCha.Cells(i - 1, 4).Value
z10(i) = ChaCha.Cells(i, 4).Value

XYZ.Cells(R, 1).Offset(0, -3).Value = x1(i)
XYZ.Cells(R + 1, 1).Offset(0, -3).Value = x2(i)
```

```
XYZ.Cells(R + 2, 1).Offset(0, -3).Value = x3(i)
XYZ.Cells(R + 3, 1).Offset(0, -3).Value = x4(i)
XYZ.Cells(R + 4, 1).Offset(0, -3).Value = x5(i)
XYZ.Cells(R + 5, 1).Offset(0, -3).Value = x6(i)
XYZ.Cells(R + 6, 1).Offset(0, -3).Value = x7(i)
XYZ.Cells(R + 7, 1).Offset(0, -3).Value = x8(i)
XYZ.Cells(R + 8, 1).Offset(0, -3).Value = x9(i)
XYZ.Cells(R + 9, 1).Offset(0, -3).Value = x10(i)

XYZ.Cells(R, 2).Offset(0, -3).Value = y1(i)
XYZ.Cells(R + 1, 2).Offset(0, -3).Value = y2(i)
XYZ.Cells(R + 2, 2).Offset(0, -3).Value = y3(i)
XYZ.Cells(R + 3, 2).Offset(0, -3).Value = y4(i)
XYZ.Cells(R + 4, 2).Offset(0, -3).Value = y5(i)
XYZ.Cells(R + 5, 2).Offset(0, -3).Value = y6(i)
XYZ.Cells(R + 6, 2).Offset(0, -3).Value = y7(i)
XYZ.Cells(R + 7, 2).Offset(0, -3).Value = y8(i)
XYZ.Cells(R + 8, 2).Offset(0, -3).Value = y9(i)
XYZ.Cells(R + 9, 2).Offset(0, -3).Value = y10(i)

XYZ.Cells(R, 3).Offset(0, -3).Value = z1(i)
XYZ.Cells(R + 1, 3).Offset(0, -3).Value = z2(i)
XYZ.Cells(R + 2, 3).Offset(0, -3).Value = z3(i)
XYZ.Cells(R + 3, 3).Offset(0, -3).Value = Z4(i)
XYZ.Cells(R + 4, 3).Offset(0, -3).Value = z5(i)
XYZ.Cells(R + 5, 3).Offset(0, -3).Value = z6(i)
XYZ.Cells(R + 6, 3).Offset(0, -3).Value = z7(i)
XYZ.Cells(R + 7, 3).Offset(0, -3).Value = z8(i)
XYZ.Cells(R + 8, 3).Offset(0, -3).Value = z9(i)
XYZ.Cells(R + 9, 3).Offset(0, -3).Value = z10(i)

End If

R = 10 * i - 10
Next i
XYZ.Cells(1, 3 * j + 1).Offset(-1).Value = "t = " + CStr(Round(j, 0)) + "yr"

Application.Wait (True)

End Sub

Sub Output_Distribution()

Set Distribution = Worksheets("1d Sed").Range("ae8")

Distribution.Offset(-1, 0).Value = "Di mm"
Distribution.Offset(-1, 1).Value = "Ffs"
Distribution.Offset(-1, 2).Value = "pfeed"
Distribution.Offset(-1, 3).Value = "Ff"

For jj = 1 To np
plt(jj) = pl(1, jj)
Ft(jj) = f(1, jj)
Next jj
```

```
pfl1(1) = 100
For jj = 2 To npp
pfl1(jj) = pfl1(jj - 1) - plt(jj - 1) * 100
Next jj

Ff1(1) = 100
For jj = 2 To npp
Ff1(jj) = Ff1(jj - 1) - Ft(jj - 1) * 100
Next jj

For jj = 1 To npp
Distribution.Cells(jj, 1) = di(jj)
Distribution.Cells(jj, 2) = Ffs(jj)
Distribution.Cells(jj, 3) = pfl1(jj)
Distribution.Cells(jj, 4) = Ff1(jj)
Next jj
End Sub
```

```
Sub OneDimReeds()

Set Inputs = Worksheets("1d Veg").Range("y4")
Set Rules = Worksheets("1d Veg").Range("ai2")
Set CrossSec = Worksheets("1d Veg").Range("aq1")
Set Data = Worksheets("1d Veg").Range("ac2")

Set AverageDepth = Worksheets("1d Veg").Range("c23")
Set Variability = Worksheets("1d Veg").Range("c49")
Set MonthlyHeights = Worksheets("1d Wat").Range("ag7")
Set MonthlyHeightsd = Worksheets("1d Veg").Range("m4")

i = 1
Do Until CrossSec.Cells(1, i) = ""
i = i + 1
Loop

NP1 = i - 1

For i = 1 To NP1

For m = 1 To 12

MonthlyHeightsd.Cells(1, m).Value = MonthlyHeights.Cells(i, m).Value

Next m

R = 0
Do
R = R + 1
Loop Until Rules.Cells(R).Value = ""
NoRls = R - 1

n = 1
Do

x = Inputs.Cells(n).Value
y = Inputs.Cells(n, 2).Value

LinguisticVariables

DOF2 = 0
DOF1 = 0

Rule_no2 = 0
Rule_no1 = 0

Depths2 = "None"
FlowVariability2 = "None"

Depths1 = "None"
FlowVariability1 = "None"

R = 1
```

```
Do

Depths = Rules.Cells(R, 2)
If Depths = AverageDepth.Cells(1, 1) Then
Rules.Cells(R, 4) = AverageDepth.Cells(2, 1)
ElseIf Depths = AverageDepth.Cells(1, 2) Then
Rules.Cells(R, 4) = AverageDepth.Cells(2, 2)
ElseIf Depths = AverageDepth.Cells(1, 3) Then
Rules.Cells(R, 4) = AverageDepth.Cells(2, 3)
ElseIf Depths = AverageDepth.Cells(1, 4) Then
Rules.Cells(R, 4) = AverageDepth.Cells(2, 4)
ElseIf Depths = AverageDepth.Cells(1, 5) Then
Rules.Cells(R, 4) = AverageDepth.Cells(2, 5)
End If

FlowVariability = Rules.Cells(R, 3)
If FlowVariability = Variability.Cells(1, 1) Then
Rules.Cells(R, 5) = Variability.Cells(2, 1)
ElseIf FlowVariability = Variability.Cells(1, 2) Then
Rules.Cells(R, 5) = Variability.Cells(2, 2)
ElseIf FlowVariability = Variability.Cells(1, 3) Then
Rules.Cells(R, 5) = Variability.Cells(2, 3)
ElseIf FlowVariability = Variability.Cells(1, 4) Then
Rules.Cells(R, 5) = Variability.Cells(2, 4)
End If

R = R + 1
Loop Until R > NoRls + 1

R = 1
Do
If Rules.Cells(R, 6).Value > 0 Then

Depths2 = Depths1
FlowVariability2 = FlowVariability1

Depths1 = Rules.Cells(R, 2).Value
FlowVariability1 = Rules.Cells(R, 3).Value

DOF2 = DOF1
DOF1 = Rules.Cells(R, 6).Value

Rule_no2 = Rule_no1
Rule_no1 = Rules.Cells(R).Value

If x < Data.Cells(1, 1).Value Then
aa1 = Data.Cells(1, 3).Value
bb1 = Data.Cells(1, 4).Value
cc1 = Data.Cells(1, 5).Value

ElseIf x > Data.Cells(NoRls + 8, 1).Value Then
aa1 = Data.Cells(NoRls + 8, 3).Value
bb1 = Data.Cells(NoRls + 8, 4).Value
cc1 = Data.Cells(NoRls + 8, 5).Value
```

```
Else

PB1 = Data.Cells(R, 3).Value
PB2 = Data.Cells(R + 8, 3).Value
H1 = Data.Cells(R, 1).Value
H2 = Data.Cells(R + 8, 1).Value

aa2 = aa1
aa1 = (H2 - x) * (PB2 - PB1) / (H2 - H1) + PB1

PB1 = Data.Cells(R, 4).Value
PB2 = Data.Cells(R + 8, 4).Value
H1 = Data.Cells(R, 1).Value
H2 = Data.Cells(R + 8, 1).Value

bb2 = bb1
bb1 = (H2 - x) * (PB2 - PB1) / (H2 - H1) + PB1

PB1 = Data.Cells(R, 5).Value
PB2 = Data.Cells(R + 8, 5).Value
H1 = Data.Cells(R, 1).Value
H2 = Data.Cells(R + 8, 1).Value

cc2 = cc1
cc1 = (H2 - x) * (PB2 - PB1) / (H2 - H1) + PB1
End If

End If

R = R + 1
Loop Until R > NoRls + 1

L1 = (aa1 + bb1 + cc1) / 3
L2 = (aa1 + bb1 + cc1 - DOF1 * aa1 + 2 * DOF1 * bb1 - DOF1 * cc1) / 3
a1 = 0.5 * (cc1 - aa1)
a2 = 0.5 * (1 - DOF1) * (cc1 - aa1 + DOF1 * aa1 - DOF1 * cc1)
Area1 = a1 - a2
If Area1 = 0 Then
Centre1 = 0
Else
Centre1 = (L1 * a1 - L2 * a2) / (a1 - a2)
End If

L1 = (aa2 + bb2 + cc2) / 3
L2 = (aa2 + bb2 + cc2 - DOF2 * aa2 + 2 * DOF2 * bb2 - DOF2 * cc2) / 3
a1 = 0.5 * (cc2 - aa2)
a2 = 0.5 * (1 - DOF2) * (cc2 - aa2 + DOF2 * aa2 - DOF2 * cc2)
Area2 = a1 - a2
If Area2 = 0 Then
Centre2 = 0
Else
Centre2 = (L1 * a1 - L2 * a2) / (a1 - a2)
End If
```

```
If Area1 > 0 Then
COG = (Centre1 * Area1 + Centre2 * Area2) / (Area1 + Area2)
Else: COG = 0
End If

Inputs.Cells(n, 3).Value = COG
CrossSec.Cells(2, i).Cells(n).Value = COG

n = n + 1
Loop Until Inputs.Cells(n).Value = ""

 Next i

End Sub

Sub LinguisticVariables()

Set Rules = Worksheets("1d Veg").Range("ai2")
Set Depths = Worksheets("1d Veg").Range("b15")
Set DepthDots = Depths.Offset(9, 1)
Set Variabilities = Worksheets("1d Veg").Range("b43")
Set VariabilityDots = Variabilities.Offset(7, 1)
Set Data = Worksheets("1d Veg").Range("ac2")

a = Depths.Cells(2).Value
b = Depths.Cells(3).Value
c = Depths.Cells(4).Value
d = Depths.Cells(5).Value
e = Depths.Cells(6).Value

Select Case x
Case Is < a
y1 = 1
y2 = 0
y3 = 0
y4 = 0
y5 = 0
Case a To b
y1 = x / (a - b) + b / (b - a)
y2 = x / (b - a) + a / (a - b)
y3 = 0
y4 = 0
y5 = 0
Case b To c
y1 = 0
y2 = x / (b - c) + c / (c - b)
y3 = x / (c - b) + b / (b - c)
y4 = 0
y5 = 0
Case c To d
y1 = 0
y2 = 0
y3 = x / (c - d) + d / (d - c)
y4 = x / (d - c) + c / (c - d)
```

```
y5 = 0
Case d To e
y1 = 0
y2 = 0
y3 = 0
y4 = x / (d - e) + e / (e - d)
y5 = x / (e - d) + d / (d - e)
Case Is > e
y1 = 0
y2 = 0
y3 = 0
y4 = 0
y5 = 1
End Select

DepthDots.Cells(1, 1).Value = y1
DepthDots.Cells(1, 2).Value = y2
DepthDots.Cells(1, 3).Value = y3
DepthDots.Cells(1, 4).Value = y4
DepthDots.Cells(1, 5).Value = y5

If x < Data.Cells(1).Value Then
a = 0
b = 0
c = 0
d = 0
ElseIf x >= Data.Cells(NoRls).Value Then
a = Data.Cells(25, 2).Value
b = Data.Cells(26, 2).Value
c = Data.Cells(27, 2).Value
d = Data.Cells(28, 2).Value
Else

R = 1
Done = False
Do

H1 = Data.Cells(R, 1).Value
H2 = Data.Cells(R + 4, 1).Value

If x >= H1 And x < H2 Then

COV21 = Data.Cells(R + 4, 2).Value
COV22 = Data.Cells(R + 5, 2).Value
COV23 = Data.Cells(R + 6, 2).Value
COV24 = Data.Cells(R + 7, 2).Value

COV11 = Data.Cells(R, 2).Value
COV12 = Data.Cells(R + 1, 2).Value
COV13 = Data.Cells(R + 2, 2).Value
COV14 = Data.Cells(R + 3, 2).Value

a = (H2 - x) * (COV11 - COV21) / (H2 - H1) + COV21
b = (H2 - x) * (COV12 - COV22) / (H2 - H1) + COV22
```

```
c = (H2 - x) * (COV13 - COV23) / (H2 - H1) + COV23
d = (H2 - x) * (COV14 - COV24) / (H2 - H1) + COV24

Done = True
Else

R = R + 4
End If
Loop Until R > NoRls Or Done

End If

Variabilities.Cells(1).Value = a
Variabilities.Cells(2).Value = b
Variabilities.Cells(3).Value = c
Variabilities.Cells(4).Value = d

Amplitude
Application.Wait (True)
End Sub

Sub Amplitude()

Set Variabilities = Worksheets("1d Veg").Range("b43")
Set VariabilityDots = Variabilities.Offset(7, 1)

a = Variabilities.Cells(1).Value
b = Variabilities.Cells(2).Value
c = Variabilities.Cells(3).Value
d = Variabilities.Cells(4).Value

Select Case y
Case Is < a
y1 = 1
y2 = 0
y3 = 0
y4 = 0
Case a To b
y1 = y / (a - b) + b / (b - a)
y2 = y / (b - a) + a / (a - b)
y3 = 0
y4 = 0
Case b To c
y1 = 0
y2 = y / (b - c) + c / (c - b)
y3 = y / (c - b) + b / (b - c)
y4 = 0
Case c To d
y1 = 0
y2 = 0
y3 = y / (c - d) + d / (d - c)
y4 = y / (d - c) + c / (c - d)
Case Is > d
y1 = 0
```

```
y2 = 0
y3 = 0
y4 = 1
End Select

VariabilityDots.Cells(1, 1).Value = y1
VariabilityDots.Cells(1, 2).Value = y2
VariabilityDots.Cells(1, 3).Value = y3
VariabilityDots.Cells(1, 4).Value = y4

End Sub
```

```
Sub TwoDimWaterFlow()

Set GrainSize = Worksheets("1d Sed").Range("as7")
Set SedimentTopography = Worksheets("2d Sed").Range("f3:y22")
Set ConcreteRock = SedimentTopography.Offset(0, jmax + 8)
Set PeakFlow = Worksheets("1d Wat").Range("x7")
Set Characteristics = Worksheets("1d Sed").Range("n5")
Set Elevation = Worksheets("1d Sed").Range("AJ7")

Set Topography = Worksheets("2d Wat").Range("i3:ab22")
imax = Topography.Rows.Count
jmax = Topography.Columns.Count
Set Roughness = Topography.Offset(0, jmax + 4)
Set Heights = Topography.Offset(imax + 4, 0)
Set Velocityx = Heights.Offset(0, jmax + 4)
Set Velocityy = Velocityx.Offset(0, jmax + 4)

Set Inputs = Worksheets("2d Wat").Range("B1")

CRSC = Inputs.Cells(3, 4).Value

Inputs.Cells(3).Value = PeakFlow.Cells(CRSC, 3).Value
Inputs.Cells(4).Value = PeakFlow.Cells(CRSC + 1, 3).Value

Inputs.Cells(5).Value = PeakFlow.Cells(CRSC, 2).Value + Elevation.Cells(CRSC, 4).Value
Inputs.Cells(6).Value = PeakFlow.Cells(CRSC + 1, 2).Value + Elevation.Cells(CRSC + 1, 4).Value

Inputs.Cells(7).Value = PeakFlow.Cells(CRSC, 4).Value
Inputs.Cells(8).Value = PeakFlow.Cells(CRSC + 1, 4).Value

U1 = Inputs.Cells(3).Value
U2 = Inputs.Cells(4).Value

H1 = Inputs.Cells(5).Value
H2 = Inputs.Cells(6).Value

Inputs.Cells(5, 4).Value = Characteristics.Cells(CRSC, 1).Value
Inputs.Cells(6, 4).Value = Characteristics.Cells(CRSC + 1, 1).Value

Inputs.Cells(7).Value = Characteristics.Cells(CRSC, 2).Value
Inputs.Cells(8).Value = Characteristics.Cells(CRSC + 1, 2).Value

B1 = Inputs.Cells(7).Value
B2 = Inputs.Cells(8).Value

sdsl1 = Inputs.Cells(5, 4).Value
sdsl2 = Inputs.Cells(6, 4).Value

e1 = Inputs.Cells(17).Value 'stabilizes shallow flows
e2 = Inputs.Cells(18).Value '0.2-1.0 is typical
e3 = Inputs.Cells(19).Value '0.1 is typical

x = Inputs.Cells(9).Value
y = Inputs.Cells(10).Value
```

```
dt = Inputs.Cells(11).Value

t = Inputs.Cells(12).Value
tLast = Inputs.Cells(13).Value

ArtVis = Inputs.Cells(14).Value
HZo = Inputs.Cells(15).Value

ti = t
f = 1

vchart = Inputs.Cells(22).Value

ip = 1
n = 0

iprint = Inputs.Cells(24).Value
Ntoprint = Inputs.Cells(25).Value

ds = GrainSize.Cells(CRSC, 3) / 1000

ok = False

Do While t <= tLast

If t = 0 Then

For j = 1 To jmax
For i = 1 To imax

H(i, j) = H1 - j * (H1 - H2) / jmax

U(i, j) = U1 - j * (U1 - U2) / jmax

V(i, j) = 0

Heights.Cells(i, j).Value = H(i, j)
Velocityx.Cells(i, j).Value = U(i, j)
Velocityy.Cells(i, j).Value = V(i, j)
ks(i, j) = 2 * ds
Roughness.Cells(i, j).Value = ks(i, j)
Next i
Next j

End If

For j = 1 To jmax
For i = 1 To imax

Z(i, j) = SedimentTopography.Cells(i, j).Value
R(i, j) = ConcreteRock.Cells(i, j).Value
Topography.Cells(i, j).Value = Z(i, j)
ks(i, j) = Roughness.Cells(i, j).Value
H(i, j) = Heights.Cells(i, j).Value
```

```
U(i, j) = Velocityx.Cells(i, j).Value
V(i, j) = Velocityy.Cells(i, j).Value

Next i
Next j

TotalArea1 = (B1 + PeakFlow.Cells(CRSC, 2).Value * sdsl1) * PeakFlow.Cells(CRSC, 2).Value

Do

TotalArea2 = 0
Inflow = 0
For i = 1 To imax
If H(i, 2) - Z(i, 2) > 0 Then
TotalArea2 = TotalArea2 + (H(i, 2) - Z(i, 2)) * y
Inflow = Inflow + 1
End If
Next i
For i = 1 To imax
If H(i, 2) - Z(i, 2) > 0 Then
H(i, 2) = H(i, 2) + 0.1 * (TotalArea1 - TotalArea2) / Inflow
End If
Next i
Loop Until (TotalArea2 - TotalArea1) < 0.1 And (TotalArea2 - TotalArea1) > -0.1

For i = 1 To imax
H(i, 1) = H(i, 2)
Next i

If f = 5 Then
f = 1
ElseIf f = 4 Then
pux2 = 2: pux1 = 0: puy2 = 2: puy1 = 0
cux2 = 0: cux1 = -2: cuy2 = 0: cuy1 = -2
pvx2 = 2: pvx1 = 0: pvy2 = 2: pvy1 = 0
cvx2 = 0: cvx1 = -2: cvy2 = 0: cvy1 = -2
f = 5
ElseIf f = 3 Then
pux2 = 2: pux1 = 0: puy2 = 0: puy1 = -2:
cux2 = 0: cux1 = -2: cuy2 = 2: cuy1 = 0
pvx2 = 2: pvx1 = 0: pvy2 = 0: pvy1 = -2
cvx2 = 0: cvx1 = -2: cvy2 = 2: cvy1 = 0
f = 4
ElseIf f = 2 Then
pux2 = 0: pux1 = -2: puy2 = 0: puy1 = -2
cux2 = 2: cux1 = 0: cuy2 = 2: cuy1 = 0
pvx2 = 2: pvx1 = 0: pvy2 = 2: pvy1 = 0
cvx2 = 0: cvx1 = -2: cvy2 = 0: cvy1 = -2
f = 3
ElseIf f = 1 Then
pux2 = 2: pux1 = 0: puy2 = 2: puy1 = 0
cux2 = 0: cux1 = -2: cuy2 = 0: cuy1 = -2
pvx2 = 0: pvx1 = -2: pvy2 = 0: pvy1 = -2
cvx2 = 2: cvx1 = 0: cvy2 = 2: cvy1 = 0
```

```
f = 2
End If

DTMIN = 1000
FLAG = 1

Do While FLAG = 1

'Initial values
For j = 1 To jmax
For i = 1 To imax
Ha(i, j) = H(i, j)
Ua(i, j) = U(i, j)
Va(i, j) = V(i, j)
Next i
Next j

For j = 2 To jmax - 1
For i = 2 To imax - 1
HZ(i, j) = H(i, j) - 0.25 * (Z(i - 1, j - 1) + Z(i + 1, j - 1) + Z(i - 1, j + 1) + Z(i + 1, j + 1))

HU(i, j) = 0.5 * (H(i, j - 1) + H(i, j + 1)) - 0.5 * (Z(i - 1, j) + Z(i + 1, j))
HV(i, j) = 0.5 * (H(i - 1, j) + H(i + 1, j)) - 0.5 * (Z(i, j - 1) + Z(i, j + 1))
Um(i, j) = 0.25 * (U(i - 1, j - 1) + U(i + 1, j - 1) + U(i - 1, j + 1) + U(i + 1, j + 1))
Vm(i, j) = 0.25 * (V(i + 1, j + 1) + V(i - 1, j - 1) + V(i - 1, j + 1) + V(i + 1, j - 1))
HB(i, j) = 0.25 * (H(i + 1, j) + H(i - 1, j) + H(i, j - 1) + H(i, j + 1)) - Z(i, j)

If HZ(i, j) <= HZo Then
HZ(i, j) = 0.9 * HZo
End If
If HU(i, j) < HZo Then
HU(i, j) = 0.9 * HZo
End If
If HV(i, j) < HZo Then
HV(i, j) = 0.9 * HZo
End If
If HB(i, j) < HZo Then
HB(i, j) = 0.9 * HZo
End If
Next i
Next j

For j = 1 To jmax
For i = 1 To imax
Select Case j
Case Is < 2
HV(i, j) = HV(i, j + 1)
Um(i, j) = Um(i, j + 1)
HV(i, j) = HV(i, j + 1)
Um(i, j) = Um(i, j + 1)
HB(i, j) = HB(i, j + 1)
Case Is > jmax - 1
HU(i, j) = HU(i, j - 1)
Vm(i, j) = Vm(i, j - 1)
```

```
HV(i, j) = HV(i, j - 1)
Um(i, j) = Um(i, j - 1)
HB(i, j) = HB(i, j - 1)
End Select

Select Case i
Case Is < 2
HU(i, j) = HU(i + 1, j)
Vm(i, j) = Vm(i + 1, j)
HV(i, j) = HV(i + 1, j)
Um(i, j) = Um(i + 1, j)
HB(i, j) = HB(i + 1, j)
Case Is = imax - 1
HU(i, j) = HU(i - 1, j)
Vm(i, j) = Vm(i - 1, j)
HV(i, j) = HV(i - 1, j)
Um(i, j) = Um(i - 1, j)
HB(i, j) = HB(i - 1, j)
End Select
Next i
Next j

'Eddy viscosity vt
For j = 1 To jmax
For i = 1 To imax

If ks(i, j) <= 0 Then
vt(i, j) = 0
Else
'For very small depth to roughness ratios:

If HZ(i, j) < (ks(i, j) * 2.71828182845905 ^ 2 / 12) Then
HZ(i, j) = 0.9 * HZo
End If
If HZ(i, j) < ks(i, j) Then
ks(i, j) = HZ(i, j)
End If

ld = 0.1 * HZ(i, j)

If j < 3 Or j > jmax - 2 Or i < 3 Or i > imax - 2 Then
Term1 = 0
Else
Term2 = 1 / (2 * y) * ((U(i + 2, j - 1) - U(i + 2, j + 1)) / 2 - (U(i - 2, j - 1) - U(i - 2, j + 1)) / 2) _
+ 1 / (2 * x) * ((V(i - 1, j + 2) - V(i + 1, j + 2)) / 2 - (V(i - 1, j - 2) - V(i + 1, j - 2)) / 2)
Term1 = e3 ^ 2 * HZ(i, j) ^ 2 * (2 * (1 / x * (U(i, j + 1) - U(i, j - 1))) ^ 2 + Term2 ^ 2 _
+ 2 * (1 / y * (V(i + 1, j) - V(i - 1, j)) ^ 2)) ^ 0.5
End If

cs(i, j) = 5.75 * WorksheetFunction.Log10(12 * HZ(i, j) / ks(i, j))

vt(i, j) = e1 + e2 * HZ(i, j) * (Um(i, j) ^ 2 + Vm(i, j) ^ 2) ^ 0.5 / cs(i, j) _
+ Term1
```

```
End If

cu = 0.09
cd = 0.17
ld = 0.1 * HZ(i, j)
k(i, j) = (cd / cu * vt(i, j) / ld)
e(i, j) = cd * k(i, j) ^ (3 / 2) / ld

Next i
Next j

For j = 2 To jmax - 1
For i = 2 To imax - 1
vtm(i, j) = 0.25 * (vt(i - 1, j - 1) + vt(i + 1, j - 1) + vt(i - 1, j + 1) + vt(i + 1, j + 1))
Next i
Next j

For j = 3 To jmax - 2
For i = 3 To imax - 2
'FRICTION SLOPE (WU, 2004):
Sfx = (U(i, j) * (U(i, j) ^ 2 + Vm(i, j) ^ 2) ^ 0.5) / HZ(i, j) / cs(i, j) ^ 2 / g / x
Sfy = (V(i, j) * (Um(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5) / HZ(i, j) / cs(i, j) ^ 2 / g / y
'FLOW DEPTH:
Hs(i, j, 1) = Ha(i, j) + dt * ( _
-(1 / x * (HU(i, j + 1) * U(i, j + 1) - HU(i, j - 1) * U(i, j - 1))) _
- (1 / y * (HV(i + 1, j) * V(i + 1, j) - HV(i - 1, j) * V(i - 1, j))))
'FLOW VELOCITY U IN THE X DIRECTION:
'Predictor step
Us(i, j, 1) = Ua(i, j) + dt * ( _
-(U(i, j) * (U(i, j + pux2) - U(i, j + pux1)) / x _
+ Vm(i, j) * (U(i + puy2, j) - U(i + puy1, j)) / y) _
- (g * (H(i, j + 1) - H(i, j - 1)) / x + g * Sfx) _
+ ((1 / x ^ 2 / HU(i, j)) * (2 * HZ(i, j + 1) * vt(i, j + 1) * (U(i, j + 2) - U(i, j)) _
- 2 * HZ(i, j - 1) * vt(i, j - 1) * (U(i, j) - U(i, j - 2)))) _
+ ((1 / HU(i, j) / x) * (2 / 3 * HZ(i, j + 1) * k(i, j + 1) - 2 / 3 * HZ(i, j - 1) * k(i, j - 1))) _
+ ((1 / y ^ 2 / HU(i, j)) * (HB(i + 1, j) * vtm(i + 1, j) * (U(i + 2, j) - U(i, j)) _
- HB(i - 1, j) * vtm(i - 1, j) * (U(i, j) - U(i - 2, j)))) _
+ ((1 / (x * y) / HU(i, j)) * (HB(i + 1, j) * vtm(i + 1, j) * (V(i + 1, j + 1) - V(i + 1, j - 1)) _
- HB(i - 1, j) * vtm(i - 1, j) * (V(i - 1, j + 1) - V(i - 1, j - 1))))
'FLOW VELOCITY U IN THE Y DIRECTION:
'Predictor step
vs(i, j, 1) = Va(i, j) + dt * ( _
-(Um(i, j) * (V(i, j + pvx2) - V(i, j + pvx1)) / x _
+ V(i, j) * (V(i + pvy2, j) - V(i + pvy1, j)) / y) _
- (g * (H(i + 1, j) - H(i - 1, j)) + g * Sfy) _
+ ((1 / (x * y) / HV(i, j)) * (HB(i, j + 1) * vtm(i, j + 1) * (U(i + 1, j + 1) - U(i - 1, j + 1)) _
- HB(i, j - 1) * vtm(i, j - 1) * (U(i + 1, j - 1) - U(i - 1, j - 1)))) _
+ ((1 / x ^ 2 / HV(i, j)) * (HB(i, j + 1) * vtm(i, j + 1) * (V(i, j + 2) - V(i, j)) _
- HB(i, j - 1) * vtm(i, j - 1) * (V(i, j) - V(i, j - 2)))) _
+ ((1 / y ^ 2 / HV(i, j)) * (2 * HZ(i + 1, j) * vt(i + 1, j) * (V(i + 2, j) - V(i, j)) _
- 2 * HZ(i - 1, j) * vt(i - 1, j) * (V(i, j) - V(i - 2, j)))) _
+ ((1 / HV(i, j) / y) * (2 / 3 * HZ(i + 1, j) * k(i + 1, j) - 2 / 3 * HZ(i - 1, j) * k(i - 1, j))))

'Store predictor values to corrector values
```

```
H(i, j) = Hs(i, j, 1)
U(i, j) = Us(i, j, 1)
V(i, j) = vs(i, j, 1)
Next i
Next j

For j = 3 To jmax - 2
For i = 3 To imax - 2
'FRICTION SLOPE (WU, 2004):
Sfx = (U(i, j) * (U(i, j) ^ 2 + Vm(i, j) ^ 2) ^ 0.5) / HZ(i, j) / cs(i, j) ^ 2 / g / x
Sfy = (V(i, j) * (Um(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5) / HZ(i, j) / cs(i, j) ^ 2 / g / y
'FLOW DEPTH:
'CORRECTOR STEP
Hs(i, j, 2) = Ha(i, j) + dt * ( _
-(1 / x * (HU(i, j + 1) * U(i, j + 1) - HU(i, j - 1) * U(i, j - 1))) _
- (1 / y * (HV(i + 1, j) * V(i + 1, j) - HV(i - 1, j) * V(i - 1, j))))
'FLOW VELOCITY U IN THE X DIRECTION:
'CORRECTOR STEP
Us(i, j, 2) = Ua(i, j) + dt * ( _
-(U(i, j) * (U(i, j - pux2) - U(i, j - pux1)) / x _
+ Vm(i, j) * (U(i - puy2, j) - U(i - puy1, j)) / y) _
- (g * (H(i, j + 1) - H(i, j - 1)) / x + g * Sfx) _
+ ((1 / x ^ 2 / HU(i, j)) * (2 * HZ(i, j + 1) * vt(i, j + 1) * (U(i, j + 2) - U(i, j)) _
- 2 * HZ(i, j - 1) * vt(i, j - 1) * (U(i, j) - U(i, j - 2)))) _
+ ((1 / HU(i, j) / x) * (2 / 3 * HZ(i, j + 1) * k(i, j + 1) - 2 / 3 * HZ(i, j - 1) * k(i, j - 1))) _
+ ((1 / y ^ 2 / HU(i, j)) * (HB(i + 1, j) * vtm(i + 1, j) * (U(i + 2, j) - U(i, j)) _
- HB(i - 1, j) * vtm(i - 1, j) * (U(i, j) - U(i - 2, j)))) _
+ ((1 / (x * y) / HU(i, j)) * (HB(i + 1, j) * vtm(i + 1, j) * (V(i + 1, j + 1) - V(i + 1, j - 1)) _
- HB(i - 1, j) * vtm(i - 1, j) * (V(i - 1, j + 1) - V(i - 1, j - 1))))
'FLOW VELOCITY U IN THE Y DIRECTION:
'CORRECTOR STEP
vs(i, j, 2) = Va(i, j) + dt * ( _
-(Um(i, j) * (V(i, j - pvx2) - V(i, j - pvx1)) / x _
+ V(i, j) * (V(i - pvy2, j) - V(i - pvy1, j)) / y) _
- (g * (H(i + 1, j) - H(i - 1, j)) + g * Sfy) _
+ ((1 / (x * y) / HV(i, j)) * (HB(i, j + 1) * vtm(i, j + 1) * (U(i + 1, j + 1) - U(i - 1, j + 1)) _
- HB(i, j - 1) * vtm(i, j - 1) * (U(i + 1, j - 1) - U(i - 1, j - 1)))) _
+ ((1 / x ^ 2 / HV(i, j)) * (HB(i, j + 1) * vtm(i, j + 1) * (V(i, j + 2) - V(i, j)) _
- HB(i, j - 1) * vtm(i, j - 1) * (V(i, j) - V(i, j - 2)))) _
+ ((1 / y ^ 2 / HV(i, j)) * (2 * HZ(i + 1, j) * vt(i + 1, j) * (V(i + 2, j) - V(i, j)) _
- 2 * HZ(i - 1, j) * vt(i - 1, j) * (V(i, j) - V(i - 2, j)))) _
+ ((1 / HV(i, j) / y) * (2 / 3 * HZ(i + 1, j) * k(i + 1, j) - 2 / 3 * HZ(i - 1, j) * k(i - 1, j))))

H(i, j) = 0.5 * (Hs(i, j, 1) + Hs(i, j, 2))
If H(i, j) < HZo Then
H(i, j) = 0.9 * HZo
End If

U(i, j) = 0.5 * (Us(i, j, 1) + Us(i, j, 2))
V(i, j) = 0.5 * (vs(i, j, 1) + vs(i, j, 2))

Next i
Next j
```

```
FLAG = 0

For j = 2 To jmax - 1
For i = 2 To imax - 1
Sox(i, j) = (H(i, j + 1) - H(i, j - 1)) / (2 * x)
Soy(i, j) = (H(i + 1, j) - H(i - 1, j)) / (2 * y)
Next i
Next j

For j = 2 To jmax - 1
For i = 2 To imax - 1
If Sox(i, j + 1) = 0 And Sox(i, j) = 0 And Sox(i, j - 1) = 0 Then
vx(i, j) = 0
Else
vx(i, j) = (Sox(i, j + 1) - 2 * Sox(i, j) + Sox(i, j - 1)) / (Abs(Sox(i, j + 1)) + 2 * Abs(Sox(i, j)) +
Abs(Sox(i, j - 1)))
End If
If Soy(i + 1, j) = 0 And Soy(i, j) = 0 And Soy(i - 1, j) = 0 Then
vy(i, j) = 0
Else
vy(i, j) = (Soy(i + 1, j) - 2 * Soy(i, j) + Soy(i - 1, j)) / (Abs(Soy(i + 1, j)) + 2 * Abs(Soy(i, j)) +
Abs(Soy(i - 1, j)))
End If
Next i
Next j

For j = 3 To jmax - 2
For i = 3 To imax - 2

Ex2 = ArtVis * WorksheetFunction.Max(vx(i, j + 1), vx(i, j))
Ex1 = ArtVis * WorksheetFunction.Max(vx(i, j), vx(i, j - 1))
Ey2 = ArtVis * WorksheetFunction.Max(vy(i + 1, j), vy(i, j))
Ey1 = ArtVis * WorksheetFunction.Max(vy(i, j), vy(i - 1, j))

H(i, j) = H(i, j) + Ex2 * (H(i, j + 1) - H(i, j)) - Ex1 * (H(i, j) - H(i, j - 1)) + Ey2 * (H(i, j + 1) - H(i, j)) -
Ey1 * (H(i, j) - H(i, j - 1))
If H(i, j) < HZo Then
H(i, j) = 0.9 * HZo
End If
U(i, j) = U(i, j) + Ex2 * (U(i, j + 1) - U(i, j)) - Ex1 * (U(i, j) - U(i, j - 1)) + Ey2 * (U(i, j + 1) - U(i, j)) -
Ey1 * (U(i, j) - U(i, j - 1))
V(i, j) = V(i, j) + Ex2 * (V(i, j + 1) - V(i, j)) - Ex1 * (V(i, j) - V(i, j - 1)) + Ey2 * (V(i, j + 1) - V(i, j)) -
Ey1 * (V(i, j) - V(i, j - 1))
Next i
Next j

For j = 1 To jmax
For i = 1 To imax
Select Case i
Case Is < 3
U(i, j) = 0
H(i, j) = H(i + 2, j)
If H(i, j) - Z(i, j) > 0 Then
```

```
V(i, j) = V(i + 2, j)
Else
V(i, j) = 0
End If
Case Is > imax - 2
U(i, j) = 0
H(i, j) = H(i - 2, j)
If H(i, j) - Z(i, j) > 0 Then
V(i, j) = V(i - 2, j)
Else
V(i, j) = 0
End If
End Select

Select Case j
Case Is < 3
V(i, j) = 0
H(i, j) = H1
If H(i, j) - Z(i, j) > 0 Then
U(i, j) = U1
Else
U(i, j) = 0
End If
Case Is > jmax - 2
V(i, j) = 0
H(i, j) = H2
If H(i, j) - Z(i, j) > 0 Then
U(i, j) = U2
Else
U(i, j) = 0
End If
End Select
Next i
Next j

For j = 2 To jmax - 1
For i = 2 To imax - 1
HZ(i, j) = H(i, j) - 0.25 * (Z(i - 1, j - 1) + Z(i + 1, j - 1) + Z(i - 1, j + 1) + Z(i + 1, j + 1))
If HZ(i, j) <= HZo Then
HZ(i, j) = 0.9 * HZo
End If
If H(i, j) - Z(i, j) > 0 Then
ReedRoughness
End If
Next i
Next j

For j = 3 To jmax - 2
For i = 3 To imax - 2

If HZ(i, j) <= HZo Then
If HZ(i - 1, j + 1) <= HZo Then
Ind4 = 0
Else
```

```
Ind4 = 1
End If
If HZ(i + 1, j + 1) <= HZo Then
Ind1 = 0
Else
Ind1 = 1
End If
If HZ(i + 1, j - 1) <= HZo Then
Ind2 = 0
Else
Ind2 = 1
End If
If HZ(i - 1, j - 1) <= HZo Then
Ind3 = 0
Else
Ind3 = 1
End If

If (Ind1 + Ind2 + Ind3 + Ind4) = 3 Then

H(i, j) = (Ha(i, j) + 1 / (Ind1 + Ind2 + Ind3 + Ind4) * (Ind4 * Ha(i - 1, j + 1) + Ind2 * Ha(i + 1, j - 1) +
Ind3 * Ha(i - 1, j - 1) + Ind1 * Ha(i + 1, j + 1))) / 2
V(i, j) = 1 / (Ind1 + Ind2 + Ind3 + Ind4) * (Ind4 * 1 / 3 * (Va(i - 1, j + 1) - 6 * Va(i - 2, j + 2)) + Ind3 *
1 / 3 * (Va(i - 1, j - 1) - 6 * Va(i - 2, j - 2)) + Ind2 * 1 / 3 * (Va(i + 1, j - 1) - 6 * Va(i + 2, j - 2)) + Ind1
* 1 / 3 * (Va(i + 1, j + 1) - 6 * Va(i + 2, j + 2)))
U(i, j) = 1 / (Ind1 + Ind2 + Ind3 + Ind4) * (Ind4 * 1 / 3 * (Ua(i - 1, j + 1) - 6 * Ua(i - 2, j + 2)) + Ind3 *
1 / 3 * (Ua(i - 1, j - 1) - 6 * Ua(i - 2, j - 2)) + Ind2 * 1 / 3 * (Ua(i + 1, j - 1) - 6 * Ua(i + 2, j - 2)) + Ind1
* 1 / 3 * (Ua(i + 1, j + 1) - 6 * Ua(i + 2, j + 2)))

ElseIf (Ind1 + Ind2 + Ind3 + Ind4) = 2 Then

If (Ind1 + Ind2) = 0 Then
H(i + 1, j) = (Ha(i + 1, j) + Ha(i, j)) / 2
H(i, j) = (Ha(i, j) + Ha(i - 1, j)) / 2
V(i + 2, j) = 0
V(i, j) = 1 / 3 * (Va(i - 1, j) - 6 * Va(i - 2, j))
V(i + 1, j) = 1 / 3 * (Va(i, j) - 6 * Va(i - 1, j))
V(i - 1, j) = 0
U(i + 2, j) = 0
U(i, j) = 1 / 3 * (Ua(i - 1, j) - 6 * Ua(i - 2, j))
U(i + 1, j) = 1 / 3 * (Ua(i, j) - 6 * Ua(i - 1, j))
U(i - 1, j) = 0
ElseIf (Ind3 + Ind4) = 0 Then
H(i - 1, j) = (Ha(i - 1, j) + Ha(i, j)) / 2
H(i, j) = (Ha(i, j) + Ha(i + 1, j)) / 2
V(i - 2, j) = 0
V(i, j) = 1 / 3 * (Va(i + 1, j) - 6 * Va(i + 2, j))
V(i - 1, j) = 1 / 3 * (Va(i, j) - 6 * Va(i + 1, j))
V(i + 1, j) = 0
U(i - 2, j) = 0
U(i, j) = 1 / 3 * (Ua(i + 1, j) - 6 * Ua(i + 2, j))
U(i - 1, j) = 1 / 3 * (Ua(i, j) - 6 * Ua(i + 1, j))
U(i + 1, j) = 0
ElseIf (Ind1 + Ind4) = 0 Then
```

```
H(i, j + 1) = (Ha(i, j + 1) + Ha(i, j)) / 2
H(i, j) = (Ha(i, j) + Ha(i, j - 1)) / 2
V(i, j + 2) = 0
V(i, j) = 1 / 3 * (Va(i, j - 1) - 6 * Va(i, j - 2))
V(i, j + 1) = 1 / 3 * (Va(i, j) - 6 * Va(i, j - 1))
V(i, j - 1) = 0
U(i, j + 2) = 0
U(i, j) = 1 / 3 * (Ua(i, j - 1) - 6 * Ua(i, j - 2))
U(i, j + 1) = 1 / 3 * (Ua(i, j) - 6 * Ua(i, j - 1))
U(i, j - 1) = 0
ElseIf (Ind2 + Ind3) = 0 Then
H(i, j - 1) = (Ha(i, j - 1) + Ha(i, j)) / 2
H(i, j) = (Ha(i, j) + Ha(i, j + 1)) / 2
U(i, j + 2) = 0
V(i, j) = 1 / 3 * (Va(i, j + 1) - 6 * Va(i, j + 2))
V(i, j - 1) = 1 / 3 * (Va(i, j) - 6 * Va(i, j + 1))
V(i, j + 1) = 0
U(i, j - 2) = 0
U(i, j) = 1 / 3 * (Ua(i, j + 1) - 6 * Ua(i, j + 2))
U(i, j - 1) = 1 / 3 * (Ua(i, j) - 6 * Ua(i, j + 1))
U(i, j + 1) = 0
End If

ElseIf (Ind1 + Ind2 + Ind3 + Ind4) = 1 Then

H(i, j) = (Ha(i, j) + 1 / (Ind1 + Ind2 + Ind3 + Ind4) * (Ind4 * Ha(i - 1, j + 1) + Ind2 * Ha(i + 1, j - 1) +
Ind3 * Ha(i - 1, j - 1) + Ind1 * Ha(i + 1, j + 1))) / 2
V(i, j) = 1 / (Ind1 + Ind2 + Ind3 + Ind4) * (Ind4 * 1 / 3 * (Va(i - 1, j + 1) - 6 * Va(i - 2, j + 2)) + Ind3 *
1 / 3 * (Va(i - 1, j - 1) - 6 * Va(i - 2, j - 2)) + Ind2 * 1 / 3 * (Va(i + 1, j - 1) - 6 * Va(i + 2, j - 2)) + Ind1
* 1 / 3 * (Va(i + 1, j + 1) - 6 * Va(i + 2, j + 2)))
U(i, j) = 1 / (Ind1 + Ind2 + Ind3 + Ind4) * (Ind4 * 1 / 3 * (Ua(i - 1, j + 1) - 6 * Ua(i - 2, j + 2)) + Ind3 *
1 / 3 * (Ua(i - 1, j - 1) - 6 * Ua(i - 2, j - 2)) + Ind2 * 1 / 3 * (Ua(i + 1, j - 1) - 6 * Ua(i + 2, j - 2)) + Ind1
* 1 / 3 * (Ua(i + 1, j + 1) - 6 * Ua(i + 2, j + 2)))

If Ind1 = 1 Then
H(i + 1, j + 1) = (Ha(i + 1, j + 1) + Ha(i, j)) / 2
U(i + 1, j + 1) = Ua(i, j)
V(i + 1, j + 1) = Va(i, j)
ElseIf Ind2 = 1 Then
H(i + 1, j - 1) = (Ha(i + 1, j - 1) + Ha(i, j)) / 2
U(i + 1, j - 1) = Ua(i, j)
V(i + 1, j - 1) = Va(i, j)
ElseIf Ind3 = 1 Then
H(i - 1, j - 1) = (Ha(i - 1, j - 1) + Ha(i, j)) / 2
U(i - 1, j - 1) = Ua(i, j)
V(i - 1, j - 1) = Va(i, j)
ElseIf Ind4 = 1 Then
H(i - 1, j + 1) = (Ha(i - 1, j + 1) + Ha(i, j)) / 2
U(i - 1, j + 1) = Ua(i, j)
V(i - 1, j + 1) = Va(i, j)
End If

ElseIf (Ind1 + Ind2 + Ind3 + Ind4) = 0 Then
H(i, j) = 0.25 * (H(i + 1, j) + H(i - 1, j) + H(i, j - 1) + H(i, j + 1))
```

```
V(i, j) = 0
U(i, j) = 0

End If
End If
Next i
Next j

For j = 2 To jmax - 1
For i = 2 To imax - 1
If Abs(U(i, j)) > vchart Then
U(i, j) = vchart * U(i, j) / Abs(U(i, j))
End If
If Abs(V(i, j)) > vchart Then
V(i, j) = vchart * V(i, j) / Abs(V(i, j))
End If
Next i
Next j


For j = 1 To jmax
For i = 1 To imax
If HZ(i, j) > HZo Then
If H(i, j) < HZo Then
H(i, j) = 0.9 * HZo
End If
If dt > (1 / ((U(i, j) + (g * H(i, j)) ^ 0.5) / x + (V(i, j) + (g * H(i, j)) ^ 0.5) / y _
+ 2 * vt(i, j) / x ^ 2 + 2 * vtn / y ^ 2)) And dt > 0 Then
FLAG = 1
End If
DTCHECK = 0.5 * 1 / ((U(i, j) + (g * H(i, j)) ^ 0.5) / x + (V(i, j) + (g * H(i, j)) ^ 0.5) / y + 2 * vt(i, j) /
x ^ 2 + 2 * vt(i, j) / y ^ 2)
If DTCHECK < DTMIN And DTCHECK > 0 Then
DTMIN = DTCHECK
End If
End If
Next i
Next j

For j = 2 To jmax - 1
For i = 2 To imax - 1
H(i, j) = 0.25 * (H(i, j + 1) + H(i, j - 1) + H(i + 1, j) + H(i - 1, j))
Next i
Next j

If FLAG = 0 Then
For j = 2 To jmax - 1
For i = 2 To imax - 1

Heights.Cells(i, j).Value = H(i, j)
If H(i, j) > Z(i, j) + HZo And Um(i, j) + Vm(i, j) <> 0 And U(i, j) + V(i, j) <> 0 And ip = 1 Then
If R(i, j) > Z(i, j) Then
ks(i, j) = 2 * ds
Else
```

```
AggRoughness
End If
End If

Roughness.Cells(i, j).Value = ks(i, j)
Velocityx.Cells(i, j).Value = U(i, j)
Velocityy.Cells(i, j).Value = V(i, j)

Inputs.Cells(11).Value = dt
Inputs.Cells(12).Value = t

Next i
Next j


dt = DTMIN
Else
dt = 0
End If

Loop

If ip = iprint Then

CRSC1 = Worksheets("Display results").Range("i19").Value
CRSC2 = Worksheets("Display results").Range("k19").Value
If CRSC >= CRSC1 And CRSC <= CRSC2 Then

ok = False
Do Until ok

If ip = iprint + 1 Then
ip = 0
End If
ip = ip + 1


Application.Wait (True)
t = t + dt

Loop

End Sub


Sub ReedRoughness()

Set Reeds = Worksheets("2d Veg").Range("f4:y23")
imax = Reeds.Rows.Count
jmax = Reeds.Columns.Count
Set BeginTime = Reeds.Offset(imax + 4, jmax + 10)

Vel = (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5
```

```
If Reeds.Cells(i, j).Interior.ColorIndex <> 2 And U(i, j) + V(i, j) <> 0 Then

'a = stem spacing
'Dsht =  Stem diameter
If Reeds.Cells(i, j).Interior.ColorIndex = 36 Then 'dark green-Dense
a = 0.05
ElseIf Reeds.Cells(i, j).Interior.ColorIndex = 6 Then 'medium dark green
a = 0.75
ElseIf Reeds.Cells(i, j).Interior.ColorIndex = 43 Then 'medium  green-HalfDense
a = 0.1
ElseIf Reeds.Cells(i, j).Interior.ColorIndex = 12 Then 'medium light green
a = 0.125
ElseIf Reeds.Cells(i, j).Interior.ColorIndex = 52 Then 'light green-Sparce
a = 0.15
End If

tb = BeginTime.Cells(i, j).Value

If tb <= 10 Then
alpha = 30
beta = -0.38
Dsht = 0.005
ElseIf tb <= 20 Then
alpha = 200
beta = -0.58
Dsht = 0.01
ElseIf tb <= 70 Then
alpha = 1000
beta = -0.8
Dsht = 0.02
ElseIf tb <= 300 Then
alpha = 210
beta = -0.58
Dsht = 0.02
ElseIf tb > 300 Then
alpha = 30
beta = -0.38
Dsht = 0.025
End If

Do

fo = 4
Do
fs = 1 / (-2 * WorksheetFunction.Log10(ks(i, j) / (3.71 * (H(i, j) - Z(i, j))) + 2.51 * 0.000001 / (Vel *
(H(i, j) - Z(i, j)) * fo ^ 0.5))) ^ 2
fo = fo + 0.25 * (fs - fo)
Loop Until (fs - fo) < 0.1 And (fs - fo) > -0.1

Vel = (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5

Sle = fs * Vel ^ 2 / (8 * g * (H(i, j) - Z(i, j)))

SheVel = Abs(Vel) / (8 / fs) ^ 0.5
```

```
Re = SheVel * ds / 0.0000013

cdrag = alpha * Re ^ beta

FV = 1.885 * (a / Dsht) ^ (-0.65) * (Dsht / (H(i, j) - Z(i, j))) ^ 0.07 * cdrag ^ 0.48

Vel = 1 / FV * Sle

Velo = Velo + 0.5 * (Vel - Velo)

Loop Until (Vel - Velo) < 0.001 And (Vel - Velo) > -0.001

If U(i, j) + V(i, j) <> 0 Then
U(i, j) = U(i, j) * Velo / (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5
End If
If U(i, j) + V(i, j) <> 0 Then
V(i, j) = V(i, j) * Velo / (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5
End If

End If

End Sub

Sub AggRoughness()

Set Bed = Worksheets("3d Sed").Range("i2")
Set NewCell = Worksheets("3d Wat").Range("b3")
Set GrainSize = Worksheets("3d Sed").Range("b12")

NewCell.Cells(9).Value = i
NewCell.Cells(10).Value = j
Interpolator
GrainSize.Value = ds

If U(i, j) + V(i, j) < 0.01 And U(i, j) + V(i, j) > -0.01 And U(i, j) + V(i, j) <> 0 Then
Theta(i, j) = WorksheetFunction.Acos(U(i, j) / (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5)
ElseIf U(i, j) + V(i, j) <> 0 Then
Theta(i, j) = WorksheetFunction.Asin(V(i, j) / (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5)
Else
Theta(i, j) = 0
End If

Worksheets("3d Sed").Range("b5").Value = Theta(i, j)
t1 = Worksheets("1d Sed").Range("c18").Value
t2 = Worksheets("2d Sed").Range("c10").Value

Worksheets("3d Sed").Range("b8").Value = t1 * Worksheets("2d Sed").Range("c11").Value * 60 + t2
* 60
Onlyks

sumbedforms = 0
sumdarcy = 0
jjmax = 10
For jj = 2 To jjmax - 1
```

```
HB(jj) = Bed.Cells(5, jj).Value
L(jj) = Bed.Cells(6, jj).Value
f(jj) = Bed.Cells(7, jj).Value
sumdarcy = sumdarcy + f(jj) * rho * (Um(i, j) ^ 2 + Vm(i, j) ^ 2) / 8
sumbedforms = sumbedforms + 1 / 8 * rho * (Um(i, j) ^ 2 + Vm(i, j) ^ 2) * HB(jj) ^ 2 / (L(jj) * (HZ(i,
j)))
Next jj

ts = sumdarcy / jjmax
tbf = sumbedforms / jjmax
If ts + tbf = 0 Then
ks(i, j) = 2 * ds
Else
csn = (rho * (Um(i, j) ^ 2 + Vm(i, j) ^ 2) / (ts + tbf)) ^ 0.5
ks(i, j) = 12 * HZ(i, j) / 10 ^ (csn / 5.75)
End If
End Sub
```

```
Sub TwoDimSediment ()

Set Topography = Worksheets("2d Sed").Range("f3:y22")
imax = Topography.Rows.Count
jmax = Topography.Columns.Count

Set Heights = Topography.Offset(imax + 4, 0)
Set Velocityx = Heights.Offset(imax + 4, 0)
Set Velocityy = Velocityx.Offset(imax + 4, 0)
Set ConcreteRock = Topography.Offset(0, jmax + 8)
Set Transport = ConcreteRock.Offset(imax + 4, 0)
Set GrainSize1 = Transport.Offset(imax + 4, 0)

Set Inputs = Worksheets("2d Sed").Range("c3")
Set Elevation = Worksheets("1d Sed").Range("AJ7")
Set XYZ = Worksheets("2d Sed").Range("bj3")
Set CountCoordinates = Worksheets("Display results").Range("ar26")

Set VegTopography = Worksheets("2d Veg").Range("f4:y23")

Set WaterTopography = Worksheets("2d Wat").Range("i3:ab22")
Set WaterHeights = WaterTopography.Offset(imax + 4, 0)
Set WaterVelocityx = WaterHeights.Offset(0, jmax + 4)
Set WaterVelocityy = WaterVelocityx.Offset(0, jmax + 4)

Qs1 = Inputs.Cells(2).Value
Qs2 = Inputs.Cells(3).Value

dti = Inputs.Cells(6).Value * 60

dte = Inputs.Cells(7).Value * 60

t = Inputs.Cells(8).Value * 60
tLast = Inputs.Cells(9).Value * 60

x = Inputs.Cells(4).Value
y = Inputs.Cells(5).Value

k1 = Inputs.Cells(34).Value
k2 = Inputs.Cells(35).Value
k3 = Inputs.Cells(36).Value

no = Inputs.Cells(11).Value

InputFlowGrainSize

ip = 1
n = 0

iprint = Inputs.Cells(41).Value

Ntoprint = Inputs.Cells(43).Value
ok = False
```

```
Velocityx.ClearContents
Velocityy.ClearContents
t = 0
Do While t <= tLast

'Model cross over
If ip = 1 Then
If t = 0 Then
Worksheets("2d Wat").Range("b12") = 0
Else
Worksheets("2d Wat").Range("b12") = 0.001
End If
Worksheets("2d Wat").Range("e3") = CRSC
TwoDimWaterFlow
Heights.Value = WaterHeights.Value
Velocityx.Value = WaterVelocityx.Value
Velocityy.Value = WaterVelocityy.Value
End If

For j = 1 To jmax
For i = 1 To imax
Z(i, j) = Topography.Cells(i, j).Value
R(i, j) = ConcreteRock.Cells(i, j).Value
H(i, j) = Heights.Cells(i, j).Value
U(i, j) = Velocityx.Cells(i, j).Value
V(i, j) = Velocityy.Cells(i, j).Value
ds(i, j, 1) = GrainSize1.Cells(i, j).Value
Transport.Cells(i, j).Value = Outflow(i, j)
Next i
Next j

InCells = 0
OutCells = 0
For i = 1 To imax
If H(i, 2) - Z(i, 2) > 0 Then
InCells = InCells + 1
End If
If H(i, jmax) - Z(i, jmax) > 0 Then
OutCells = OutCells + 1
End If
Next i

For i = 1 To imax
If H(i, 2) - Z(i, 2) > 0 Then
Inflow(i, 2) = Qs1 * dte / (y * InCells) + Outflows / InCells - Qs2 * dte / (y * OutCells)
End If
Next i

Transport.ClearContents

'Equilibruim slope
For j = 2 To jmax - 1
For i = 2 To imax - 1
```

```
Outflow(i, j) = 0

If H(i, j) > Z(i, j) Then

If U(i, j) + V(i, j) < 0.01 And U(i, j) + V(i, j) > -0.01 And U(i, j) + V(i, j) <> 0 Then
Theta(i, j) = WorksheetFunction.Acos(U(i, j) / (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5)
ElseIf U(i, j) + V(i, j) <> 0 Then
Theta(i, j) = WorksheetFunction.Asin(V(i, j) / (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5)
Else
Theta(i, j) = 0
End If

Velu(i, j) = (U(i, j) ^ 2 + V(i, j) ^ 2) ^ 0.5

'Height differences between the cell and downstream cells
'1
If Theta(i, j) > -0.464 And Theta(i, j) < 0.464 Then
Difa = Z(i, j) - Z(i - 1, j + 1)
Difb = (2 ^ 0.5) * (Z(i, j) - Z(i, j + 1))
Difc = Z(i, j) - Z(i + 1, j + 1)
L(i, j) = x
b(i, j) = y
Veld = (U(i, j + 1) ^ 2 + V(i, j + 1) ^ 2) ^ 0.5
zd = Z(i, j + 1)
wd = H(i, j + 1)
'where sle is the energy slope, H(i, j) and H(i, j + 1) are the high water marks, L(i, j) is the distance between the two sections
'2
ElseIf Theta(i, j) > -1.107 And Theta(i, j) < -0.464 Then
Difa = (2 ^ 0.5) * (Z(i, j) - Z(i, j + 1))
Difb = Z(i, j) - Z(i + 1, j + 1)
Difc = (2 ^ 0.5) * (Z(i, j) - Z(i + 1, j))
L(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
b(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
Veld = (U(i + 1, j + 1) ^ 2 + V(i + 1, j + 1) ^ 2) ^ 0.5
zd = Z(i + 1, j + 1)
wd = H(i + 1, j + 1)
'3
ElseIf Theta(i, j) > -2.034 And Theta(i, j) < -1.107 Then
Difa = Z(i, j) - Z(i + 1, j + 1)
Difb = (2 ^ 0.5) * (Z(i, j) - Z(i + 1, j))
Difc = Z(i, j) - Z(i + 1, j - 1)
L(i, j) = y
b(i, j) = x
Veld = (U(i + 1, j) ^ 2 + V(i + 1, j) ^ 2) ^ 0.5
zd = Z(i + 1, j)
wd = H(i + 1, j)
'4
ElseIf Theta(i, j) > -2.678 And Theta(i, j) < -2.034 Then
Difa = (2 ^ 0.5) * (Z(i, j) - Z(i + 1, j))
Difb = Z(i, j) - Z(i + 1, j - 1)
Difc = (2 ^ 0.5) * (Z(i, j) - Z(i, j - 1))
L(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
b(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
```

```
Veld = (U(i + 1, j - 1) ^ 2 + V(i + 1, j - 1) ^ 2) ^ 0.5
zd = Z(i + 1, j - 1)
wd = H(i + 1, j - 1)
'5
ElseIf Theta(i, j) > 2.678 Or Theta(i, j) < -2.678 Then
Difa = (2 ^ 0.5) * (Z(i, j) - Z(i + 1, j - 1))
Difb = Z(i, j) - Z(i, j - 1)
Difc = (2 ^ 0.5) * (Z(i, j) - Z(i - 1, j - 1))
L(i, j) = x
b(i, j) = y
Veld = (U(i + 1, j + 1) ^ 2 + V(i + 1, j + 1) ^ 2) ^ 0.5
zd = Z(i + 1, j + 1)
wd = H(i + 1, j + 1)
'6
ElseIf Theta(i, j) > 2.034 And Theta(i, j) < 2.678 Then
Difa = (2 ^ 0.5) * (Z(i, j) - Z(i, j - 1))
Difb = Z(i, j) - Z(i - 1, j - 1)
Difc = (2 ^ 0.5) * (Z(i, j) - Z(i - 1, j))
L(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
b(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
Veld = (U(i - 1, j - 1) ^ 2 + V(i - 1, j - 1) ^ 2) ^ 0.5
zd = Z(i - 1, j - 1)
wd = H(i - 1, j - 1)
'7
ElseIf Theta(i, j) > 1.107 And Theta(i, j) < 2.034 Then
Difa = Z(i, j) - Z(i - 1, j - 1)
Difb = (2 ^ 0.5) * (Z(i, j) - Z(i - 1, j))
Difc = Z(i, j) - Z(i - 1, j + 1)
L(i, j) = y
b(i, j) = x
Veld = (U(i - 1, j) ^ 2 + V(i - 1, j) ^ 2) ^ 0.5
zd = Z(i - 1, j)
wd = H(i - 1, j)
'8
ElseIf Theta(i, j) > 0.464 And Theta(i, j) < 1.107 Then
Difa = (2 ^ 0.5) * (Z(i, j) - Z(i - 1, j))
Difb = Z(i, j) - Z(i - 1, j + 1)
Difc = (2 ^ 0.5) * (Z(i, j) - Z(i, j + 1))
L(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
b(i, j) = (x ^ 2 + y ^ 2) ^ 0.5
Veld = (U(i - 1, j + 1) ^ 2 + V(i - 1, j + 1) ^ 2) ^ 0.5
zd = Z(i - 1, j + 1)
wd = H(i - 1, j + 1)
End If

End If

Next i
Next j

TwoDimTransportVolume

For j = 2 To jmax - 1
For i = 2 To imax - 1
```

```
If H(i, j) - Z(i, j) > 2 * Worksheets("2d Wat").Range("b15").Value And Velu(i, j) > 0 Then

cici(i, j) = 1
If VegTopography.Cells(i, j).Interior.ColorIndex <> Inputs.Cells(26).Offset(0, -1).Interior.ColorIndex
Then
'Sparse
If VegTopography.Cells(i, j).Interior.ColorIndex = Inputs.Cells(29).Offset(0, -1).Interior.ColorIndex
Then
cici(i, j) = Inputs.Cells(29).Value
'More dense
ElseIf VegTopography.Cells(i, j).Interior.ColorIndex = Inputs.Cells(30).Offset(0, -
1).Interior.ColorIndex Then
cici(i, j) = Inputs.Cells(30).Value
'Dense
ElseIf VegTopography.Cells(i, j).Interior.ColorIndex = Inputs.Cells(31).Offset(0, -
1).Interior.ColorIndex Then
cici(i, j) = Inputs.Cells(31).Value
'Very dense
ElseIf VegTopography.Cells(i, j).Interior.ColorIndex = Inputs.Cells(32).Offset(0, -
1).Interior.ColorIndex Then
cici(i, j) = Inputs.Cells(32).Value
'ElseIf ConcreteRock.Cells(i, j).Interior.ColorIndex = Inputs.Cells(27).Offset(0, -
2).Interior.ColorIndex Then
'ElseIf ConcreteRock.Cells(i, j).Interior.ColorIndex = Inputs.Cells(28).Offset(0, -
2).Interior.ColorIndex Then

End If
End If

Storage(i, j) = cici(i, j) * S(i, j)
If Storage(i, j) > 1 Then
Storage(i, j) = 1
End If

'Any uphills?
If Difa < 0 Or Difb < 0 Or Difc < 0 Then
If Difa <= Difb And Difa <= Difc Then
 Dif1 = 1
 Dif2 = Difb - Difa + 1
 Dif3 = Difc - Difa + 1

ElseIf Difb <= Difa And Difb <= Difc Then
 Dif2 = 1
 Dif1 = Difa - Difb + 1
 Dif3 = Difc - Difb + 1

ElseIf Difc <= Difb And Difc <= Difa Then
 Dif3 = 1
 Dif1 = Difa - Difc + 1
 Dif2 = Difb - Difc + 1

End If
Else: Dif1 = Difa
```

```
Dif2 = Difb
Dif3 = Difc
End If

Total_Dif = Dif1 + Dif2 + Dif3

If Total_Dif = 0 Then
Dif1 = 1
Dif2 = 1
Dif3 = 1
Total_Dif = 3
End If

'O1
O1 = Storage(i, j) * Dif1 / Total_Dif
'O2
O2 = Storage(i, j) * Dif2 / Total_Dif
'O3
O3 = Storage(i, j) * Dif3 / Total_Dif

'Cant go lower than rigid member i.e. floor or obstacle
If Z(i, j) <= R(i, j) Then
O1 = (Z(i, j) - R(i, j) + Inflow(i, j)) * Dif1 / Total_Dif
O2 = (Z(i, j) - R(i, j) + Inflow(i, j)) * Dif2 / Total_Dif
O3 = (Z(i, j) - R(i, j) + Inflow(i, j)) * Dif3 / Total_Dif
Z(i, j) = R(i, j)
ConcreteRock.Cells(i, j).Interior.ColorIndex = Inputs.Cells(28).Offset(0, -1).Interior.ColorIndex
Else
ConcreteRock.Cells(i, j).Interior.ColorIndex = Inputs.Cells(26).Offset(0, -1).Interior.ColorIndex
End If

Z(i, j) = Z(i, j) + Inflow(i, j) - (O1 + O2 + O3)

'1
If Theta(i, j) > -0.464 And Theta(i, j) < 0.464 Then
Outflow(i - 1, j + 1) = Outflow(i - 1, j + 1) + O1
Outflow(i, j + 1) = Outflow(i, j + 1) + O2
Outflow(i + 1, j + 1) = Outflow(i + 1, j + 1) + O3
'2
ElseIf Theta(i, j) > -1.107 And Theta(i, j) < -0.464 Then
Outflow(i, j + 1) = Outflow(i, j + 1) + O1
Outflow(i + 1, j + 1) = Outflow(i + 1, j + 1) + O2
Outflow(i + 1, j) = Outflow(i + 1, j) + O3
'3
ElseIf Theta(i, j) > -2.034 And Theta(i, j) < -1.107 Then
Outflow(i + 1, j + 1) = Outflow(i + 1, j + 1) + O1
Outflow(i + 1, j) = Outflow(i + 1, j) + O2
Outflow(i + 1, j - 1) = Outflow(i + 1, j - 1) + O3
'4
ElseIf Theta(i, j) > -2.678 And Theta(i, j) < -2.034 Then
Outflow(i + 1, j) = Outflow(i + 1, j) + O1
Outflow(i + 1, j - 1) = Outflow(i + 1, j - 1) + O2
Outflow(i, j - 1) = Outflow(i, j - 1) + O3
'5
```

```
ElseIf Theta(i, j) > 2.678 Or Theta(i, j) < -2.678 Then
Outflow(i + 1, j - 1) = Outflow(i + 1, j - 1) + O1
Outflow(i, j - 1) = Outflow(i, j - 1) + O2
Outflow(i - 1, j - 1) = Outflow(i - 1, j - 1) + O3
'6
ElseIf Theta(i, j) > 2.034 And Theta(i, j) < 2.678 Then
Outflow(i, j - 1) = Outflow(i, j - 1) + O1
Outflow(i - 1, j - 1) = Outflow(i - 1, j - 1) + O2
Outflow(i - 1, j) = Outflow(i - 1, j) + O3
'7
ElseIf Theta(i, j) > 1.107 And Theta(i, j) < 2.034 Then
Outflow(i - 1, j - 1) = Outflow(i - 1, j - 1) + O1
Outflow(i - 1, j) = Outflow(i - 1, j) + O2
Outflow(i - 1, j + 1) = Outflow(i - 1, j + 1) + O3
'8
ElseIf Theta(i, j) > 0.464 And Theta(i, j) < 1.107 Then
Outflow(i - 1, j) = Outflow(i - 1, j) + O1
Outflow(i - 1, j + 1) = Outflow(i - 1, j + 1) + O2
Outflow(i, j + 1) = Outflow(i, j + 1) + O3
End If

End If

Next i
Next j

B2 = Inputs.Cells(19).Value
sdsl2 = Inputs.Cells(21).Value
Z(1, jmax) = Z(imax / 2, jmax) + ((imax - 2) * y / 2 - B2 / 2) / sdsl2
Z(1, 1) = Z(1, 3)

For i = 2 To imax

If H(i, 2) - Z(i, 2) > 0 Then
If i <> imax / 2 Then
Z(i, 2) = Z(i - 1, 2) - y / sdsl2
End If
If i > imax / 2 - (B2 / 2) / y Then
Z(i, 2) = Z(imax / 2, 2)
End If
If i > imax / 2 + (B2 / 2) / y Then
Z(i, 2) = Z(i - 1, 2) + y / sdsl2
End If
End If

Z(i, 1) = Z(i, 2)

If H(i, jmax) - Z(i, jmax) > 0 Then
If i <> imax / 2 Then
Z(i, jmax) = Z(i - 1, jmax) - y / sdsl2
End If
If i > imax / 2 - (B2 / 2) / y Then
Z(i, jmax) = Z(imax / 2, jmax)
End If
```

```
If i > imax / 2 + (B2 / 2) / y Then
Z(i, jmax) = Z(i - 1, jmax) + y / sdsl2
End If
End If

Z(i, jmax - 1) = Z(i, jmax)

Next i

If Worksheets("2d Sed").Range("c47") = True Then
For j = 2 To jmax - 1
For i = 2 To imax - 1
If VegTopography.Cells(i, j).Interior.ColorIndex = Inputs.Cells(26).Offset(0, -1).Interior.ColorIndex
Then
Avalances
End If
Next i
Next j
End If

For j = 1 To jmax
For i = 1 To imax
Topography.Cells(i, j).Value = Z(i, j)
Transport.Cells(i, j).Value = Outflow(i, j)
Inflow(i, j) = Outflow(i, j)
GrainSize1.Cells(i, j).Value = ds(i, j, 1)
Next i
Next j

Outflows = 0
For i = 1 To imax
If H(i, 19) - Z(i, 19) > 0 Then
If Outflow(i, 19) < 0 Then
Outflow(i, 19) = 0
End If
Outflows = Outflows + Outflow(i, 19)
End If
Next i

t1 = Worksheets("1d Sed").Range("c18").Value
If t = 0 Then
pr = (t1 - 1) * Ntoprint
End If

End If

If ip = iprint + 1 Then
ip = 0
Inputs.Cells(39).Value = pr
pr = pr + 1
End If
ip = ip + 1

Inputs.Cells(8).Value = t / 60
```

```
t = t + dte
Application.Wait (True)
Loop

End Sub

Sub InputFlowGrainSize()

Set Inputs = Worksheets("2d Sed").Range("c3")
Set Elevation = Worksheets("1d Sed").Range("AJ7")
Set GrainSize = Elevation.Offset(0, 8)

CRSC = Inputs.Cells(6, 1).Value

Inputs.Cells(14).Value = GrainSize.Cells(CRSC, 4)
Inputs.Cells(15).Value = GrainSize.Cells(CRSC + 1, 4)

Inputs.Cells(12).Value = Elevation.Cells(CRSC, 1)
Inputs.Cells(13).Value = Elevation.Cells(CRSC + 1, 1)

x1 = Inputs.Cells(12).Value
x2 = Inputs.Cells(13).Value
ds1 = Inputs.Cells(14).Value / 1000
ds2 = Inputs.Cells(15).Value / 1000

For j = 1 To jmax
For i = 1 To imax
ds(i, j, 1) = ds1 - j * (ds1 - ds2) / jmax
GrainSize1.Cells(i, j).Value = ds(i, j, 1)
Next i
Next j

End Sub

Sub TwoDimTransportVolume()

Set Roughness = Worksheets("2d Wat").Range("ag3:az22")

ps = 0.35

For j = 2 To jmax - 1
For i = 2 To imax - 1

zui = Z(i, j)
zu = zui

If H(i, j) - Z(i, j) > 2 * Worksheets("2d Wat").Range("b15").Value And Velu(i, j) > 0 Then

If Velu(i, j) > 0 Then
fo = 1
Do
ks(i, j) = 2 * ds(i, j, 1)
fs(i, j) = 1 / (-2 * WorksheetFunction.Log10(ks(i, j) / (3.71 * (H(i, j) - zu)) + 2.51 * vs / (Velu(i, j) *
(H(i, j) - zu) * fo ^ 0.5))) ^ 2
```

```
fo = fo + 0.1 * (fs(i, j) - fo)
Loop Until (fs(i, j) - fo) < 0.1 And (fs(i, j) - fo) > -0.1
Sle(i, j) = fs(i, j) * Velu(i, j) ^ 2 / (8 * g * (H(i, j) - zu))
Else
Sle(i, j) = 0
End If

Rs = (ds(i, j, 1) * (ss * g / vs ^ 2) ^ (1 / 3)) ^ (3 / 2)
If Rs < 6.61 Then
Shields = 0.1414 * Rs ^ -0.2306
ElseIf Rs > 282.84 Then
Shields = 0.045
Else
Shields = (1 + (0.0223 * Rs) ^ 2.8358) ^ 0.3542 / (3.0946 * Rs ^ 0.6769)
End If

If Sle(i, j) = 0 Then
ib = 0
Else
W = rho * g * (H(i, j) - zu) * Sle(i, j) * Velu(i, j)
wo = 5.75 * (Shields * (ss - 1) * rho) ^ (3 / 2) * (g / rho) ^ 0.5 * ds(i, j, 1) ^ (3 / 2) *
WorksheetFunction.Log10(12 * (H(i, j) - zu) / ds(i, j, 1))
End If

If W < wo Then
zu = zui
Else
ib = ss / (ss - 1) * 0.1 * ((W - wo) / 0.5) ^ (3 / 2) * ((H(i, j) - zu) / 0.1) ^ (-2 / 3) * (ds(i, j, 1) / 0.0011) ^
(-0.5)

qb = ib / (rho * ss * (1 - ps))  'ib is dry weight

zu = zui - qb * dte / b(i, j)
End If

S(i, j) = (zui - zu)
Else
S(i, j) = 0
End If

Next i
Next j

End Sub

Sub Template()

Dim Z(101, 101) As Double

Set Inputs = Worksheets("2d Sed").Range("c3")

Set Topography = Worksheets("2d Sed").Range("f3:y22")
imax = Topography.Rows.Count
jmax = Topography.Columns.Count
```

```
Set ConcreteRock = Topography.Offset(0, jmax + 8)

CRSC = Inputs.Cells(6, 1).Value
Set Outcrop = ConcreteRock.Offset((CRSC + 2) * (imax + 4), 0)

Set Characteristics = Worksheets("1d Sed").Range("n5")

Set Elevation = Worksheets("1d Sed").Range("AJ7")
Set GrainSize = Elevation.Offset(0, 8)
Set Load = GrainSize.Offset(0, 8)

Set SedimentInputs = Worksheets("1d Sed").Range("c6")

Set PeakFlow = Worksheets("1d Wat").Range("x7")

Set XYZ = Worksheets("2d Sed").Range("bj3")

Set ChaCha = Worksheets("1d Sed").Range("n5")

sli = (Elevation.Cells(CRSC, 4).Value - Elevation.Cells(CRSC + 1, 4).Value) / (Elevation.Cells(CRSC
+ 1, 1).Value - Elevation.Cells(CRSC, 1).Value) + 0.0000000001

Inputs.Cells(20).Value = Characteristics.Cells(CRSC, 1).Value
Inputs.Cells(21).Value = Characteristics.Cells(CRSC + 1, 1).Value

Inputs.Cells(18).Value = Characteristics.Cells(CRSC, 2).Value
Inputs.Cells(19).Value = Characteristics.Cells(CRSC + 1, 2).Value

Inputs.Cells(1).Value = sli
Ntoprint = Inputs.Cells(43).Value
B1 = Inputs.Cells(18).Value
B2 = Inputs.Cells(19).Value

x = Inputs.Cells(4).Value
y = Inputs.Cells(5).Value

sdsl1 = Inputs.Cells(20).Value
sdsl2 = Inputs.Cells(21).Value

dte = Inputs.Cells(7).Value
qbTf = SedimentInputs.Cells(2).Value
Inputs.Cells(2).Value = Load.Cells(CRSC, 4) * qbTf / SedimentInputs.Cells(3).Value
Inputs.Cells(3).Value = Load.Cells(CRSC + 1, 4) * qbTf / SedimentInputs.Cells(3).Value

If Worksheets("1d Sed").Range("c19").Value = True Then

R(1, 1) = (imax / 2 * x - B1) / sdsl1 + ChaCha.Cells(CRSC, 4).Value
R(imax, 1) = R(1, 1)

For i = 2 To imax

R(i, 1) = R(i - 1, 1) - y / sdsl1
If i >= imax / 2 - (B1 / 2) / y Then
R(i, 1) = R(i - 1, 1)
```

```
End If
If i > imax / 2 + (B1 / 2) / y Then
R(i, 1) = R(i - 1, 1) + y / sdsl1
End If
Next i

slic = (ChaCha.Cells(CRSC, 4).Value - ChaCha.Cells(CRSC + 1, 4).Value) / (Elevation.Cells(CRSC +
1, 1).Value - Elevation.Cells(CRSC, 1).Value) + 0.0000000001

For j = 2 To jmax

R(imax / 2, j) = R(imax / 2, j - 1) - slic * x

Next j

R(1, jmax) = R(imax / 2, jmax) + ((imax - 2) * y / 2 - B2 / 2) / sdsl2

For i = 2 To imax
If i <> imax / 2 Then
R(i, jmax) = R(i - 1, jmax) - y / sdsl2
End If
If i > imax / 2 - (B2 / 2) / y Then
R(i, jmax) = R(imax / 2, jmax)
End If
If i > imax / 2 + (B2 / 2) / y Then
R(i, jmax) = R(i - 1, jmax) + y / sdsl2
End If
Next i

For j = 1 To jmax
For i = 1 To imax
If i <> imax / 2 Then
R(i, j) = R(i, 1) - (R(i, 1) - R(i, jmax)) * j / jmax
End If
R(i, j) = R(i, j) + Outcrop.Cells(i, j).Value
ConcreteRock.Cells(i, j).Value = R(i, j)
Next i
Next j
Else
ConcreteRock.ClearContents
End If

Z(1, 1) = (imax / 2 * x - B1) / sdsl1 + Elevation.Cells(CRSC, 4).Value
Z(imax, 1) = Z(1, 1)

For i = 2 To imax

Z(i, 1) = Z(i - 1, 1) - y / sdsl1
If i >= imax / 2 - (B1 / 2) / y Then
Z(i, 1) = Z(i - 1, 1)
End If
If i > imax / 2 + (B1 / 2) / y Then
Z(i, 1) = Z(i - 1, 1) + y / sdsl1
End If
```

```
Next i

For j = 2 To jmax

Z(imax / 2, j) = Z(imax / 2, j - 1) - sli * x

Next j

Z(1, jmax) = Z(imax / 2, jmax) + ((imax - 2) * y / 2 - B2 / 2) / sdsl2

For i = 2 To imax
If i <> imax / 2 Then
Z(i, jmax) = Z(i - 1, jmax) - y / sdsl2
End If
If i > imax / 2 - (B2 / 2) / y Then
Z(i, jmax) = Z(imax / 2, jmax)
End If
If i > imax / 2 + (B2 / 2) / y Then
Z(i, jmax) = Z(i - 1, jmax) + y / sdsl2
End If
Next i

For j = 1 To jmax
For i = 1 To imax
If i <> imax / 2 Then
Z(i, j) = Z(i, 1) - (Z(i, 1) - Z(i, jmax)) * j / jmax
End If

If Z(i, j) <= R(i, j) Then
Z(i, j) = R(i, j)
ConcreteRock.Cells(i, j).Interior.ColorIndex = Inputs.Cells(28).Offset(0, -1).Interior.ColorIndex
Else
ConcreteRock.Cells(i, j).Interior.ColorIndex = Inputs.Cells(26).Offset(0, -1).Interior.ColorIndex
End If

Topography.Cells(i, j).Value = Z(i, j)
Next i
Next j

'Model cross over 2d Veg
Worksheets("2d Veg").Range("b11") = 0
Worksheets("2d Veg").Range("b12") = 180
Worksheets("2d Veg").Range("b5") = CRSC
TwoDimVeg
Inputs.Cells(8).Value = 0
TwoDimSedimentRoute

'Model cross over 2d Veg
Worksheets("2d Veg").Range("b12") = 360
TwoDimVeg

End Sub

Sub Avalances()
```

```
If ds(i, j, 1) < 0.0001 Then
Latact = x * Tan(0.523598776)
ElseIf ds(i, j, 1) < 0.001 Then
Latact = x * Tan(0.558505361)
ElseIf ds(i, j, 1) < 0.01 Then
Latact = x * Tan(0.610865238)
ElseIf ds(i, j, 1) < 0.05 Then
Latact = x * Tan(0.645771823)
ElseIf ds(i, j, 1) >= 0.1 Then
Latact = x * Tan(0.698131701)
End If

Lat1 = Z(i - 1, j + 1) - Z(i, j)
Lat2 = Z(i, j + 1) - Z(i, j)
Lat3 = Z(i + 1, j + 1) - Z(i, j)
Lat4 = Z(i + 1, j) - Z(i, j)
Lat5 = Z(i + 1, j - 1) - Z(i, j)
Lat6 = Z(i, j - 1) - Z(i, j)
Lat7 = Z(i - 1, j - 1) - Z(i, j)
Lat8 = Z(i - 1, j) - Z(i, j)

If Lat1 <= Latact Then
Lat1 = 0
End If
If Lat2 <= Latact Then
Lat2 = 0
End If
If Lat3 <= Latact Then
Lat3 = 0
End If
If Lat4 <= Latact Then
Lat4 = 0
End If
If Lat5 <= Latact Then
Lat5 = 0
End If
If Lat6 <= Latact Then
Lat6 = 0
End If
If Lat7 <= Latact Then
Lat7 = 0
End If
If Lat8 <= Latact Then
Lat8 = 0
End If

Lat_total = Lat1 + Lat2 + Lat3 + Lat4 + Lat5 + Lat6 + Lat7 + Lat8

If Lat_total > 0 Then

Out1 = 0.5 * (Lat1 - Latact) * Lat1 / Lat_total
Out2 = 0.5 * (Lat2 - Latact) * Lat2 / Lat_total
Out3 = 0.5 * (Lat3 - Latact) * Lat3 / Lat_total
```

```
Out4 = 0.5 * (Lat4 - Latact) * Lat4 / Lat_total
Out5 = 0.5 * (Lat5 - Latact) * Lat5 / Lat_total
Out6 = 0.5 * (Lat6 - Latact) * Lat6 / Lat_total
Out7 = 0.5 * (Lat7 - Latact) * Lat7 / Lat_total
Out8 = 0.5 * (Lat8 - Latact) * Lat8 / Lat_total

Else
Out1 = 0
Out2 = 0
Out3 = 0
Out4 = 0
Out5 = 0
Out6 = 0
Out7 = 0
Out8 = 0
End If

'Cant go lower than rigid member i.e. floor or obstacle
If Z(i, j) + (Out1 + Out2 + Out3 + Out4 + Out5 + Out6 + Out7 + Out8) <= R(i, j) Then
Out1 = 0
Out2 = 0
Out3 = 0
Out4 = 0
Out5 = 0
Out6 = 0
Out7 = 0
Out8 = 0
End If

Z(i - 1, j + 1) = Z(i - 1, j + 1) - Out1
Z(i, j + 1) = Z(i, j + 1) - Out2
Z(i + 1, j + 1) = Z(i + 1, j + 1) - Out3
Z(i + 1, j) = Z(i + 1, j) - Out4
Z(i + 1, j - 1) = Z(i + 1, j - 1) - Out5
Z(i - 1, j) = Z(i - 1, j) - Out6
Z(i - 1, j - 1) = Z(i - 1, j - 1) - Out7
Z(i - 1, j) = Z(i - 1, j) - Out8

Z(i, j) = Z(i, j) + (Out1 + Out2 + Out3 + Out4 + Out5 + Out6 + Out7 + Out8)

End Sub
```

```
Sub TwoDimVeg()

Set Topography = Worksheets("2d Veg").Range("f4:y23")
imax = Topography.Rows.Count
jmax = Topography.Columns.Count
Set ActualBiomass = Topography.Offset(imax + 4, 0)
Set Rhizomes = ActualBiomass.Offset(imax + 4, 0)
Set BeginTime = ActualBiomass.Offset(0, jmax + 10)
Set ShootLengths = Rhizomes.Offset(0, jmax + 10)
Set NewRhizomes = Rhizomes.Offset(imax + 4, 0)
Set Expansion = NewRhizomes.Offset(0, jmax + 10)
Set Shoots = NewRhizomes.Offset(imax + 4, 0)
Set Roots = Shoots.Offset(imax + 4, 0)
Set Panicles = Roots.Offset(imax + 4, 0)
Set SedimentTopography = Worksheets("2d Sed").Range("f3:y22")
Set WaterHeights = Worksheets("2d Wat").Range("i3:ab22").Offset(imax + 4, 0)

Set Inputs = Worksheets("2d Veg").Range("b3")
Set Percent = Worksheets("1d Veg").Range("y4")
Set CrossSec = Worksheets("1d Veg").Range("ap2")
Set Colour = Worksheets("2d Veg").Range("ad5")
Set Variables = Worksheets("3d Veg").Range("b31")
Set Elevation = Worksheets("1d Sed").Range("AJ7")
Set XYZ = Worksheets("2d Veg").Range("bi6")
Set CountCoordinates = Worksheets("Display results").Range("ar46")

CRSC = Inputs.Cells(3).Value
Set InitColours = Shoots.Offset((CRSC - 1) * (imax + 4), jmax + 10)

dt = Inputs.Cells(4).Value
t = Inputs.Cells(9).Value
tLast = Inputs.Cells(10).Value

ip = 1
n = 0

iprint = Inputs.Cells(12).Value
Ntoprint = Inputs.Cells(13).Value

t1 = Worksheets("1d Sed").Range("c18").Value

cut = False

If t = 0 Then

ActualBiomass.ClearContents
Rhizomes.ClearContents
BeginTime.ClearContents
ShootLengths.ClearContents
NewRhizomes.ClearContents
Expansion.ClearContents
Shoots.ClearContents
Roots.ClearContents
Panicles.ClearContents
```

```
For j = 1 To jmax
For i = 1 To imax

Topography.Cells(i, j).Interior.ColorIndex = InitColours.Cells(i, j).Interior.ColorIndex

If Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(1, 3).Value Then
Brhio(i, j) = Colour.Cells(1).Value * 1000
Rhizomes.Cells(i, j).Value = Brhio(i, j) / 1000
ElseIf Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(2, 3).Value Then
Brhio(i, j) = Colour.Cells(2).Value * 1000
Rhizomes.Cells(i, j).Value = Brhio(i, j) / 1000
Rhizomes.Cells(i, j).Value = Brhio(i, j) / 1000
ElseIf Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(3, 3).Value Then
Brhio(i, j) = Colour.Cells(3).Value * 1000
Rhizomes.Cells(i, j).Value = Brhio(i, j) / 1000
ElseIf Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(4, 3).Value Then
Brhio(i, j) = Colour.Cells(4).Value * 1000
Rhizomes.Cells(i, j).Value = Brhio(i, j) / 1000
ElseIf Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(5, 3).Value Then
Brhio(i, j) = Colour.Cells(5).Value * 1000
Rhizomes.Cells(i, j).Value = Brhio(i, j) / 1000
End If
Next i
Next j

End If

Topography.Value = SedimentTopography.Value
Bmax = Inputs.Cells(5).Value * 1000
BMin = Inputs.Cells(6).Value * 1000

For j = 1 To jmax
zmin = Elevation.Cells(CRSC, 4).Value - (Elevation.Cells(CRSC, 4).Value - Elevation.Cells(CRSC +
1, 4).Value) * j / (jmax - 1)

For i = 1 To imax
tb(i, j) = BeginTime.Cells(i, j).Value
B(i, j) = ActualBiomass.Cells(i, j).Value * 1000
Brhi(i, j) = Rhizomes.Cells(i, j).Value * 1000
Bn(i, j) = Roots.Cells(i, j).Value * 1000
Bsht(i, j) = NewRhizomes.Cells(i, j).Value * 1000
Brt(i, j) = Shoots.Cells(i, j).Value * 1000
Bp(i, j) = Panicles.Cells(i, j).Value * 1000
ShootHeight(i, j) = ShootLengths.Cells(i, j).Value

Z(i, j) = Topography.Cells(i, j).Value

H = 1
ok = False
Do
If Z(i, j) - zmin < CrossSec.Cells(1, 1).Value Then
BioFrac(i, j) = CrossSec.Cells(1, CRSC).Value / 100
ok = True
```

```
ElseIf Z(i, j) - zmin >= CrossSec.Cells(H, 1).Value And Z(i, j) - zmin < CrossSec.Cells(H + 1,
1).Value Then
BioFrac(i, j) = CrossSec.Cells(H, CRSC).Value / 100
ok = True
ElseIf CrossSec.Cells(H, 1) = "" Then
BioFrac(i, j) = CrossSec.Cells(H - 1, CRSC).Value / 100
ok = True
End If

H = H + 1
Loop Until ok

Expansion.Cells(i, j).Value = Bmax * BioFrac(i, j) / 1000

Next i
Next j

Do

i = Inputs.Cells(18).Value
j = Inputs.Cells(19).Value

If WaterHeights.Cells(i, j).Value > SedimentTopography.Cells(i, j).Value And BioFrac(i, j) > 0 Then
Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(1, 3).Value
End If

tfin = t + dt
Variables.Cells(3).Value = tfin

For j = 2 To jmax - 1
For i = 2 To imax - 1

If B(i, j) > BMin Then
Ind1 = 0
Ind2 = 0
Ind3 = 0
Ind4 = 0

If B(i, j + 1) < B(i, j) And B(i, j + 1) < Bmax * BioFrac(i, j + 1) / 1000 Then
Ind1 = 1
End If
If B(i + 1, j) < B(i, j) And B(i + 1, j) < Bmax * BioFrac(i + 1, j) / 1000 Then
Ind2 = 1
End If
If B(i, j - 1) < B(i, j) And B(i, j - 1) < Bmax * BioFrac(i, j - 1) / 1000 Then
Ind3 = 1
End If
If B(i - 1, j) < B(i, j) And B(i - 1, j) < Bmax * BioFrac(i - 1, j) / 1000 Then
Ind4 = 1
End If

If Ind1 = 1 Then
Brhi(i, j + 1) = Brhi(i, j + 1) + Ind1 * Bn(i, j) / (Ind1 + Ind2 + Ind3 + Ind4)
Ind1 = 1
```

```
End If
If Ind2 = 1 Then
Brhi(i + 1, j) = Brhi(i + 1, j) + Ind2 * Bn(i, j) / (Ind1 + Ind2 + Ind3 + Ind4)
Ind2 = 1
End If
If Ind3 = 1 Then
Brhi(i, j - 1) = Brhi(i, j - 1) + Ind3 * Bn(i, j) / (Ind1 + Ind2 + Ind3 + Ind4)
Ind3 = 1
End If
If Ind4 = 1 Then
Brhi(i - 1, j) = Brhi(i - 1, j) + Ind4 * Bn(i, j) / (Ind1 + Ind2 + Ind3 + Ind4)
Ind4 = 1
End If

Bn(i, j) = 0

End If

Next i
Next j

For j = 1 To jmax
For i = 1 To imax

Variables.Cells(2).Value = t

Bo(i, j) = Brhi(i, j) + Bn(i, j) + Bsht(i, j) + Brt(i, j) + Bp(i, j)

If Bo(i, j) > 0 Then

If B(i, j) = 0 Then
tb(i, j) = t
End If

Variables.Cells(4).Value = tb(i, j)
Variables.Cells(7).Value = ShootHeight(i, j)
Variables.Cells(10).Value = Brhi(i, j)
Variables.Cells(11).Value = Brt(i, j)
Variables.Cells(12).Value = Bn(i, j)
Variables.Cells(13).Value = Bsht(i, j)
Variables.Cells(14).Value = Bp(i, j)

PhragmitesAustralisGrowthModel

ShootHeight(i, j) = Variables.Cells(7).Value
Brhi(i, j) = Variables.Cells(10).Value
Bn(i, j) = Variables.Cells(11).Value
Bsht(i, j) = Variables.Cells(12).Value
Brt(i, j) = Variables.Cells(13).Value
Bp(i, j) = Variables.Cells(14).Value

End If

B(i, j) = Brhi(i, j) + Bn(i, j) + Bsht(i, j) + Brt(i, j) + Bp(i, j)
```

```
If BioFrac(i, j) = 0 Then
B(i, j) = 0
Brhi(i, j) = 0
Bn(i, j) = 0
Bsht(i, j) = 0
Brt(i, j) = 0
Bp(i, j) = 0
ShootHeight(i, j) = 0
ElseIf B(i, j) > Bmax * BioFrac(i, j) Then
B(i, j) = Bmax * BioFrac(i, j)
Brhi(i, j) = Brhi(i, j) * Bmax * BioFrac(i, j) / B(i, j)
Bn(i, j) = Bn(i, j) * Bmax * BioFrac(i, j) / B(i, j)
Bsht(i, j) = Bsht(i, j) * Bmax * BioFrac(i, j) / B(i, j)
Brt(i, j) = Brt(i, j) * Bmax * BioFrac(i, j) / B(i, j)
Bp(i, j) = Bp(i, j) * Bmax * BioFrac(i, j) / B(i, j)
ShootHeight(i, j) = ShootHeight(i, j) * Bmax * BioFrac(i, j) / B(i, j)
End If

If B(i, j) > 0 And B(i, j) <= Colour.Cells(1).Value * 1000 Then
Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(1, 3).Value
ElseIf B(i, j) > Colour.Cells(1).Value * 1000 And B(i, j) <= Colour.Cells(2).Value * 1000 Then
Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(2, 3).Value
ElseIf B(i, j) > Colour.Cells(2).Value * 1000 And B(i, j) <= Colour.Cells(3).Value * 1000 Then
Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(3, 3).Value
ElseIf B(i, j) > Colour.Cells(3).Value * 1000 And B(i, j) <= Colour.Cells(4).Value * 1000 Then
Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(4, 3).Value
ElseIf B(i, j) > Colour.Cells(4).Value * 1000 And B(i, j) <= Colour.Cells(5).Value * 1000 Then
Topography.Cells(i, j).Interior.ColorIndex = Colour.Cells(5, 3).Value
End If

Next i
Next j

For j = 1 To jmax
B(1, j) = B(2, j)

Brhi(1, j) = Brhi(2, j)
Bn(1, j) = Bn(2, j)
Bsht(1, j) = Bsht(2, j)
Brt(1, j) = Brt(2, j)
Bp(1, j) = Bp(2, j)
ShootHeight(1, j) = ShootHeight(2, j)

B(imax, j) = B(imax - 1, j)
Brhi(imax, j) = Brhi(imax - 1, j)
Bn(imax, j) = Bn(imax - 1, j)
Bsht(imax, j) = Bsht(imax - 1, j)
Brt(imax, j) = Brt(imax - 1, j)
Bp(imax, j) = Bp(imax - 1, j)
ShootHeight(imax, j) = ShootHeight(imax - 1, j)

Next j

For i = 2 To jmax - 1
```

```
B(i, 1) = B(i, 2)
Brhi(i, 1) = Brhi(i, 2)
Bn(i, 1) = Bn(i, 2)
Bsht(i, 1) = Bsht(i, 2)
Brt(i, 1) = Brt(i, 2)
Bp(i, 1) = Bp(i, 2)
ShootHeight(i, 1) = ShootHeight(i, 2)

B(i, jmax) = B(i, jmax - 1)
Brhi(i, jmax) = Brhi(i, jmax - 1)
Bn(i, jmax) = Bn(i, jmax - 1)
Bsht(i, jmax) = Bsht(i, jmax - 1)
Brt(i, jmax) = Brt(i, jmax - 1)
Bp(i, jmax) = Bp(i, jmax - 1)
ShootHeight(i, jmax) = ShootHeight(i, jmax - 1)

Next i

For j = 1 To jmax
For i = 1 To imax
BeginTime.Cells(i, j).Value = tb(i, j)
ActualBiomass.Cells(i, j).Value = B(i, j) / 1000
Rhizomes.Cells(i, j).Value = Brhi(i, j) / 1000
Roots.Cells(i, j).Value = Bn(i, j) / 1000
NewRhizomes.Cells(i, j).Value = Bsht(i, j) / 1000
Shoots.Cells(i, j).Value = Brt(i, j) / 1000
Panicles.Cells(i, j).Value = Bp(i, j) / 1000
ShootLengths.Cells(i, j).Value = ShootHeight(i, j)
Next i
Next j

For j = 1 To jmax
For i = 1 To imax
InitColours.Cells(i, j).Interior.ColorIndex = Topography.Cells(i, j).Interior.ColorIndex
Next i
Next j

Inputs.Cells(15).Value = "t = " + CStr(Round(t, 0)) + "days"

ip = ip + 1
If ip = iprint + 1 Then
ip = 0
End If

Application.Wait (True)

Inputs.Cells(9).Value = t
t = t + dt

Loop While t <= tLast

End Sub
```

```
Sub Interpolator ()

Set Inputs = Worksheets("3d Wat").Range("b3")
Set Inflows = Worksheets("3d Wat").Range("b14")

Set Topography = Worksheets("3d Wat").Range("i3:ab22")
Set Heights = Topography.Offset(Topography.Rows.Count + 4, 0)
Set Velocity = Heights.Offset(Topography.Rows.Count + 4, 0)
Set Friction = Velocity.Offset(Topography.Rows.Count + 4, 0)
Set Direction = Friction.Offset(Topography.Rows.Count + 4, 0)

Set Topography2dWat = Worksheets("2d Wat").Range("i3:ab22")
imax = Topography.Rows.Count
jmax = Topography.Columns.Count

Set Heights2dWat = Topography2dWat.Offset(imax + 4, 0)
Set Velocityx2dWat = Heights2dWat.Offset(0, jmax + 4)
Set Velocityy2dWat = Velocityx2dWat.Offset(0, jmax + 4)

Set GrainSize1 = Worksheets("2d Sed").Range("ah51:ba70")

Inputs.Cells(8).Value = Worksheets("2d Wat").Range("e3")

i = Inputs.Cells(9).Value
j = Inputs.Cells(10).Value

U1 = Velocityx2dWat.Cells(i - 1, j).Value
U2 = Velocityx2dWat.Cells(i, j + 1).Value
U3 = Velocityx2dWat.Cells(i + 1, j).Value
U4 = Velocityx2dWat.Cells(i, j - 1).Value

V1 = Velocityy2dWat.Cells(i - 1, j).Value
V2 = Velocityy2dWat.Cells(i, j + 1).Value
V3 = Velocityy2dWat.Cells(i + 1, j).Value
V4 = Velocityy2dWat.Cells(i, j - 1).Value

H1 = Heights2dWat.Cells(i - 1, j).Value
H2 = Heights2dWat.Cells(i, j + 1).Value
H3 = Heights2dWat.Cells(i + 1, j).Value
H4 = Heights2dWat.Cells(i, j - 1).Value

z1 = Topography2dWat.Cells(i - 1, j).Value
z2 = Topography2dWat.Cells(i, j + 1).Value
z3 = Topography2dWat.Cells(i + 1, j).Value
Z4 = Topography2dWat.Cells(i, j - 1).Value

Inflows.Cells(1).Value = U1
Inflows.Cells(2).Value = U2
Inflows.Cells(3).Value = U3
Inflows.Cells(4).Value = U4

Inflows.Cells(5).Value = V1
Inflows.Cells(6).Value = V2
Inflows.Cells(7).Value = V3
```

```
Inflows.Cells(8).Value = V4

Inflows.Cells(1, 4).Value = z1
Inflows.Cells(2, 4).Value = z2
Inflows.Cells(3, 4).Value = z3
Inflows.Cells(4, 4).Value = Z4

Inflows.Cells(5, 4).Value = H1
Inflows.Cells(6, 4).Value = H2
Inflows.Cells(7, 4).Value = H3
Inflows.Cells(8, 4).Value = H4

imax = Topography.Rows.Count
jmax = Topography.Columns.Count

For j = 1 To jmax
For i = 1 To imax
If i = 1 Then
Z(i, j) = (z2 - z1) * j / jmax + z1
H(i, j) = (H2 - H1) * j / jmax + H1
Velx(i, j) = (U2 - U1) * j / jmax + U1
Vely(i, j) = (V2 - V1) * j / jmax + V1
ElseIf i = imax Then
Z(i, j) = (z3 - Z4) * j / jmax + Z4
H(i, j) = (H3 - H4) * j / jmax + H4
Velx(i, j) = (U3 - U4) * j / jmax + U4
Vely(i, j) = (V3 - V4) * j / jmax + V4
ElseIf j = 1 Then
Z(i, j) = (Z4 - z1) * i / imax + z1
H(i, j) = (H4 - H1) * i / imax + H1
Velx(i, j) = (U4 - U1) * i / imax + U1
Vely(i, j) = (V4 - V1) * i / imax + V1
ElseIf j = jmax Then
Z(i, j) = (z3 - z2) * i / imax + z2
H(i, j) = (H3 - H2) * i / imax + H2
Velx(i, j) = (U3 - U2) * i / imax + U2
Vely(i, j) = (V3 - V2) * i / imax + V2
Else
Z(i, j) = z1
H(i, j) = H1
Velx(i, j) = U1
Vely(i, j) = V1
End If
Next i
Next j

Do Until rep = Worksheets("3d Wat").Range("b24").Value
For j = 2 To jmax - 1
For i = 2 To imax - 1

a = Z(i + 1, j)
b = Z(i - 1, j)
c = Z(i, j + 1)
d = Z(i, j - 1)
```

```
Z(i, j) = Z(i, j) + (a + b + c + d - 4 * Z(i, j)) / 4

a = H(i + 1, j)
b = H(i - 1, j)
c = H(i, j + 1)
d = H(i, j - 1)

H(i, j) = H(i, j) + (a + b + c + d - 4 * H(i, j)) / 4

a = Velx(i + 1, j)
b = Velx(i - 1, j)
c = Velx(i, j + 1)
d = Velx(i, j - 1)

Velx(i, j) = Velx(i, j) + (a + b + c + d - 4 * Velx(i, j)) / 4

a = Vely(i + 1, j)
b = Vely(i - 1, j)
c = Vely(i, j + 1)
d = Vely(i, j - 1)

Vely(i, j) = Vely(i, j) + (a + b + c + d - 4 * Vely(i, j)) / 4

Next i
Next j
rep = rep + 1
Loop

For j = 1 To jmax
For i = 1 To imax

Topography.Cells(i, j).Value = Z(i, j)
Heights.Cells(i, j).Value = H(i, j)
Vel(i, j) = (Velx(i, j) ^ 2 + Vely(i, j) ^ 2) ^ 0.5
Velocity.Cells(i, j).Value = Vel(i, j)

Next i
Next j

End Sub
```

```
Sub Onlyks()

Set Topography = Worksheets("3d Wat").Range("i3:r22")
Set Heights = Topography.Offset(Topography.Rows.Count + 4, 0)
Set Velocity = Heights.Offset(Topography.Rows.Count + 4, 0)
Set Friction = Velocity.Offset(Topography.Rows.Count + 4, 0)
Set Direction = Friction.Offset(Topography.Rows.Count + 4, 0)
Set GrainSize1 = Worksheets("2d Sed").Range("ah51:ba70")
Set Bed = Worksheets("3d Sed").Range("i2")

Set Inputs = Worksheets("3d Sed").Range("b5")

jjmax = 20

ds = Inputs.Cells(8).Value

t = Inputs.Cells(4).Value

ThetaEst = WorksheetFunction.Radians(Inputs.Value)

For jj = 2 To jjmax - 1

If ThetaEst >= 0 And ThetaEst < WorksheetFunction.Radians(45) Then
i = Round(jjmax / 2 * (1 + Tan(Theta(jj))) - jj * Tan(Theta(jj)), 0)
j = jj
ElseIf ThetaEst >= WorksheetFunction.Radians(45) And _
ThetaEst < WorksheetFunction.Radians(90) Then
i = jjmax - jj + 1
j = Round(jjmax / 2 * (1 - Tan(Theta(jj))) + jj * Tan(Theta(jj)), 0)
ElseIf ThetaEst >= WorksheetFunction.Radians(90) _
And ThetaEst < WorksheetFunction.Radians(135) Then
i = jjmax - jj + 1
j = Round(jjmax / 2 * (1 + Tan(Theta(jj))) - jj * Tan(Theta(jj)), 0)
ElseIf ThetaEst >= WorksheetFunction.Radians(135) And _
ThetaEst < WorksheetFunction.Radians(180) Then
i = Round(jjmax / 2 * (1 + Tan(Theta(jj))) - jj * Tan(Theta(jj)), 0)
j = jjmax - jj + 1
ElseIf ThetaEst >= WorksheetFunction.Radians(-45) And _
ThetaEst < 0 Then
i = Round(jjmax / 2 * (1 - Tan(Theta(jj))) + jj * Tan(Theta(jj)), 0)
j = jj
ElseIf ThetaEst >= WorksheetFunction.Radians(-90) And _
ThetaEst < WorksheetFunction.Radians(-45) Then
i = jj
j = Round(jjmax / 2 * (1 - Tan(Theta(jj))) + jj * Tan(Theta(jj)), 0)
ElseIf ThetaEst >= WorksheetFunction.Radians(-135) And _
ThetaEst < WorksheetFunction.Radians(-90) Then
i = jj
j = Round(jjmax / 2 * (1 + Tan(Theta(jj))) - jj * Tan(Theta(jj)), 0)
ElseIf ThetaEst >= WorksheetFunction.Radians(-180) And _
ThetaEst < WorksheetFunction.Radians(-135) Then
i = Round(jjmax / 2 * (1 - Tan(Theta(jj))) + jj * Tan(Theta(jj)), 0)
j = jjmax - jj + 1
End If
```

```
ds = Inputs.Cells(8).Value
ks = 2 * ds
Li(jj) = 175 * (ds * 1000) ^ 0.75 / 1000
fs(jj) = Bed.Cells(7, jj).Value
Vel(jj) = Velocity.Cells(i, j).Value
H(jj) = Heights.Cells(i, j).Value - Topography.Cells(i, j).Value
HBi(jj) = Li(jj) * fs(jj) / 1.88

If Vel(jj) = 0 Then
fs(jj) = 0
Else
fo = 1
Do
If H(jj) < 0 Then
H(jj) = 0.01
End If
fs(jj) = 1 / (-2 * WorksheetFunction.Log10(ks / (3.71 * (H(jj))) + 2.51 * vs / (Vel(jj) * (H(jj)) * fo ^
0.5))) ^ 2
fo = fo + 0.25 * (fs(jj) - fo)
Loop Until (fs(jj) - fo) < 0.1 And (fs(jj) - fo) > -0.1

End If

If t = 0 Or fs(jj) = 0 Then

L(jj) = Li(jj)
HB(jj) = HBi(jj)

'initial ripple length
Else

Us = fs(jj) / 8 * Vel(jj) ^ 2

tb = rho * Us

ts = tb / (rho * (ss - 1) * g * ds)

Rs = (ds * (ss * g / vs ^ 2) ^ (1 / 3)) ^ (3 / 2)
If Rs < 6.61 Then
Shields = 0.1414 * Rs ^ -0.2306
ElseIf Rs > 282.84 Then
Shields = 0.045
Else
Shields = (1 + (0.0223 * Rs) ^ 2.8358) ^ 0.3542 / _
(3.0946 * Rs ^ 0.6769)
End If

'Ripples:
ter = 2.08 * 10 ^ 8 * (ts / Shields) ^ -2.42 * (ds / Us)
'Dunes:
te = 2.05 * 10 ^ -2 * (ds / H(jj)) ^ -3.5 * _
(ts / Shields) ^ -1.12 * (ds / Us)
```

```
If te < ter Then te = ter

GP = ds * (g * (ss - 1) / vs ^ 2) ^ (1 / 3)
gammaL = 0.14 * GP ^ 0.33
gammaH = 0.22 * GP ^ 0.22

HBe = 1 / 6 * H(jj)
Le = 5 * H(jj)

L(jj) = (t / te) ^ gammaL * (Le - Li(jj)) + Li(jj)
HB(jj) = (t / te) ^ gammaH * (HBe - HBi(jj)) + HBi(jj)
If te < t Then
L(jj) = Le
HB(jj) = HBe
End If
End If

Bed.Cells(5, jj).Value = HB(jj)
Bed.Cells(6, jj).Value = L(jj)
Bed.Cells(7, jj).Value = fs(jj)

Next jj

End Sub
```

```
Sub PhragmitesAustralisGrowthModel()

Set Parrameters = Worksheets("3d Veg").Range("b2")
Set Variables = Worksheets("3d Veg").Range("b31")
Set Month = Worksheets("3d Veg").Range("d2")
Set EventTime = Worksheets("3d Veg").Range("e17")
Set XYZ = Worksheets("3d Veg").Range("t2")

t = Variables.Cells(2).Value
tfin = Variables.Cells(3).Value
cc = 1

ts = EventTime.Cells(6).Value + tb

'read input parameters
'Maximum specific growth rate of roots at 20dC
gm = Parrameters.Cells(1).Value
'Specific respiration rate of roots at 20dC
betasht = Parrameters.Cells(2).Value
'Specific respiration  rate of shoots at 20dC
betart = Parrameters.Cells(3).Value
'Specific respiration rate of old rhizomes at 20dC
betarhi = Parrameters.Cells(4).Value
'Specific respiration rate of new rhizomes at 20dC
betan = Parrameters.Cells(5).Value
'Specific respiration rate of panicles at 20dC
betap = Parrameters.Cells(6).Value

'Specific mortality rate of roots at 20dC
gammart = Parrameters.Cells(12).Value
'Specific mortality rate of old rhizomes at 20dC
gammarhi = Parrameters.Cells(13).Value
'Specific mortality rate of new rhizomes at 20dC
gamman = Parrameters.Cells(14).Value
'Fraction of current photosynthesis translocation to below ground structures
eph = Parrameters.Cells(15).Value
'Fraction of shoot assimilates translocation to below ground structures
esht = Parrameters.Cells(16).Value
'Fraction of shoot assimilates translocated for old rhizomes
y = Parrameters.Cells(17).Value
'Fraction of shoot assimilates translocation for inflorescence
p = Parrameters.Cells(18).Value
'Fraction fo current photosynthesis translocation to inflorencence
k = Parrameters.Cells(19).Value
'Fraction of shoot biomass for elongation
q = Parrameters.Cells(20).Value
'Fraction mobilized from rhizomes for root formation
x = Parrameters.Cells(21).Value
'Maximum specific net daily photosynthesis rate at 20dC
Pm = Parrameters.Cells(22).Value
'Half saturation constant of age for shoot phtosynthesis
Kage = Parrameters.Cells(23).Value
'Half saturation constant of PAR for shoot phtosynthesis
Kpar = Parrameters.Cells(24).Value
```

```
'Half saturation constant of age for root growth
Krt = Parrameters.Cells(25).Value
'Temperature constant
Theta = Parrameters.Cells(26).Value
'Conversion constant of carbon dioxcide to ash-free dry weight
kco = Parrameters.Cells(27).Value

Do While t <= tfin

'Specific mortality rate of shoots from tb-tp at 20dC
If t >= tb And t < tp Then
gammasht = Parrameters.Cells(7).Value
'Specific mortality rate of shoots from tb-ts at 20dC
ElseIf t >= tp And t < ts Then
gammasht = Parrameters.Cells(8).Value
'Specific mortality rate of shoots after ts at 20dC
ElseIf t >= ts Then
gammasht = Parrameters.Cells(9).Value
End If
'Specific mortality rate of panicles from tp-ts at 20dC
If t >= tp And t < ts Then
gammap = Parrameters.Cells(10).Value
'Specific mortality rate of panicles after ts at 20dC
ElseIf t >= ts Then
gammap = Parrameters.Cells(11).Value
Else
gammap = 0
End If

Do While c < 13
If t - tb <= 31 Then
c = 0
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 60 Then
c = 1
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 90 Then
c = 2
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 120 Then
c = 3
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
```

```
ElseIf t - tb <= 150 Then
c = 4
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 180 Then
c = 5
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 210 Then
c = 6
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 240 Then
c = 7
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 270 Then
c = 8
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 300 Then
c = 9
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 330 Then
c = 10
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb <= 360 Then
c = 11
Temp = Month.Cells.Offset(c, 1).Value
GlobalRadiation = Month.Cells.Offset(c, 2).Value
SunAngle = Month.Cells.Offset(c, 3).Value
c = 13
ElseIf t - tb > 360 Then
tb = tb + 360
End If

Loop
```

```
ke = -0.0008 * SunAngle ^ 2 + 0.0706 * SunAngle - 0.4
Ipar = 0.45 * GlobalRadiation

'Rhizome Biomass (g/m2)
Brhio = Variables.Cells(10).Value
ShootHeighto = Variables.Cells(7).Value

If t = tb Then
'In the model, growth was initiated by using a fraction of the initial rhizome biomass carried
'over from the last growing season.
GrowthInitiationParameter = 0.06 * Brhio ^ -0.19 'g m-1 m-2
InitialSpecificDensityOfShoots = 0.22 * Brhio ^ 0.74 'g m-1 m-2
'Eqs. (1) and (2) were used in the model again and the other parameters were fine-tuned. The
'growth initiation parameter and the initial specific density defined by the above equations
'were used to calculate the initial shoot biomass and initial shoot height given by:
Bshto = GrowthInitiationParameter * Brhio 'InitialRhizomeBiomass
ShootHeighto = Bshto / InitialSpecificDensityOfShoots
Variables.Cells(7).Value = ShootHeight
Bno = 0
Bpo = 0

i = 1

Else
'Newly formed rhizomes Biomass
Bno = Variables.Cells(12).Value
'Shoot Biomass (g/m2)
Bshto = Variables.Cells(13).Value
'Panicles Biomass (g/m2)
Bpo = Variables.Cells(14).Value
'Reed lenth (m)

End If

'Root Biomass = (g / m2)
Brto = Variables.Cells(11).Value

bleaf = 0.25 * Bshto
LAI = 0.01355 * bleaf ^ 1.02
'where bleaf is the leaf biomass
'LAI = leaf surface area per unit ground area

Iipar = Ipar * Exp(-ke * LAI)

If t >= tb And t < te Then
frhi = 1
Else
frhi = 0
End If
If t >= te And t <= ts Then
fsht = 1
Else
fsht = 0
End If
```

```
If t >= tb And t <= ts Then
fph = 1
Else
fph = 0
End If
If t >= tr And t <= tp Then
frt = 1
Else
frt = 0
End If
If t >= tf Then
ff = 1
Else
ff = 0
End If

Agert = t - tr
If Agert < 0 Then
Agert = 0
End If

'The supply of photosynthesized material for root growth (Grt) is given by
Grt = gm * Theta ^ (Temp - 20) * (Krt / (Krt + Agert)) * Brto
'where gm is the maximum specific growth rate of roots at 20dC; Krt is the half saturation
'coefficient of root age, and Agert is the age of roots in days from the start of root growth.

alpharhi = 0.58 * (Brhio) ^ -0.5
Rhif = alpharhi * Theta ^ (Temp - 20) * Brhio
'where alpharhi is the specific transfer rate of rhizome biomass.
' Rhif is the
'mobilization of stored material from rhizome to roots and shoots during the initial stage of growth;

Agesht = t - tb '?
i = 1

Bsht = Bshto
Rsht = betasht * Theta ^ (Temp - 20) * Bsht
Dsht = gammasht * Theta ^ (Temp - 20) * Bsht
Phsht = Pm * kco * Theta ^ (Temp - 20) * (Iipar / (Kpar + Iipar)) * (Kage / (Kage + Agesht)) * Bsht
k1Bsht = dt * (Phsht - Rsht - Dsht + (1 - x) * Rhif * frhi _
- esht * Bsht * fsht - eph * Phsht * fph - (Bsht * p + Phsht * k) * ff - Grt * frt)
Bsht = Bshto + k1Bsht / 2
Rsht = betasht * Theta ^ (Temp - 20) * Bsht
Dsht = gammasht * Theta ^ (Temp - 20) * Bsht
Phsht = Pm * kco * Theta ^ (Temp - 20) * (Iipar / (Kpar + Iipar)) * (Kage / (Kage + Agesht)) * Bsht
k2Bsht = dt * (Phsht - Rsht - Dsht + (1 - x) * Rhif * frhi _
- esht * Bsht * fsht - eph * Phsht * fph - (Bsht * p + Phsht * k) * ff - Grt * frt)
Bsht = Bshto + k2Bsht / 2
Rsht = betasht * Theta ^ (Temp - 20) * Bsht
Dsht = gammasht * Theta ^ (Temp - 20) * Bsht
Phsht = Pm * kco * Theta ^ (Temp - 20) * (Iipar / (Kpar + Iipar)) * (Kage / (Kage + Agesht)) * Bsht
k3Bsht = dt * (Phsht - Rsht - Dsht + (1 - x) * Rhif * frhi _
- esht * Bsht * fsht - eph * Phsht * fph - (Bsht * p + Phsht * k) * ff - Grt * frt)
Bsht = Bshto + k3Bsht
```

```
Rsht = betasht * Theta ^ (Temp - 20) * Bsht
Dsht = gammasht * Theta ^ (Temp - 20) * Bsht
Phsht = Pm * kco * Theta ^ (Temp - 20) * (Iipar / (Kpar + Iipar)) * (Kage / (Kage + Agesht)) * Bsht
k4Bsht = dt * (Phsht - Rsht - Dsht + (1 - x) * Rhif * frhi _
- esht * Bsht * fsht - eph * Phsht * fph - (Bsht * p + Phsht * k) * ff - Grt * frt)
Bsht = Bshto + (k1Bsht + 2 * k2Bsht + 2 * k3Bsht + k4Bsht) / 6

Brhi = Brhio
'The amount of material mobilized from the rhizomes was taken to be proportional to its
'biomass and daily mean temperature
Rrhi = betarhi * Theta ^ (Temp - 20) * Brhi
Drhi = gammarhi * Theta ^ (Temp - 20) * Brhi
'R and D are the respiration and mortality (gm-2 per day), respectively, which are proportional to their
biomasses and the mean daily temperature.

Phsht = Pm * kco * Theta ^ (Temp - 20) * (Iipar / (Kpar + Iipar)) * (Kage / (Kage + Agesht)) * Bshto

'in this model, we assumed that rhizome reserves were allocated in
'proportion to the shoot biomass of each layer as
RhizomeReserve = (1 - x) * Rhif * frhi
'x is the fraction of Rhif allocated for root growth and the rest for shoots

Rrhi = betarhi * Theta ^ (Temp - 20) * Brhi
Drhi = gammarhi * Theta ^ (Temp - 20) * Brhi
Rhif = alpharhi * Theta ^ (Temp - 20) * Brhi
k1Brhi = dt * (-Rrhi - Drhi - Rhif * frhi + y * esht * Bshto * fsht + y * eph * Phsht * fph)  'B(gm-2/d)
Brhi = Brhio + k1Brhi / 2
Rrhi = betarhi * Theta ^ (Temp - 20) * Brhi
Drhi = gammarhi * Theta ^ (Temp - 20) * Brhi
Rhif = alpharhi * Theta ^ (Temp - 20) * Brhi
k2Brhi = dt * (-Rrhi - Drhi - Rhif * frhi + y * esht * Bshto * fsht + y * eph * Phsht * fph)  'B(gm-2/d)
Brhi = Brhio + k2Brhi / 2
Rrhi = betarhi * Theta ^ (Temp - 20) * Brhi
Drhi = gammarhi * Theta ^ (Temp - 20) * Brhi
Rhif = alpharhi * Theta ^ (Temp - 20) * Brhi
k3Brhi = dt * (-Rrhi - Drhi - Rhif * frhi + y * esht * Bshto * fsht + y * eph * Phsht * fph)  'B(gm-2/d)
Brhi = Brhio + k3Brhi
Rrhi = betarhi * Theta ^ (Temp - 20) * Brhi
Drhi = gammarhi * Theta ^ (Temp - 20) * Brhi
Rhif = alpharhi * Theta ^ (Temp - 20) * Brhi
k4Brhi = dt * (-Rrhi - Drhi - Rhif * frhi + y * esht * Bshto * fsht + y * eph * Phsht * fph)  'B(gm-2/d)
Brhi = Brhio + (k1Brhi + 2 * k2Brhi + 2 * k3Brhi + k4Brhi) / 6

Brt = Brto
Grt = gm * Theta ^ (Temp - 20) * (Krt / (Krt + Agert)) * Brt
Rrt = betart * Theta ^ (Temp - 20) * Brt
Drt = gammart * Theta ^ (Temp - 20) * Brt
k1Brt = dt * (Grt * frt - Rrt - Drt + x * Rhif * frhi)
Brt = Brto + k1Brt / 2
Grt = gm * Theta ^ (Temp - 20) * (Krt / (Krt + Agert)) * Brt
Rrt = betart * Theta ^ (Temp - 20) * Brt
Drt = gammart * Theta ^ (Temp - 20) * Brt
k2Brt = dt * (Grt * frt - Rrt - Drt + x * Rhif * frhi)
Brt = Brto + k2Brt / 2
```

```
Grt = gm * Theta ^ (Temp - 20) * (Krt / (Krt + Agert)) * Brt
Rrt = betart * Theta ^ (Temp - 20) * Brt
Drt = gammart * Theta ^ (Temp - 20) * Brt
k3Brt = dt * (Grt * frt - Rrt - Drt + x * Rhif * frhi)
Brt = Brto + k3Brt
Grt = gm * Theta ^ (Temp - 20) * (Krt / (Krt + Agert)) * Brt
Rrt = betart * Theta ^ (Temp - 20) * Brt
Drt = gammart * Theta ^ (Temp - 20) * Brt
k4Brt = dt * (Grt * frt - Rrt - Drt + x * Rhif * frhi)
Brt = Brto + (k1Brt + 2 * k2Brt + 2 * k3Brt + k4Brt) / 6

Bn = Bno
Rn = betan * Theta ^ (Temp - 20) * Bn
Dn = gamman * Theta ^ (Temp - 20) * Bn
k1Bn = dt * (-Rn - Dn + (1 - y) * esht * Bshto * fsht + (1 - y) * eph * Phsht * fph)
Bn = Bno + k1Bn / 2
Rn = betan * Theta ^ (Temp - 20) * Bn
Dn = gamman * Theta ^ (Temp - 20) * Bn
k2Bn = dt * (-Rn - Dn + (1 - y) * esht * Bshto * fsht + (1 - y) * eph * Phsht * fph)
Bn = Bno + k2Bn / 2
Rn = betan * Theta ^ (Temp - 20) * Bn
Dn = gamman * Theta ^ (Temp - 20) * Bn
k3Bn = dt * (-Rn - Dn + (1 - y) * esht * Bshto * fsht + (1 - y) * eph * Phsht * fph)
Bn = Bno + k3Bn
Rn = betan * Theta ^ (Temp - 20) * Bn
Dn = gamman * Theta ^ (Temp - 20) * Bn
k4Bn = dt * (-Rn - Dn + (1 - y) * esht * Bshto * fsht + (1 - y) * eph * Phsht * fph)
Bn = Bno + (k1Bn + 2 * k2Bn + 2 * k3Bn + k4Bn) / 6

Bp = Bpo
Rp = betap * Theta ^ (Temp - 20) * Bp
Dp = gammap * Theta ^ (Temp - 20) * Bp
k1Bp = dt * (-Rp - Dp + Phsht * k * ff + p * Bshto * ff)
Bp = Bpo + k1Bp / 2
Rp = betap * Theta ^ (Temp - 20) * Bp
Dp = gammap * Theta ^ (Temp - 20) * Bp
k2Bp = dt * (-Rp - Dp + Phsht * k * ff + p * Bshto * ff)
Bp = Bpo + k2Bp / 2
Rp = betap * Theta ^ (Temp - 20) * Bp
Dp = gammap * Theta ^ (Temp - 20) * Bp
k3Bp = dt * (-Rp - Dp + Phsht * k * ff + p * Bshto * ff)
Bp = Bpo + k3Bp
Rp = betap * Theta ^ (Temp - 20) * Bp
Dp = gammap * Theta ^ (Temp - 20) * Bp
k4Bp = dt * (-Rp - Dp + Phsht * k * ff + p * Bshto * ff)
Bp = Bpo + (k1Bp + 2 * k2Bp + 2 * k3Bp + k4Bp) / 6

'Reed Height
If Bsht = 0 Then
ShootHeight = ShootHeighto
Else
ShootHeight = ShootHeighto * (1 + (Bsht - Bshto) * q / (Bsht - (Bsht - Bshto) * q))
'where Q is the fraction of biomass contributed to shoot elongation from each layer.
End If
```

```
'Return values
Brhio = Brhi
Brto = Brt
Bno = Bn
Bshto = Bsht
Bpo = Bp
'ShootHeighto = ShootHeight

Variables.Cells(10).Value = Brhi '+ Bn
Variables.Cells(11).Value = Brt
Variables.Cells(12).Value = Bn '=0
Variables.Cells(13).Value = Bsht
Variables.Cells(14).Value = Bp
Variables.Cells(7).Value = ShootHeight

Variables.Cells(2).Value = t
t = t + dt

Loop

End Sub
```