LISTING PROGRAM D-LiQ

```
'Define Global Variabel
Global NOM, NEQ, TMAX
Global MASS_SOIL(10), STIFFNESS_SOIL(10)
Global MASS_SOIL_REVERSE(10), STIFFNESS_SOIL_REVERSE(10)
Global MATRIX_MASS(10, 10), MATRIX_STIFFNESS(10, 10), MATRIX_DAMPING(10, 10)
Global MATRIX_UNIT_MASS(10, 10), MATRIX_UNIT_STIFFNESS(10, 10)
Global UNIT_STIFFNESS, UNIT_MASS
Global OMEGAF(10), OMEGA(10)
Global RT(10), RC(10), ELV_SOIL(10)
Global MASS_DIAG(10), SK(10)
Global MTX_SK(10, 10), INV_MTX_M(10, 10), INV_MTX_SK(10, 10)
Global REMTX(10, 10), STIFF_CONST(10)
Global transSTIFF_STEP(10, 6000), transREST_FORCE(10, 6000)
Global INDISP(6000, 10), INVELO(6000, 10), DELTA_DISP(6000, 10), REST_FORCE(6000,
10), KP(6000, 10), MTX_DISP(6000, 10)

'Define Global Constant
Const ACC_GRAVITY = 980
Const CON_PI = 3.1415
Const DR = 0.05
Const ALPHA = 0.025
Const BETA = 0.5
Const DT = 0.01
Const Nbeta = 0.25
Const Ngama = 0.5

Sub Computing()

'==================== Input Number of Modes

NOM = 6
TMAX = 22

Dynamic_Properties
Mass_Matrix
```

```
Stiffness_Matrix
Eigen_Problem
Mode_Shape
Damping_Matrix
Elastic_Response
Stress_Strain

End Sub

Sub Dynamic_Properties()

Dim H_SOIL(10), WET_DENSITY(10), PLASTICITY(10), KO(10), OCR(10), kx(10)
Dim VOID_RATIO(10), COHESION(10), FRICTION_ANGLE(10), SOIL_TYPE(10), OS(10)
Dim WEIGHT_SOIL(10), VER_STRESS_1(10), HOR_STRESS_2(10), HOR_STRESS_3(10)
Dim EFFECTIVE_STRESS(10), SHEAR_MODULUS(10), YIELD_STRESS(10),
YIELD_FORCE(10), YIELD_POINT(10)

For i = 1 To NOM

'==================== Read Data From Soil_Profile Sheet

   H_SOIL(i) = Sheet1.Cells(i + 1, 2)
   WET_DENSITY(i) = Sheet1.Cells(i + 1, 3)
   PLASTICITY(i) = Sheet1.Cells(i + 1, 4)
   OCR(i) = Sheet1.Cells(i + 1, 5)
   VOID_RATIO(i) = Sheet1.Cells(i + 1, 6)
   COHESION(i) = Sheet1.Cells(i + 1, 7)
   FRICTION_ANGLE(i) = Sheet1.Cells(i + 1, 8)
   SOIL_TYPE(i) = Sheet1.Cells(i + 1, 9)
   OS(i) = Sheet1.Cells(i + 1, 10)

'==================== Command To Determine Soil Weight

   WEIGHT_SOIL(i) = WET_DENSITY(i) * 100 * 100 * 0.5 * H_SOIL(i) / 1000

'==================== Command To Determine Soil Mass
```

```
    If i < 2 Then
      MASS_SOIL(i) = WEIGHT_SOIL(i) / ACC_GRAVITY
    Else
      MASS_SOIL(i) = (WEIGHT_SOIL(i - 1) + WEIGHT_SOIL(i)) / ACC_GRAVITY
    End If

'==================== Command To Determine Lateral Stress Coefficient

    If SOIL_TYPE(i) = 2 Then ' (1 for sand ; 2 for clay)
      If PLASTICITY(i) <= 40 Then
        KO(i) = 0.4 + 0.007 * PLASTICITY(i)
      Else
        KO(i) = 0.68 + 0.001 * PLASTICITY(i)
      End If
    Else
      KO(i) = 1 - Sin(FRICTION_ANGLE(i) * CON_PI / 180)
    End If

'==================== Command To Determine Soil Coefficient

    If PLASTICITY(i) < 9 Then
      kx(i) = 0
    ElseIf PLASTICITY(i) > 10 And PLASTICITY(i) < 19 Then
      kx(i) = 0.18
    ElseIf PLASTICITY(i) > 20 And PLASTICITY(i) < 39 Then
      kx(i) = 0.3
    ElseIf PLASTICITY(i) > 40 And PLASTICITY(i) < 79 Then
      kx(i) = 0.41
    ElseIf PLASTICITY(i) > 80 And PLASTICITY(i) < 99 Then
      kx(i) = 0.48
    ElseIf PLASTICITY(i) > 100 Then
      kx(i) = 0.5
    End If

'==================== Command To Determine Effective Stress

    If i < 2 Then
```

```
    VER_STRESS_1(i) = WET_DENSITY(i) * H_SOIL(i) / 1000
  Else
    VER_STRESS_1(i) = WET_DENSITY(i) * H_SOIL(i) / 1000 + VER_STRESS_1(i - 1)
  End If

  HOR_STRESS_2(i) = VER_STRESS_1(i) * KO(i)
  HOR_STRESS_3(i) = HOR_STRESS_2(i)
  EFFECTIVE_STRESS(i) = (VER_STRESS_1(i) + HOR_STRESS_2(i) + HOR_STRESS_3(i))
/ 3

'==================== Command To Determine Shear Modulus

  If SOIL_TYPE(i) = 2 Then ' (1 for sand ; 2 for clay)
    SHEAR_MODULUS(i) = (331 * OCR(i) ^ kx(i)) * ((2.97 - VOID_RATIO(i)) ^ 2 / (1 +
VOID_RATIO(i))) * EFFECTIVE_STRESS(i) ^ 0.5
  Else
    SHEAR_MODULUS(i) = (700) * ((2.17 - VOID_RATIO(i)) ^ 2 / (1 + VOID_RATIO(i))) *
EFFECTIVE_STRESS(i) ^ 0.5
  End If

'==================== Command To Determine Initialize Stiffness

  STIFFNESS_SOIL(i) = SHEAR_MODULUS(i) * 100 * 100 / H_SOIL(i)

'==================== Command To Determine Yield Stress

  YIELD_STRESS(i) = ((((1 + KO(i)) / 2) * EFFECTIVE_STRESS(i) *
Sin(FRICTION_ANGLE(i) * CON_PI / 180) + COHESION(i) * Cos(FRICTION_ANGLE(i) *
CON_PI / 180)) ^ 2 - (((1 - KO(i)) / 2) * EFFECTIVE_STRESS(i)) ^ 2) ^ 0.5

'==================== Command To Determine Yield Force

  YIELD_FORCE(i) = OS(i) * YIELD_STRESS(i) * 100 * 100

'==================== Command To Determine Yield Point

  YIELD_POINT(i) = YIELD_FORCE(i) / (STIFFNESS_SOIL(i) / 3)
```

```
'==================== Write Data To Dynamic_Properties Sheet

    Sheet2.Cells(i + 1, 2) = WEIGHT_SOIL(i)
    Sheet2.Cells(i + 1, 3) = MASS_SOIL(i)
    Sheet2.Cells(i + 1, 4) = KO(i)
    Sheet2.Cells(i + 1, 5) = EFFECTIVE_STRESS(i)
    Sheet2.Cells(i + 1, 6) = SHEAR_MODULUS(i)
    Sheet2.Cells(i + 1, 7) = STIFFNESS_SOIL(i)
    Sheet2.Cells(i + 1, 8) = YIELD_STRESS(i)
    Sheet2.Cells(i + 1, 9) = YIELD_FORCE(i)
    Sheet2.Cells(i + 1, 10) = YIELD_POINT(i)

Next i

For i = 1 To NOM
    RT(i) = YIELD_FORCE(NOM + 1 - i)
    RC(i) = -RT(i)
    ELV_SOIL(i) = H_SOIL(NOM + 1 - i)
Next i

End Sub

Sub Mass_Matrix()

'==================== Command To Reverse Mass Array

For i = 1 To NOM
    MASS_SOIL_REVERSE(i) = MASS_SOIL(NOM + 1 - i)
Next i

'==================== Command To Assemble Mass Matrix

For i = 1 To NOM
    For j = 1 To NOM
        If i = j Then
            MATRIX_MASS(i, j) = MASS_SOIL_REVERSE(i)
```

```
        MASS_DIAG(i) = MATRIX_MASS(i, j)
      Else
        MATRIX_MASS(i, j) = 0
      End If
      'Write Data To Matrix
      Sheet3.Cells(i + 1, j + 1) = MATRIX_MASS(i, j)
    Next j
Next i

UNIT_MASS = MASS_SOIL_REVERSE(1)

For i = 1 To NOM
    For j = 1 To NOM
      MATRIX_UNIT_MASS(i, j) = MATRIX_MASS(i, j) / UNIT_MASS
    Next j
Next i

End Sub

Sub Stiffness_Matrix()

Dim ARRAY_STIFFNESS(10, 10)

'==================== Command To Reverse Array

For i = 1 To NOM
    STIFFNESS_SOIL_REVERSE(i) = STIFFNESS_SOIL(NOM + 1 - i)
Next i

'==================== Command To Determine Stiffness Prime

For i = 1 To NOM
    For j = 1 To NOM
      If i = j Then
        ARRAY_STIFFNESS(i, j) = STIFFNESS_SOIL_REVERSE(i)
        SK(i) = ARRAY_STIFFNESS(i, j)
      Else
```

```
        ARRAY_STIFFNESS(i, j) = 0
      End If
    Next j
Next i

'==================== Command To Assemble Stiffness Matrix

For i = 1 To NOM
   For j = 1 To NOM
      If i = j Then
        If i < NOM Then
          MATRIX_STIFFNESS(i, j) = STIFFNESS_SOIL_REVERSE(i) +
STIFFNESS_SOIL_REVERSE(i + 1)
        Else
          MATRIX_STIFFNESS(i, j) = STIFFNESS_SOIL_REVERSE(i)
        End If
      ElseIf i <> j And (i + 1 = j) Then
        MATRIX_STIFFNESS(i, j) = -STIFFNESS_SOIL_REVERSE(i + 1)
      ElseIf i <> j And (j + 1 = i) Then
        MATRIX_STIFFNESS(i, j) = -STIFFNESS_SOIL_REVERSE(i)
      Else
        MATRIX_STIFFNESS(i, j) = 0
      End If
      'Write Data To Matrix Sheet
      Sheet3.Cells(i + 11, j + 1) = MATRIX_STIFFNESS(i, j)
   Next j
Next i

UNIT_STIFFNESS = STIFFNESS_SOIL_REVERSE(1)

For i = 1 To NOM
   For j = 1 To NOM
      MATRIX_UNIT_STIFFNESS(i, j) = MATRIX_STIFFNESS(i, j) / UNIT_STIFFNESS
   Next j
Next i

End Sub
```

```
Sub Eigen_Problem()

Dim AAA(10, 10), BB(10, 10)
Dim P(10), AA(10), COLB(10)
Dim U(40), V(40), DU(40), DV(40), X(40), D(40)
Dim DD(10), B(10), C(10)
Dim UK, UM

UK = UNIT_STIFFNESS
UM = UNIT_MASS

For i = 1 To NOM
   For j = 1 To NOM
      AAA(i, j) = MATRIX_UNIT_STIFFNESS(i, j) / MATRIX_UNIT_MASS(i, i)
   Next j
Next i
For i = 1 To NOM
   For j = 1 To NOM
      BB(i, j) = AAA(i, j)
   Next j
Next i
For iter = 1 To NOM - 1
   TRACE = 0
   For i = 1 To NOM
      TRACE = TRACE + BB(i, i)
   Next i
   AK = iter
   P(iter) = TRACE / AK
   For i = 1 To NOM
      BB(i, i) = BB(i, i) - P(iter)
   Next i
   For j = 1 To NOM
      For i = 1 To NOM
         COLB(i) = BB(i, j)
      Next i
      For i = 1 To NOM
```

```
            BB(i, j) = 0
            For ix = 1 To NOM
               BB(i, j) = BB(i, j) + AAA(i, ix) * COLB(ix)
            Next ix
         Next i
      Next j
Next iter

P(NOM) = BB(1, 1)
R = (-1)
For iter = 1 To NOM
   P(iter) = R * (P(iter))
Next iter

For i = 1 To NOM
   AA(i) = P(i)
Next i

AA(0) = 1
NE = NOM

If NE = 2 Then KZ = 1: GoTo 1870
Eps = 0.00001
KZ = 1
R = 0
If NE = 2 Then GoTo 1810

1430 U(KZ) = AA(NE - 1) / AA(NE - 2)
    V(KZ) = AA(NE) / AA(NE - 2)
1450 B(0) = AA(0)
    B(1) = AA(1) - U(KZ)
    For i = 2 To NE
    B(i) = AA(i) - (B(i - 1) * U(KZ)) - (B(i - 2) * V(KZ))
    Next i
    C(0) = B(0)
    C(1) = B(1) - U(KZ)
    For i = 2 To NE - 1
```

```
    C(i) = B(i) - (C(i - 1) * U(KZ)) - (C(i - 2) * V(KZ))
    Next i
    DU(KZ) = (B(NE - 1) * C(NE - 2) - B(NE) * C(NE - 3)) / (C(NE - 2) ^ 2 - C(NE - 1) * C(NE - 3))
    DV(KZ) = (C(NE - 2) * B(NE) - C(NE - 1) * B(NE - 1)) / (C(NE - 2) ^ 2 - C(NE - 1) * C(NE - 3))
    U(KZ) = U(KZ) + DU(KZ)
    V(KZ) = V(KZ) + DV(KZ)
    If Abs(DU(KZ)) + Abs(DV(KZ)) <= Eps Then GoTo 1610
    GoTo 1450
1610 X(KZ) = (-U(KZ) + (U(KZ) ^ 2 - 4 * 1 * V(KZ)) ^ 0.5) / 2
    X(KZ + 1) = (-U(KZ) - (U(KZ) ^ 2 - 4 * 1 * V(KZ)) ^ 0.5) / 2
    D(KZ + R) = X(KZ): D(KZ + 1 + R) = X(KZ + 1)
    NE = NE - 2
    For s = 0 To NE
    AA(s) = B(s)
    Next s
    If NE = 2 Then GoTo 1810
    If NE < 2 Then GoTo 1780
    KZ = KZ + 1
    R = R + 1
    GoTo 1430
1780 D(2 * KZ + 1) = -AA(1)
    GoTo 1960
1810 X(KZ) = (-AA(1) + (AA(1) ^ 2 - 4 * 1 * AA(2)) ^ 0.5) / 2
    X(KZ + 1) = (-AA(1) - (AA(1) ^ 2 - 4 * 1 * AA(2)) ^ 0.5) / 2
    D(2 * KZ + 1) = X(KZ): D(2 * KZ + 2) = X(KZ + 1)
    GoTo 1930
1870 X(KZ) = (-AA(1) + (AA(1) ^ 2 - 4 * 1 * AA(2)) ^ 0.5) / 2
    X(KZ + 1) = (-AA(1) - (AA(1) ^ 2 - 4 * 1 * AA(2)) ^ 0.5) / 2
    D(KZ) = X(KZ): D(KZ + 1) = X(KZ + 1)
1930 Rem
1960 For ji = 1 To NOM: DD(ji) = D(ji): Next ji
    For j = 1 To NOM - 1
    For i = 1 To NOM - j
    If DD(i) < DD(i + 1) Then GoTo 2100
    T = DD(i)
```

```
    DD(i) = DD(i + 1)
    DD(i + 1) = T
2100 Next i
    Next j
    'Determine Angular Frequency
    For j = 1 To NOM
    OMEGAF(j) = (DD(j) * UK / UM) ^ 0.5
    Next j

For i = 1 To NOM
    OMEGA(i) = OMEGAF(NOM + 1 - i)
    'Write Data To Dynamic_Properties Sheet
    Sheet2.Cells(i + 1, 11) = OMEGA(i)
Next i

End Sub

Sub Mode_Shape()

Dim UV(10, 10), UT(10, 10)
Dim P(10), MS(10, 10), PM(10, 10), MM(10), PA(10)

MVC = NOM

If NOM > 2 Then GoTo 2470
For j = 1 To NOM: UV(1, j) = 1: Next j

2430 For j = 1 To NOM
    UV(2, j) = ((SK(1) + SK(2) - OMEGAF(j) ^ 2 * MASS_DIAG(1))) / SK(2)
    Next j
    GoTo 2570
2470 For j = 1 To MVC
    UV(1, j) = 1
    Next j
    For j = 1 To MVC
    For i = 3 To NOM
    UV(2, j) = ((SK(1) + SK(2) - OMEGAF(j) ^ 2 * MASS_DIAG(1))) / SK(2)
```

```
    UV(i, j) = (-UV(i - 2, j) * SK(i - 1) + (SK(i - 1) + SK(i) - OMEGAF(j) ^ 2 * MASS_DIAG(i -
1)) * UV(i - 1, j)) / SK(i)
    Next i
    Next j
2570 For i = 1 To MVC
    P(i) = 0
    For j = 1 To NOM
    UT(i, j) = UV(j, i)
    P(i) = P(i) + UT(i, j) * MASS_DIAG(j)
    Next j
    Next i
    For i = 1 To NOM
    MS(i, i) = MASS_DIAG(i)
    Next i
    For i = 1 To MVC
    For j = 1 To NOM
    PM(i, j) = 0
    PM(i, j) = PM(i, j) + UT(i, j) * MS(j, j)
    Next j
    Next i
    For i = 1 To MVC
    MM(i) = 0
    For j = 1 To NOM
    MM(i) = MM(i) + PM(i, j) * UV(j, i)
    PA(i) = P(i) / MM(i)
    Next j
    Next i

For i = 1 To NOM
   For j = 1 To NOM
      'Write Data To Matrix Sheet
      Sheet3.Cells(i + 21, j + 1) = UV(i, j)
   Next j
Next i

For i = 1 To NOM
   'Write Data To Matrix Sheet
```

```
   Sheet3.Cells(i + 21, 9) = PA(i)
Next i

End Sub

Sub Damping_Matrix()

'==================== Command To Assemble Damping Matrix

BETA_D = (2 * DR * OMEGAF(5) - 2 * DR * OMEGAF(3)) / (OMEGAF(5) ^ 2 - OMEGAF(3)
^ 2)
ALFA_D = (2 * DR * OMEGAF(5)) - (BETA_D * OMEGAF(5) ^ 2)

For i = 1 To NOM
   For j = 1 To NOM
      MATRIX_DAMPING(i, j) = ALFA_D * MATRIX_MASS(i, j) + BETA_D *
MATRIX_STIFFNESS(i, j)
      'Write Data To Dynamic_Properties Sheet
      Sheet3.Cells(i + 31, j + 1) = MATRIX_DAMPING(i, j)
   Next j
Next i

End Sub

Sub Elastic_Response()

Dim YB(6000), MTX_A(10, 10), MTX_B(10, 10), MTX_C(10, 10)
Dim DISP(10, 6000), VELO(10, 6000), ACC(10, 6000)
Dim arrDISP(10), arrVELO(10), arrACC(10)
Dim arrDISP_1(10), arrVELO_1(10), arrACC_1(10)
Dim DELTA_EQ(10, 6000), DELTA_P0(10, 6000), DELTA_P1(10)
Dim DELTA_A(10), DELTA_V(10), DELTA_D(10)
Dim MTX_VELO(6000, 10), MTX_ACC(6000, 10), MTX_REST(6000, 10)
Dim decTemp1, decTemp2, decTempD
Dim INDELTA_DISP(10, 6000)
Dim arrSTIFF_STEP_1(10), arrREST_FORCE_1(10)
```

```
'Input EQ Data

NEQ = (TMAX / DT) + 1

For i = 1 To NEQ
    YB(i) = Sheet7.Cells(i + 2, 2)
Next i

'Numerical Solution By Beta Newmark Direct Integration Method

For iter = 1 To NEQ

A1 = 1 / (Nbeta * DT ^ 2)
A2 = Ngama / (Nbeta * DT)
A3 = 1 / (Nbeta * DT)
A4 = Ngama / Nbeta
A5 = 1 / (2 * Nbeta)
A6 = ((Ngama / (2 * Nbeta)) - 1) * DT
A7 = (1 - Ngama / (2 * Nbeta)) * DT

If iter < 2 Then
  For ii = 1 To NOM
     STIFF_CONST(ii) = SK(ii) / 3
  Next ii
Else
  For ii = 1 To NOM
     STIFF_CONST(ii) = arrSTIFF_STEP_1(ii)
  Next ii
End If

Call Rematrix

'=================== Command To Determine Damping Matrix

'BETA_D = (2 * DR * OMEGAF(3) - 2 * DR * OMEGAF(1)) / (OMEGAF(3) ^ 2 - OMEGAF(1)
^ 2)
'ALFA_D = (2 * DR * OMEGAF(3)) - (BETA_D * OMEGAF(3) ^ 2)
```

```
'For ii = 1 To NOM
   'For jj = 1 To NOM
      'MTX_C(ii, jj) = ALFA_D * MATRIX_MASS(ii, jj) + BETA_D * REMTX(ii, jj)
   'Next jj
'Next ii

'==================== Command To Determine Effective Stiffness

For ii = 1 To NOM
   For jj = 1 To NOM
      MTX_SK(ii, jj) = REMTX(ii, jj) + A1 * MATRIX_MASS(ii, jj) + A2 *
MATRIX_DAMPING(ii, jj)
   Next jj
Next ii

'==================== Command To Determine 'A' Matrix

For ii = 1 To NOM
   For jj = 1 To NOM
      MTX_A(ii, jj) = A3 * MATRIX_MASS(ii, jj) + A4 * MATRIX_DAMPING(ii, jj)
   Next jj
Next ii

'==================== Command To Determine 'B' Matrix

For ii = 1 To NOM
   For jj = 1 To NOM
      MTX_B(ii, jj) = A5 * MATRIX_MASS(ii, jj) + A6 * MATRIX_DAMPING(ii, jj)
   Next jj
Next ii

'==================== Command To Determine Initial Condition

For ii = 1 To NOM
   DISP(ii, 1) = 0
   VELO(ii, 1) = 0
```

```
    ACC(ii, 1) = 0
Next ii

If iter < 2 Then
  For ii = 1 To NOM
    arrVELO(ii) = 0
    arrACC(ii) = 0
  Next ii
Else
  For ii = 1 To NOM
    arrVELO(ii) = arrVELO_1(ii)
    arrACC(ii) = arrACC_1(ii)
  Next ii
End If

For ii = 1 To NOM
    DELTA_EQ(ii, iter) = YB(iter + 1) - YB(iter)
Next ii

'Increment Force
For ii = 1 To NOM
    DELTA_P0(ii, iter) = (DELTA_EQ(ii, iter)) * MASS_DIAG(ii)
Next ii

For ii = 1 To NOM
    decTemp1 = 0
    For jj = 1 To NOM
        decTemp1 = decTemp1 + (MTX_A(ii, jj) * arrVELO(jj))
    Next jj
    decTemp2 = 0
    For jj = 1 To NOM
        decTemp2 = decTemp2 + (MTX_B(ii, jj) * arrACC(jj))
    Next jj
    DELTA_P1(ii) = DELTA_P0(ii, iter) + decTemp1 + decTemp2
Next ii

Call Inv_StiffMatrix
```

```
'Increment Displacement
For ii = 1 To NOM
   decTempD = 0
   For jj = 1 To NOM
      decTempD = decTempD + (INV_MTX_SK(ii, jj) * DELTA_P1(jj))
   Next jj
   DELTA_D(ii) = decTempD
Next ii

'Increment  Velocity
For ii = 1 To NOM
   DELTA_V(ii) = A2 * DELTA_D(ii) - A4 * arrVELO(ii) - A7 * arrACC(ii)
Next ii

'Increment  Acceleration
For ii = 1 To NOM
   DELTA_A(ii) = A1 * DELTA_D(ii) - A3 * arrVELO(ii) - A5 * arrACC(ii)
Next ii

'Next Step Displacement
For ii = 1 To NOM
   DISP(ii, iter + 1) = DISP(ii, iter) + DELTA_D(ii)
Next ii

'Next Step Velocity
For ii = 1 To NOM
   VELO(ii, iter + 1) = VELO(ii, iter) + DELTA_V(ii)
Next ii

'Next Step Acceleration
For ii = 1 To NOM
   ACC(ii, iter + 1) = ACC(ii, iter) + DELTA_A(ii)
Next ii

'Set Next Step Result
For ii = 1 To NOM
```

```
    arrDISP_1(ii) = DISP(ii, iter + 1)
    arrVELO_1(ii) = VELO(ii, iter + 1)
    arrACC_1(ii) = ACC(ii, iter + 1)
Next ii

'Built Transpose Matrix
For xx = 1 To NOM
For yy = 1 To NEQ

DELTA_DISP(yy, xx) = INDELTA_DISP(xx, yy)
INDISP(yy, xx) = DISP(xx, yy)
INVELO(yy, xx) = VELO(xx, yy)

Next yy
Next xx

Call Q_Hyst

For ii = 1 To NOM
    arrSTIFF_STEP_1(ii) = KP(iter + 1, ii)
Next ii

Next iter

'Set Response Matrix
For iter = 1 To NEQ
For ii = 1 To NOM
    MTX_DISP(iter, ii) = DISP(ii, iter)
    MTX_VELO(iter, ii) = VELO(ii, iter)
    MTX_ACC(iter, ii) = ACC(ii, iter)
    'MTX_REST(iter, ii) = REST_FORCE(iter, ii)
    'Write Data To Result Sheet
    Sheet7.Cells(iter + 2, ii + 2) = MTX_DISP(iter, ii)
    Sheet7.Cells(iter + 2, ii + 8) = MTX_VELO(iter, ii)
    Sheet7.Cells(iter + 2, ii + 14) = MTX_ACC(iter, ii)
    'sheet7.Cells(iter + 2, ii + 20) = MTX_REST(iter, ii)
Next ii
```

```
Next iter

End Sub

Sub Inv_StiffMatrix()

Dim OP2_MATRIX(20, 20)

'Inverse Effective Stiffness Matrix By Gauss Elimination Method
'=====================================================================
For NN = 1 To NOM
   For MMM = 1 To NOM
      OP2_MATRIX(MMM, NN) = MTX_SK(MMM, NN)
   Next MMM
Next NN

For NN = 1 To NOM
   For MMM = 1 To NOM
      If NN = MMM Then
         OP2_MATRIX(MMM, NN + NOM) = 1
      Else
         OP2_MATRIX(MMM, NN + NOM) = 0
      End If
   Next MMM
Next NN

For kj = 1 To NOM
  If OP2_MATRIX(kj, kj) = 0 Then
     For NN = kj To NOM
      If OP2_MATRIX(NN, kj) <> 0 Then line_1 = NN: Exit For
     Next NN
     For MMM = kj To NOM * 2
       temporary_2 = OP2_MATRIX(kj, MMM)
       OP2_MATRIX(kj, MMM) = OP2_MATRIX(line_1, MMM)
       OP2_MATRIX(line_1, MMM) = temporary_2
     Next MMM
  End If
```

```
   elem2 = OP2_MATRIX(kj, kj)
  For NN = kj To 2 * NOM
   OP2_MATRIX(kj, NN) = OP2_MATRIX(kj, NN) / elem2
  Next NN

  For NN = 1 To NOM
      If NN = kj And NN = 10 Then Exit For
      If NN = kj And NN < 10 Then NN = NN + 1
     If OP2_MATRIX(NN, kj) <> 0 Then
       multiplier_2 = OP2_MATRIX(NN, kj) / OP2_MATRIX(kj, kj)
       For MMM = kj To 2 * NOM
         OP2_MATRIX(NN, MMM) = OP2_MATRIX(NN, MMM) - OP2_MATRIX(kj, MMM) *
multiplier_2
       Next MMM
     End If
  Next NN
Next kj

For NN = 1 To NOM
   For kj = 1 To NOM
      INV_MTX_SK(NN, kj) = OP2_MATRIX(NN, NOM + kj)
   Next kj
Next NN

End Sub

Sub Rematrix()

For i = 1 To NOM
   For j = 1 To NOM
      If i = j Then
        If i < NOM Then
          REMTX(i, j) = STIFF_CONST(i) + STIFF_CONST(i + 1)
        Else
          REMTX(i, j) = STIFF_CONST(i)
        End If
```

```
        ElseIf i <> j And (i + 1 = j) Then
          REMTX(i, j) = -STIFF_CONST(i + 1)
        ElseIf i <> j And (j + 1 = i) Then
          REMTX(i, j) = -STIFF_CONST(i)
        Else
          REMTX(i, j) = 0
        End If
    Next j
Next i

End Sub

Sub Q_Hyst()

Dim STIFFNESS_EFF(6000, 10), KEY(6000, 10), YT(6000, 10), YC(6000, 10)

Dim YIELDT(10), YIELDC(10), YTF(10), YCF(10), RMA(10), RMI(10), RUA(10), RUI(10)
Dim K(10), SKN1C(10), SKN1T(10), SKN2(10), N(10), L(10), X0C(10), X0T(10)
Dim DISPMA(10), DISPMI(10)

For j = 1 To NOM
    YIELDT(j) = RT(j) / (SK(j) / 3)
    YIELDC(j) = RC(j) / (SK(j) / 3)
    YTF(j) = RT(j) / (SK(j) / 3)
    YCF(j) = RC(j) / (SK(j) / 3)
    RMA(j) = RT(j)
    RMI(j) = RC(j)
    RUA(j) = RMA(j)
    RUI(j) = RMI(j)
Next j

'Initial Parameters

For j = 1 To NOM
    INVELO(1, j) = 0
    INDISP(1, j) = 0
    KP(1, j) = SK(j) / 3
```

```
    REST_FORCE(1, j) = 0
    KEY(1, j) = 0
    YT(1, j) = YIELDT(j)
    YC(1, j) = YIELDC(j)
    K(j) = 1
    SKN1C(j) = 0
    SKN1T(j) = 0
Next j


For j = 1 To NOM
For i = 1 To NEQ

    If (INDISP(i + 1, j) < YCF(j) Or INDISP(i + 1, j) > YTF(j)) Then
       K(j) = 2
    End If

'KEY Conditions

       If (K(j) = 1) Then
         If (INDISP(i + 1, j) > YCF(j) And INDISP(i + 1, j) < YTF(j)) Then
            KEY(i + 1, j) = 0
         End If
       Else
         K(j) = 2
          If (INDISP(i + 1, j) > YIELDT(j)) Then
            KEY(i + 1, j) = 1
            L(j) = 1
          ElseIf (INDISP(i + 1, j) < YIELDC(j)) Then
            KEY(i + 1, j) = -1
            L(j) = 2
          ElseIf (INDISP(i + 1, j) > YIELDC(j) And INDISP(i + 1, j) < YIELDT(j)) Then
          KEY(i + 1, j) = 0
           If (L(j) = 2) Then
             If (INDISP(i + 1, j) < X0C(j)) Then
                  N(j) = 1
              Else
```

```
                N(j) = 2
            End If
           Else
            If (INDISP(i + 1, j) > X0T(j)) Then
                N(j) = 1
             Else
                N(j) = 2
            End If
           End If
          End If
        End If

'Conditionality Requirements for Next Step

'Elastic Response KEY = 0

      If (K(j) = 1) Then
        If (KEY(i + 1, j) = 0) Then
          YT(i + 1, j) = YTF(j)
          YC(i + 1, j) = YCF(j)
          KP(i + 1, j) = SK(j) / 3
          REST_FORCE(i + 1, j) = RMA(j) - (YTF(j) - INDISP(i + 1, j)) * (SK(j) / 3)
        End If
      Else
       If (K(j) = 2# And KEY(i + 1, j) = 0) Then
          If (N(j) = 1) Then
          YT(i + 1, j) = YIELDT(j)
          YC(i + 1, j) = YIELDC(j)
          If ((INVELO(i, j) * INVELO(i + 1, j) > 0 And INVELO(i + 1, j) > 0) Or (INVELO(i, j) *
INVELO(i + 1, j) > 0 And INVELO(i + 1, j) < 0) Or (INVELO(i, j) * INVELO(i + 1, j) < 0 And
INVELO(i + 1, j) > 0) Or (INVELO(i, j) * INVELO(i + 1, j) < 0 And INVELO(i + 1, j) < 0)) Then
              If (L(j) = 2) Then
                If (INDISP(i + 1, j) > DISPMI(j)) Then
                  If (SKN1C(j) = 0) Then
                    KP(i + 1, j) = SKN1T(j)
                  Else
                    KP(i + 1, j) = SKN1C(j)
```

```
                End If
              Else
                KP(i + 1, j) = SKN2(j)
                L(j) = 1
                N(j) = 2
              End If
              REST_FORCE(i + 1, j) = RMI(j) + (INDISP(i + 1, j) - DISPMI(j)) * KP(i + 1, j)
            Else
              If (INDISP(i + 1, j) < DISPMA(j)) Then
                If (SKN1T(j) = 0) Then
                  KP(i + 1, j) = SKN1C(j)
                Else
                  KP(i + 1, j) = SKN1T(j)
                End If
              Else
                KP(i + 1, j) = SKN2(j)
                L(j) = 2
                N(j) = 2
              End If
              REST_FORCE(i + 1, j) = RMA(j) - (DISPMA(j) - INDISP(i + 1, j)) * KP(i + 1, j)
            End If
          End If
        ElseIf (N(j) = 2) Then
          YT(i + 1, j) = YIELDT(j)
          YC(i + 1, j) = YIELDC(j)
          If (L(j) = 2) Then
            SKN2(j) = RUA(j) / (YIELDT(j) - X0C(j))
            If (INVELO(i, j) * INVELO(i + 1, j) > 0 And INVELO(i + 1, j) > 0) Then
              KP(i + 1, j) = SKN2(j)
              REST_FORCE(i + 1, j) = (-(X0C(j) - INDISP(i + 1, j)) / (YIELDT(j) - X0C(j))) *
(RUA(j))
            ElseIf (INVELO(i, j) * INVELO(i + 1, j) < 0) Then
              If (SKN1T(j) = 0) Then
                KP(i + 1, j) = SKN1C(j)
              Else
                KP(i + 1, j) = SKN1T(j)
              End If
```

```
            If (INDISP(i, j) < INDISP(i + 1, j)) Then
              REST_FORCE(i + 1, j) = (-(X0C(j) - INDISP(i + 1, j)) / (YIELDT(j) - X0C(j))) *
(RUA(j))
              DISPMA(j) = INDISP(i + 1, j)
              RMA(j) = REST_FORCE(i + 1, j)
              X0T(j) = (DISPMA(j) - RMA(j) / KP(i + 1, j))
              N(j) = 1
              L(j) = 1
            Else
              DISPMA(j) = INDISP(i, j)
              RMA(j) = REST_FORCE(i, j)
              X0T(j) = (DISPMA(j) - RMA(j) / KP(i + 1, j))
              REST_FORCE(i + 1, j) = RMA(j) - (DISPMA(j) - INDISP(i + 1, j)) * KP(i + 1, j)
              N(j) = 1
              L(j) = 1
            End If
          End If
        Else
          SKN2(j) = RUI(j) / (YIELDC(j) - X0T(j))
          If (INVELO(i, j) * INVELO(i + 1, j) > 0 And INVELO(i + 1, j) < 0) Then
            KP(i + 1, j) = SKN2(j)
            REST_FORCE(i + 1, j) = (-(X0T(j) - INDISP(i + 1, j)) / (YIELDC(j) - X0T(j))) *
(RUI(j))
          ElseIf (INVELO(i, j) * INVELO(i + 1, j) < 0) Then
            If (SKN1C(j) = 0) Then
              KP(i + 1, j) = SKN1T(j)
            Else
              KP(i + 1, j) = SKN1C(j)
            End If
            If (INDISP(i, j) > INDISP(i + 1, j)) Then
              REST_FORCE(i + 1, j) = (-(X0T(j) - INDISP(i + 1, j)) / (YIELDC(j) - X0T(j))) *
(RUI(j))
              RMI(j) = REST_FORCE(i + 1, j)
              DISPMI(j) = INDISP(i + 1, j)
              X0C(j) = (DISPMI(j) - RMI(j) / KP(i + 1, j))
              N(j) = 1
              L(j) = 2
```

```
            Else
              RMI(j) = REST_FORCE(i, j)
              DISPMI(j) = INDISP(i, j)
              X0C(j) = (DISPMI(j) - RMI(j) / KP(i + 1, j))
              REST_FORCE(i + 1, j) = RMI(j) + (INDISP(i + 1, j) - DISPMI(j)) * KP(i + 1, j)
              N(j) = 1
              L(j) = 2
            End If
          End If
        End If
      End If
    End If
  End If

'Plastic Response in Compression KEY = -1

  If (KEY(i + 1, j) = -1) Then
    L(j) = 2
    If (INVELO(i, j) * INVELO(i + 1, j) < 0# And INVELO(i + 1, j) >= 0#) Then
      If (DELTA_DISP(i, j) > 0#) Then
        DISPMI(j) = INDISP(i, j)
        YIELDC(j) = DISPMI(j)
        SKN1C(j) = (SK(j) / 3) * (YCF(j) / YIELDC(j)) ^ 0.5
        KP(i + 1, j) = SKN1C(j)
        RMI(j) = REST_FORCE(i, j)
        RUI(j) = RMI(j)
        RUA(j) = -RUI(j)
        X0C(j) = (YIELDC(j) - RUI(j) / SKN1C(j))
        REST_FORCE(i + 1, j) = RUI(j) + (INDISP(i + 1, j) - YIELDC(j)) * KP(i + 1, j)
        YIELDT(j) = -YIELDC(j)
        YT(i + 1, j) = YIELDT(j)
        YC(i + 1, j) = YIELDC(j)
        KEY(i + 1, j) = 0
Else
DISPMI(j) = INDISP(i + 1, j)
REST_FORCE(i + 1, j) = RUI(j) + (DISPMI(j) - YIELDC(j)) * (ALPHA * (SK(j) / 3))
YIELDC(j) = DISPMI(j)
```

```
SKN1C(j) = (SK(j) / 3) * (YCF(j) / YIELDC(j)) ^ 0.5
KP(i + 1, j) = SKN1C(j)
RMI(j) = REST_FORCE(i + 1, j)
RUI(j) = RMI(j)
RUA(j) = -RUI(j)
X0C(j) = (YIELDC(j) - RUI(j) / SKN1C(j))
YIELDT(j) = -YIELDC(j)
YT(i + 1, j) = YIELDT(j)
YC(i + 1, j) = YIELDC(j)
End If
Else
KP(i + 1, j) = ALPHA * (SK(j) / 3)
REST_FORCE(i + 1, j) = RUI(j) + (INDISP(i + 1, j) - YIELDC(j)) * (ALPHA * (SK(j) / 3))
YT(i + 1, j) = YIELDT(j)
YC(i + 1, j) = YIELDC(j)
End If
End If

'Plastic Response in Tension KEY = 1

If (KEY(i + 1, j) = 1) Then
L(j) = 1
If (INVELO(i, j) * INVELO(i + 1, j) < 0# And INVELO(i + 1, j) < 0#) Then
If (DELTA_DISP(i, j) < 0#) Then
DISPMA(j) = INDISP(i, j)
YIELDT(j) = DISPMA(j)
SKN1T(j) = (SK(j) / 3) * (YTF(j) / YIELDT(j)) ^ 0.5
KP(i + 1, j) = SKN1T(j)
RMA(j) = REST_FORCE(i, j)
RUA(j) = RMA(j)
RUI(j) = -RUA(j)
X0T(j) = (YIELDT(j) - RUA(j) / SKN1T(j))
REST_FORCE(i + 1, j) = RUA(j) - (YIELDT(j) - INDISP(i + 1, j)) * KP(i + 1, j)
YIELDC(j) = -YIELDT(j)
YT(i + 1, j) = YIELDT(j)
YC(i + 1, j) = YIELDC(j)
KEY(i + 1, j) = 0
```

```
Else
DISPMA(j) = INDISP(i + 1, j)
REST_FORCE(i + 1, j) = RUA(j) + (DISPMA(j) - YIELDT(j)) * (ALPHA * (SK(j) / 3))
YIELDT(j) = DISPMA(j)
SKN1T(j) = (SK(j) / 3) * (YTF(j) / YIELDT(j)) ^ 0.5
KP(i + 1, j) = SKN1T(j)
RMA(j) = REST_FORCE(i + 1, j)
RUA(j) = RMA(j)
RUI(j) = -RUA(j)
X0T(j) = (YIELDT(j) - RUA(j) / SKN1T(j))
YIELDC(j) = -YIELDT(j)
YT(i + 1, j) = YIELDT(j)
YC(i + 1, j) = YIELDC(j)
End If
Else
KP(i + 1, j) = ALPHA * (SK(j) / 3)
REST_FORCE(i + 1, j) = RUA(j) + (INDISP(i + 1, j) - YIELDT(j)) * (ALPHA * (SK(j) / 3))
YT(i + 1, j) = YIELDT(j)
YC(i + 1, j) = YIELDC(j)
End If
End If

Next i
Next j

End Sub

Sub Stress_Strain()

Dim MTX_REG(6000, 10), MTX_TEG(6000, 10)

For ii = 1 To NOM
For iter = 1 To NEQ
MTX_REG(iter, ii) = (MTX_DISP(iter, ii) * 100) / ELV_SOIL(ii)
MTX_TEG(iter, ii) = REST_FORCE(iter, ii) / (100 * 100)
Sheet7.Cells(iter + 2, ii + 20) = MTX_REG(iter, ii)
Sheet7.Cells(iter + 2, ii + 26) = MTX_TEG(iter, ii)
```

```
Next iter
Next ii


End Sub
```