

```

Option Explicit
Option Base 1
Public KVRange As Range, End_Release As Long
Const Pi As Double = 3.14159265358979
=====
Function FEMact(Beams As Variant, Nodes As Variant, DistLoads As Variant, PointLoads As Variant, Moments As Variant) As Variant
    Dim i As Long, Alpha As Double, Beta As Double, NumLoads As Long, FEMA() As Double
    Dim MomInc(1 To 2) As Double, TLoad1 As Double, Tload2 As Double, NumBeams As Long, L As Double, Theta As Double
    Dim LoadInc(1 To 2) As Double, j As Long, k As Long, n As Long, lastk As Long
    Dim NumRows As Long, Node1 As Long, Node2 As Long, DX As Double, DY As Double

    If TypeName(Beams) = "Range" Then Beams = Beams.Value2
    If TypeName(Nodes) = "Range" Then Nodes = Nodes.Value2
    If TypeName(DistLoads) = "Range" Then DistLoads = DistLoads.Value2
    If TypeName(PointLoads) = "Range" Then PointLoads = PointLoads.Value2
    If TypeName(Moments) = "Range" Then Moments = Moments.Value2

    NumBeams = UBound(Beams)
    NumRows = UBound(DistLoads)

    ReDim FEMA(1 To NumBeams, 1 To 6)

    For n = 1 To NumBeams
        Node1 = Beams(n, 3)
        Node2 = Beams(n, 4)
        DX = Nodes(Node2, 2) - Nodes(Node1, 2)
        DY = Nodes(Node2, 3) - Nodes(Node1, 3)
        L = (DX ^ 2 + DY ^ 2) ^ 0.5
        Theta = WorksheetFunction.Atan2(DX, DY)

        lastk = k
        NumLoads = 0
        If k + 1 <= UBound(DistLoads) Then
            Do While DistLoads(k + 1, 1) = n
                NumLoads = NumLoads + 1
                k = k + 1
            Loop
            If k + 1 > UBound(DistLoads) Then Exit Do
            ' Fixed end moments due to distributed loads
            For i = lastk + 1 To lastk + NumLoads
                Alpha = DistLoads(i, 2) / L
                Beta = DistLoads(i, 5) / L
                TLoad1 = DistLoads(i, 3) * L * (1 - Alpha - Beta)

                If DistLoads(i, 3) = DistLoads(i, 4) Then
                    MomInc(1) = TLoad1 * L * (1 / 12 + (Alpha ^ 3) / 4 _
                        - (Alpha ^ 2) * 5 / 12 + Alpha / 12 - (Alpha ^ 2 * Beta) / 4 _
                        + Alpha * Beta / 6 + (Alpha * Beta ^ 2) / 4 + Beta / 12 _
                        + (Beta) ^ 2 / 12 - (Beta ^ 3) / 4)

                    If Alpha = Beta Then
                        MomInc(2) = -MomInc(1)
                    Else
                        MomInc(2) = -TLoad1 * L * (1 / 12 + (Beta ^ 3) / 4 _
                            - (Beta ^ 2) * 5 / 12 + Beta / 12 - (Beta ^ 2 * Alpha) / 4 _
                            + Beta * Alpha / 6 + (Beta * Alpha ^ 2) / 4 + Alpha / 12 _
                            + (Alpha) ^ 2 / 12 - (Alpha ^ 3) / 4)
                    End If
                Else
                    TLoad1 = TLoad1 / 2
                End If
            Next i
        End For
    Next n
End Function

```

```

Frame4.txt
Tload2 = L * (1 - Alpha - Beta) * DistLoads(i, 4) / 2
MomInc(1) = TLoad1 * L * (1 / 10 + (Alpha ^ 3) * 2 / 5 _
    - (Alpha ^ 2) * 7 / 10 + Alpha / 5 - (Alpha ^ 2 * Beta) * 3 / 10 _
    + Alpha * Beta * 4 / 15 + (Alpha * Beta ^ 2) / 5 _
    + Beta / 30 - (Beta) ^ 2 / 30 - (Beta ^ 3) / 10) _
MomInc(2) = -TLoad1 * L * (1 / 15 - (Alpha ^ 3) * 2 / 5 _
    + (Alpha ^ 2) / 5 + Alpha * 2 / 15 + (Alpha ^ 2 * Beta) * 3 / 10 _
    + Alpha * Beta * 1 / 15 - (Alpha * Beta ^ 2) / 5 - Beta / 30 _
    - (Beta) ^ 2 * 2 / 15 + (Beta ^ 3) / 10)

MomInc(2) = MomInc(2) - Tload2 * L * (1 / 10 + (Beta ^ 3) * 2 / 5 _
    - (Beta ^ 2) * 7 / 10 + Beta / 5 - (Beta ^ 2 * Alpha) * 3 / 10 _
    + Beta * Alpha * 4 / 15 + (Beta * Alpha ^ 2) / 5 + Alpha / 30 _
    - (Alpha) ^ 2 / 30 - (Alpha ^ 3) / 10)

MomInc(1) = MomInc(1) + Tload2 * L * (1 / 15 - (Beta ^ 3) * 2 / 5 _
    + (Beta ^ 2) / 5 + Beta * 2 / 15 + (Beta ^ 2 * Alpha) * 3 / 10 _
    + Beta * Alpha * 1 / 15 - (Beta * Alpha ^ 2) / 5 - Alpha / 30 _
    - (Alpha) ^ 2 * 2 / 15 + (Alpha ^ 3) / 10)

End If

If UCase(DistLoads(i, 6)) = "Y" Then
    If Theta <> 0 Then
        MomInc(1) = MomInc(1) * Cos(Theta)
        MomInc(2) = MomInc(2) * Cos(Theta)
    End If
Else
    MomInc(1) = MomInc(1) * -Sin(Theta)
    MomInc(2) = MomInc(2) * -Sin(Theta)
End If

FEMA(n, 3) = MomInc(1) + FEMA(n, 3)
FEMA(n, 6) = MomInc(2) + FEMA(n, 6)
Next i

' Fixed end moments due to point loads

For i = lastk + 1 To lastk + NumLoads

    Alpha = PointLoads(i, 1) / L
    MomInc(1) = PointLoads(i, 2) * Alpha * (1 - Alpha) ^ 2 * L
    MomInc(2) = -PointLoads(i, 2) * Alpha ^ 2 * (1 - Alpha) * L
    If UCase(PointLoads(i, 3)) = "Y" Then
        FEMA(n, 3) = MomInc(1) * Cos(Theta) + FEMA(n, 3)
        FEMA(n, 6) = MomInc(2) * Cos(Theta) + FEMA(n, 6)
    Else
        FEMA(n, 3) = -MomInc(1) * Sin(Theta) + FEMA(n, 3)
        FEMA(n, 6) = -MomInc(2) * Sin(Theta) + FEMA(n, 6)
    End If
End If
Next i

' Fixed end moments and end reactions due to point moments

For i = lastk + 1 To lastk + NumLoads
    Alpha = Moments(i, 1) / L
    MomInc(1) = Moments(i, 2) * (3 * Alpha - 1) * (Alpha - 1)
    MomInc(2) = -Moments(i, 2) * (2 - 3 * Alpha) * Alpha

```

```

FEMA(n, 3) = MomInc(1) + FEMA(n, 3)
FEMA(n, 6) = MomInc(2) + FEMA(n, 6)
LoadInc(1) = -Moments(i, 2) / L
LoadInc(2) = Moments(i, 2) / L
FEMA(n, 1) = FEMA(n, 1) - LoadInc(1) * Sin(Theta)
FEMA(n, 4) = FEMA(n, 4) - LoadInc(2) * Sin(Theta)
FEMA(n, 2) = FEMA(n, 2) + LoadInc(1) * Cos(Theta)
FEMA(n, 5) = FEMA(n, 5) + LoadInc(2) * Cos(Theta)

Next i

' End reactions due to distributed loads

For i = lastk + 1 To lastk + NumLoads
Alpha = DistLoads(i, 2) / L
Beta = DistLoads(i, 5) / L

TLoad1 = DistLoads(i, 3) * L * (1 - Alpha - Beta)

If DistLoads(i, 3) = DistLoads(i, 4) Then
LoadInc(1) = TLoad1 * (1 - Alpha + Beta) / 2
LoadInc(2) = TLoad1 - LoadInc(1)
Else
TLoad1 = TLoad1 / 2
TLoad2 = L * (1 - Alpha - Beta) * DistLoads(i, 4) / 2
LoadInc(1) = TLoad1 * (2 - 2 * Alpha + Beta) / 3

LoadInc(1) = LoadInc(1) + TLoad2 * (1 + 2 * Beta - Alpha) / 3
LoadInc(2) = (TLoad1 + TLoad2) - LoadInc(1)
End If
If UCase(DistLoads(i, 6)) = "X" Then j = 1 Else j = 2
FEMA(n, j) = LoadInc(1) + FEMA(n, j)
FEMA(n, j + 3) = LoadInc(2) + FEMA(n, j + 3)
Next i

' End reactions due to point loads

For i = lastk + 1 To lastk + NumLoads
Alpha = PointLoads(i, 1) / L
If UCase(PointLoads(i, 3)) = "X" Then j = 1 Else j = 2

LoadInc(1) = PointLoads(i, 2) * (1 - Alpha)
LoadInc(2) = PointLoads(i, 2) - LoadInc(1)
FEMA(n, j) = LoadInc(1) + FEMA(n, j)
FEMA(n, j + 3) = LoadInc(2) + FEMA(n, j + 3)

Next i

' Correct end reactions for resultant end moments
If Sin(Theta) <> 0 Then
FEMA(n, 1) = FEMA(n, 1) - (FEMA(n, 3) + FEMA(n, 6)) * Sin(Theta) / L
FEMA(n, 4) = FEMA(n, 4) + (FEMA(n, 3) + FEMA(n, 6)) * Sin(Theta) / L
End If
If Cos(Theta) <> 0 Then
FEMA(n, 2) = FEMA(n, 2) + (FEMA(n, 3) + FEMA(n, 6)) * Cos(Theta) / L
FEMA(n, 5) = FEMA(n, 5) - (FEMA(n, 3) + FEMA(n, 6)) * Cos(Theta) / L
End If

```

```

        End If
        lastk = k
    Next n
    FEMact = FEMA
End Function

```

```

=====

Sub FormKg()
    Dim BeamA() As Double, KParamA(1 To 7) As Double, PHI As Double, Nodes As Variant, DistLoads As Variant
    Dim PointLoads As Variant, Moments As Variant, NumLoads As Long, iErr As Long
    Dim BeamProps As Variant, BeamCon As Variant, NRest As Variant, NCoord As Variant
    Dim NumBeams As Long, NumFree As Long, TNumFree As Long, KGA() As Double, KGI As Variant
    Dim KBCA() As Double, KGCA() As Double, L As Long, Start As Long, m As Long
    Dim FreeA(1 To 1, 1 To 6) As Long, GFreeA() As Double, i As Long, j As Long, k As Long, gj As Long, gk As Long
    Dim FEMA As Variant, FA() As Double, F_aA() As Double, GNumFree As Long
    Dim DefaA As Variant, DefacA() As Double, NFreeA() As Long, NumNodes As Long, TNumNodes As Long, DNode As Long
    Dim NumFix As Long, NumFixnodes As Long, NumFixA(1 To 3) As Long, NumProps As Long, DefA As Variant, NFG As Variant, React() As
Double
    Dim BeamER As Variant, NumER As Long, NumERBeams As Long, ERBeam As Long
    Dim ER As Long, EREnd As Long, RNode As Long, GBFreea() As Long, LastFree As Long
    Dim Node1 As Long, Node2 As Long, BA As Double, BI As Double, BE As Double, BL As Double
    Dim BSA As Double, BG As Double
    Dim NLabels() As String, BLabels() As String
    Dim DX As Double, DY As Double, PropNum As Long, Theta As Double
    Dim KBAA() As Double, BAA() As Double, RowNum As Long
    Const StiffAct As Double = 100000000#
    ' Resize ranges

    NumBeams = Range("numbeams").Value2
    ' Range("beamnum").value = NumBeams
    NumFixnodes = Range("numrest").Value2
    NumLoads = Range("numloads").Value2
    NumProps = Range("numprops").Value2
    NumER = Range("numer").Value2
    NumERBeams = Range("numberbeams").Value2
    NumNodes = Range("numcoords").Value2
    TNumNodes = NumNodes + NumERBeams ' dummy node for each beam with an end release

    Range("beamcon").Resize(NumBeams, 4).Name = "beamcon"
    If NumERBeams > 0 Then
        Range("beamer").Resize(NumERBeams, 7).Name = "beamer"
    End If
    Range("restraints").Resize(NumFixnodes, 4).Name = "restraints"
    Range("coords").Resize(TNumNodes, 3).Name = "coords"
    Range("beamprop").Resize(NumProps, 6).Name = "beamprop"
    Range("distloads").Resize(NumLoads, 6).Name = "distloads"
    Range("pointloads").Resize(NumLoads, 3).Name = "pointloads"
    Range("moments").Resize(NumLoads, 2).Name = "moments"

    BeamProps = Range("beamprop").Value2
    BeamCon = Range("beamcon").Value2
    NCoord = Range("coords").Value2
    BeamER = Range("BeamER").Value2
    DistLoads = Range("distloads").Value2
    PointLoads = Range("pointloads").Value2
    Moments = Range("moments").Value2

    ReDim BeamA(1 To NumBeams, 1 To 2)
    ReDim KBAA(1 To NumBeams * 6, 1 To 7)

```

```

' Create BeamA, beam properties
For i = 1 To NumBeams
    DX = NCoord(BeamCon(i, 4), 2) - NCoord(BeamCon(i, 3), 2)
    DY = NCoord(BeamCon(i, 4), 3) - NCoord(BeamCon(i, 3), 3)
    BeamA(i, 1) = (DX ^ 2 + DY ^ 2) ^ 0.5
    BeamA(i, 2) = ATn2(DX, DY)
Next i

' Count fixed freedoms
NRest = Range("restraints").Value2

GNumFree = (NumNodes) * 3 + NumER
For i = 1 To NumFixnodes
    For j = 2 To 4
        If (NRest(i, j) = "F") Then
            NumFix = NumFix + 1
            NumFixA(j - 1) = NumFixA(j - 1) + 1
        End If
    Next j
Next i
If NumFixA(1) = 0 Or NumFixA(2) = 0 Then
    iErr = MsgBox("There must be at least one node with restraint in X and Y directions", vbCritical)
    Exit Sub
End If
TNumFree = GNumFree - NumFix

ReDim KGCA(1 To GNumFree, 1 To GNumFree)
ReDim KGA(1 To TNumFree, 1 To TNumFree)
ReDim React(1 To GNumFree, 1 To 1)
ReDim GFreeA(1 To GNumFree, 1 To 2)
ReDim FA(1 To GNumFree, 1 To 1)
ReDim F_aA(1 To TNumFree, 1 To 1)
ReDim DefacA(1 To GNumFree, 1 To 1)
ReDim NFreeA(1 To GNumFree)
ReDim GBFreea(1 To NumBeams, 1 To 6)

' Create dummy nodes for each released beam end

If NumERBeams > 0 Then
    For i = 1 To NumERBeams
        DNode = NumNodes + i
        ERBeam = BeamER(i, 1)
        For j = 2 To 4
            ER = ER + BeamER(i, j)
        Next j
        If ER > 0 Then EREnd = 1 Else EREnd = 2

        NCoord(DNode, 1) = DNode
        RNode = BeamCon(ERBeam, 2 + EREnd)
        NCoord(DNode, 2) = NCoord(RNode, 2)
        NCoord(DNode, 3) = NCoord(RNode, 3)
    Next i
End If
FEMA = FEMact(BeamCon, NCoord, DistLoads, PointLoads, Moments)

' Create array of freedoms for each beam, GBFreeA
For i = 1 To NumBeams
    Node1 = BeamCon(i, 3)

```

```

Node2 = BeamCon(i, 4)
For j = 1 To 3
    GBFreea(i, j) = (Node1 - 1) * 3 + j
Next j
If GBFreea(i, 3) > LastFree Then LastFree = GBFreea(i, 3)
For j = 4 To 6
    GBFreea(i, j) = (Node2 - 1) * 3 + (j - 3)
Next j
If GBFreea(i, 6) > LastFree Then LastFree = GBFreea(i, 6)
Next i

For i = 1 To NumERBeams
    ERBeam = BeamER(i, 1)
    For j = 2 To 7
        ER = BeamER(i, j)
        If ER > 0 Then
            LastFree = LastFree + 1
            GBFreea(ERBeam, j - 1) = LastFree
        End If
    Next j
Next i

' Create global array of active freedoms, GFreeA,
' and array of number of freedoms at each node, NFreeA

k = 1
For i = 1 To NumNodes
    If NRest(k, 1) = i Then
        For j = 1 To 3
            If NRest(k, j + 1) = Empty Or NRest(k, j + 1) <> "F" Then
                L = L + 1
                m = m + 1
                GFreeA(L, 1) = j + (i - 1) * 3
                If NRest(k, j + 1) <> 0 Then GFreeA(L, 2) = NRest(k, j + 1)
            End If
        Next j
        If k < NumFixnodes Then k = k + 1
    Else
        For j = 1 To 3
            L = L + 1
            m = m + 1
            GFreeA(L, 1) = j + (i - 1) * 3
        Next j
    End If
    NFreeA(i) = m
    m = 0
Next i
k = NumNodes * 3
For i = 1 To NumERBeams

    For j = 1 To 6

        If BeamER(i, j + 1) = 1 Then
            k = k + 1
            L = L + 1
            m = m + 1
            GFreeA(L, 1) = k
        End If
    Next j
Next i

```

```

        End If
    Next j

    NFreeA(i + NumNodes) = m
    m = 0
Next i

' Create complete global stiffness matrix
For i = 1 To NumBeams

    For j = 1 To 6
        FreeA(1, j) = GBFreea(i, j)
    Next j

    PropNum = BeamCon(i, 2)
    BA = BeamProps(PropNum, 2)
    BI = BeamProps(PropNum, 3)
    BE = BeamProps(PropNum, 4)
    BSA = BeamProps(PropNum, 5)
    BG = BeamProps(PropNum, 6)
    BL = BeamA(i, 1)
    Theta = BeamA(i, 2)
    KParamA(1) = BE * BA / BL
    KParamA(2) = 12 * BE * BI / (BL ^ 3)
    KParamA(3) = BE * BI / BL
    KParamA(4) = 6 * BE * BI / (BL ^ 2)
    KParamA(5) = Cos(Theta)
    KParamA(6) = Sin(Theta)
    KParamA(7) = KParamA(3)

    If BSA <> 0 And BG <> 0 Then
        PHI = 12 * BE * BI / (BG * BSA * BL ^ 2)
        KParamA(2) = KParamA(2) / (1 + PHI)
        KParamA(3) = KParamA(3) * (1 + PHI / 4) / (1 + PHI)
        KParamA(4) = KParamA(4) / (1 + PHI)
        KParamA(7) = KParamA(7) * (1 - PHI / 2) / (1 + PHI)
    End If

    iErr = FormKL(KParamA, KBCA)

    For j = 1 To 6
        gj = FreeA(1, j)
        For k = 1 To 6
            gk = FreeA(1, k)
            KGCA(gj, gk) = KGCA(gj, gk) + KBCA(j, k)
        Next k
    Next j

    For j = 1 To 6
        RowNum = (i - 1) * 6 + j
        For k = 1 To 6
            KBAA(RowNum, k) = KBCA(j, k)
        Next k
        KBAA(RowNum, 7) = KParamA(j)
    Next j
Next i

```

```

' Extract active global stiffness matrix, KGA, from KGCA
For j = 1 To TNumFree
    gj = GFreeA(j, 1)
    For k = 1 To TNumFree
        gk = GFreeA(k, 1)
        KGA(j, k) = KGCA(gj, gk)
    Next k
    If GFreeA(j, 2) <> 0 Then KGA(j, j) = KGA(j, j) * StifFact
Next j

' Create array of applied end actions, FA
For i = 1 To (NumBeams)
    For j = 1 To 6
        FreeA(1, j) = GBFreeA(i, j)
    Next j

    For j = 1 To 6
        FA(FreeA(1, j), 1) = FA(FreeA(1, j), 1) + FEMA(i, j)
    Next j
Next i

' Extract array of actions applied to active freedoms, F_aA, from FA
For i = 1 To TNumFree
    F_aA(i, 1) = FA(GFreeA(i, 1), 1)
    If GFreeA(i, 2) <> 0 Then F_aA(i, 1) = F_aA(i, 1) + GFreeA(i, 2) * KGA(i, i)
Next i

' Transfer F_aA to range f_a
With Range("f_a")
    .ClearContents
    .Resize(TNumFree, 1).Name = "f_a"
End With
Range("f_a").Value2 = F_aA

' Resize range defa, solve system for active freedoms
With Range("defa")
    .ClearContents
    .Resize(TNumFree, 1).Name = "defa"
End With
Defa = GESolve(KGA, F_aA)
Range("defa").Value = Defa

' Create array of all node deflections, DefacA
DefaA = Range("defa").Value2
For i = 1 To TNumFree
    DefacA(GFreeA(i, 1), 1) = DefaA(i, 1)
Next i

' Transfer DefacA to range defc
With Range("defc")
    .ClearContents
    .Resize(GNumFree, 1).Name = "defc"
End With
Range("defc").Value2 = DefacA

' Create arrays of node and beam labels and transfer to range "nlabels" and "blabels"

```

```

ReDim NLabels(1 To GNumFree, 1)
For i = 1 To NumNodes
    j = (i - 1) * 3
    NLabels(j + 1, 1) = "DX" & i
    NLabels(j + 2, 1) = "DY" & i
    NLabels(j + 3, 1) = "RZ" & i
Next i
j = 0
L = 0

j = NumNodes * 3
For i = 1 To NumERBeams
    For k = 2 To 7
        If BeamER(i, k) = 1 Then
            j = j + 1
            L = L + 1
            NLabels(j, 1) = "Rel" & BeamER(i, 1) & "-" & k - 1
        End If
    Next k
Next i
With Range("nlabels")
    .ClearContents
    .Resize(GNumFree, 1).Name = "nlabels"
End With
Range("nlabels").Value = NLabels

ReDim BLabels(1 To NumBeams * 2, 1)
For i = 1 To NumBeams
    j = (i - 1) * 2
    BLabels(j + 1, 1) = i & "-" & 1
    BLabels(j + 2, 1) = i & "-" & 2
Next i

With Range("blabels")
    .ClearContents
    .Resize(NumBeams * 2, 1).Name = "blabels"
End With
Range("blabels").Value = BLabels

NFG = MMultv(KGCA, DefacA)

' Resize range react, and re-apply array formula "=nfg-fg"
With Range("react")
    .ClearContents
    .Resize(GNumFree, 1).Name = "react"
End With
For i = 1 To GNumFree
    React(i, 1) = NFG(i, 1) - FA(i, 1)
Next i
Range("react").Value = React

' Find beam actions
iErr = BeamActions(KBAA, DefacA, FEMA, GBFreea, BAA)

With Range("beam_act")
    .ClearContents
    .Resize(NumBeams * 2, 3).Name = "beam_act"
End With

```

```

Range("beam_act").Value = BAA

End Sub
=====
Function FormKL(k() As Double, ByRef KB() As Double) As Long
    Dim Csq As Double, Ssq As Double, CS As Double
    Dim i As Long, j As Long

    ReDim KB(1 To 6, 1 To 6)

    Csq = k(5) ^ 2
    Ssq = k(6) ^ 2
    CS = k(5) * k(6)

    KB(1, 1) = Csq * k(1) + Ssq * k(2)
    KB(1, 2) = CS * (k(1) - k(2))
    KB(1, 3) = -k(6) * k(4)
    KB(1, 4) = -KB(1, 1)
    KB(1, 5) = -KB(1, 2)
    KB(1, 6) = KB(1, 3)

    KB(2, 2) = Ssq * k(1) + Csq * k(2)
    KB(2, 3) = k(5) * k(4)
    KB(2, 4) = KB(1, 5)
    KB(2, 5) = -KB(2, 2)
    KB(2, 6) = KB(2, 3)

    KB(3, 3) = 4 * k(3)
    KB(3, 4) = -KB(1, 3)
    KB(3, 5) = -KB(2, 6)
    KB(3, 6) = 2 * k(7)

    KB(4, 4) = KB(1, 1)
    KB(4, 5) = KB(1, 2)
    KB(4, 6) = -KB(1, 6)

    KB(5, 5) = KB(2, 2)
    KB(5, 6) = -KB(2, 6)

    KB(6, 6) = KB(3, 3)

    i = 2
    KB(i, 1) = KB(1, i)

    i = 3
    For j = 1 To 2
        KB(i, j) = KB(j, i)
    Next j

    i = 4
    For j = 1 To 3
        KB(i, j) = KB(j, i)
    Next j

    i = 5
    For j = 1 To 4
        KB(i, j) = KB(j, i)
    Next j

```

```

i = 6
For j = 1 To 5
    KB(i, j) = KB(j, i)
Next j

FormKL = 0

End Function
=====
Function BeamActions(KBAA As Variant, DefacA As Variant, FEMA As Variant, GBFreea As Variant, BAA() As Double) As Long
    Dim NumBeams As Long, i As Long, j As Long, k As Long, FreeNum As Long, NF As Double
    Dim KBA(1 To 6, 1 To 6), BeamNum As Long, DefA(1 To 6, 1 To 1) As Double
    Dim BeamAct As Variant, CT As Double, ST As Double, FX As Double, FY As Double

    NumBeams = UBound(GBFreea) - LBound(GBFreea) + 1
    ReDim BAA(1 To NumBeams * 2, 1 To 3)
    For i = 1 To NumBeams
        BeamNum = (i - 1) * 6
        For j = 1 To 6
            For k = 1 To 6
                KBA(j, k) = KBAA(BeamNum + j, k)
            Next k
            FreeNum = GBFreea(i, j)
            DefA(j, 1) = DefacA(FreeNum, 1)
        Next j
        BeamAct = MMultv(KBA, DefA)
        CT = KBAA(BeamNum + 5, 7)
        ST = KBAA(BeamNum + 6, 7)
        FX = BeamAct(1, 1)
        FY = BeamAct(2, 1)

        BAA((i - 1) * 2 + 1, 1) = -((BeamAct(1, 1) - FEMA(i, 1)) * ST - (BeamAct(2, 1) - FEMA(i, 2)) * CT)
        BAA((i - 1) * 2 + 1, 2) = -((BeamAct(2, 1) - FEMA(i, 2)) * ST + (BeamAct(1, 1) - FEMA(i, 1)) * CT)
        BAA((i - 1) * 2 + 1, 3) = -(BeamAct(3, 1) - FEMA(i, 3))

        BAA((i - 1) * 2 + 2, 1) = ((BeamAct(4, 1) - FEMA(i, 4)) * ST - (BeamAct(5, 1) - FEMA(i, 5)) * CT)
        BAA((i - 1) * 2 + 2, 2) = ((BeamAct(5, 1) - FEMA(i, 5)) * ST + (BeamAct(4, 1) - FEMA(i, 4)) * CT)
        BAA((i - 1) * 2 + 2, 3) = BeamAct(6, 1) - FEMA(i, 6)
    Next i
End Function
=====
Function ATn2(X As Variant, Y As Variant) As Double

    ' Inverse tangent based on X and Y coordinates

    ' X and Y both zero produces an error.

    If X = 0 Then
        If Y = 0 Then
            ATn2 = 1 / 0
        ElseIf Y > 0 Then
            ATn2 = Pi / 2
        Else

```

```

        ATn2 = -Pi / 2
    End If
ElseIf X > 0 Then
    If Y = 0 Then
        ATn2 = 0
    Else
        ATn2 = Atn(Y / X)
    End If
Else
    If Y = 0 Then
        ATn2 = Pi
    Else
        ATn2 = (Pi - Atn(Abs(Y) / Abs(X))) * Sgn(Y)
    End If
End If

End Function
=====

Public Function GESolve(Mat As Variant, Vect As Variant) As Variant
    Dim NumEq As Long, MBound As Long, i As Long, j As Long, k As Long
    Dim Df As Double, SSum As Double, ResultA() As Double

    NumEq = UBound(Mat) - LBound(Mat) + 1
    If NumEq <= 0 Then Exit Function
    MBound = NumEq
    ReDim ResultA(1 To MBound, 1 To 1)

    For k = 1 To MBound - 1
        For i = k + 1 To MBound
            If Abs(Mat(k, k)) < Abs(Mat(i, k)) Then
                Call swapRowMat(i, k, MBound, Mat)
                Call swapRowMat(i, k, 1, Vect)
            End If
            If Mat(k, k) <> 0 Then
                Df = Mat(i, k) / Mat(k, k)
                For j = k To MBound
                    Mat(i, j) = Mat(i, j) - Mat(k, j) * Df
                Next j
                Vect(i, 1) = Vect(i, 1) - Vect(k, 1) * Df
            End If
        Next i
    Next k

```

```

ResultA(MBound, 1) = Vect(MBound, 1) / Mat(MBound, MBound)
For i = MBound - 1 To 1 Step -1
    SSum = 0
    For j = i + 1 To MBound
        SSum = SSum + Mat(i, j) * ResultA(j, 1)
    Next j
    If Mat(i, i) <> 0 Then ResultA(i, 1) = (Vect(i, 1) - SSum) / Mat(i, i)
Next i
GESolve = ResultA

End Function
=====
Private Sub swapRowMat(ByVal r1 As Long, ByVal r2 As Long, MBound As Long, Mat)
    Dim i As Long, mB As Long, dVal As Double

    For i = 1 To MBound
        dVal = Mat(r1, i)
        Mat(r1, i) = Mat(r2, i)
        Mat(r2, i) = dVal
    Next i
End Sub
=====
Function MMultv(Mat1 As Variant, Mat2 As Variant) As Variant
    Dim MatRes() As Double, NumRows As Long, NumCols As Long, i As Long, j As Long
    Dim k As Long

    NumRows = UBound(Mat1) - LBound(Mat1) + 1
    NumCols = UBound(Mat2, 2) - LBound(Mat2, 2) + 1

    ReDim MatRes(1 To NumRows, 1 To NumCols)

    For i = 1 To NumCols
        For j = 1 To NumRows
            For k = 1 To NumRows
                MatRes(j, i) = MatRes(j, i) + Mat1(j, k) * Mat2(k, i)
            Next k
        Next j
    Next i

    MMultv = MatRes
End Function

```

This document was created with Win2PDF available at <http://www.daneprairie.com>.  
The unregistered version of Win2PDF is for evaluation or non-commercial use only.