3-D Beam Finite Element Programming - A Practical Guide
Part 3 – Advanced features
(Software Included)

February 2025

Paolo Varagnolo: freelance engineer - Italy, info@studioingegneriavaragnolo.com
Private Practice

Contents

1 Introduction

The program MdFem has been presented in [9] for the basic static analysis, and in [10] for the modal response spectrum analysis. In this paper some advanced features in static analysis will be presented.

The aim of this paper is to describe some techniques useful in the analysis of frame structures. For each new feature examples will be presented, along with the related code. As in the previous works cited, we want to focus on the programmer's point of view.

The theoretical and mathematical framework will therefore not be addressed, since it is already widely available in many books and papers.

The MdFem program implements a 3-D Beam element. It is a straight, 2 nodes element: at each node there are 3 translational and 3 rotational degrees of freedom (dof). This element is capable of transmitting axial and shear forces, along with torque and bending moments.

The new features described in this paper are:

1) spring elements;
2) release of moments at the element ends;
3) elements capable of only tensile stresses;
4) elements capable of only compressive stresses (gap elements);
5) beams on elastic soil.

Several validation examples will be presented in this paper, comparing MdFem results with the results of Sap4 (in the version of 1994 by Bruce F. Maison, based on the original 1973 Sap4 developed by K. J. Bathe, E. L. Wilson, F. E. Peterson from the University of California, Berkley) and SismiCad (a widly used commercial program by Concrete S.r.L. – Padova – Italy).

2 Spring elements

The springs can be defined for each of the 6 degrees of freedom (dof) of a node. In this paper they are supposed to be elastic. The dof where the spring is applied must not be restrained.

The translational springs behave according to the Hooke's law and their values are expressed as the ratio of a force and the displacement in the direction of the force. The intensity of the translational springs in the x, y, z global directions are therefore:

$$K_x = F_x / u_x$$

$$K_y = F_y / u_y$$

$$K_z = F_z / u_z$$

The concept can be extended to the rotational dof, where the intensity of the springs is defined as the ratio of a moment and the rotation about the same axis.

$$K_{xx} = M_{xx} / \theta_x$$

$$K_{yy} = M_{yy} / \theta_y$$

$$K_{zz} = M_{zz} / \theta_z$$

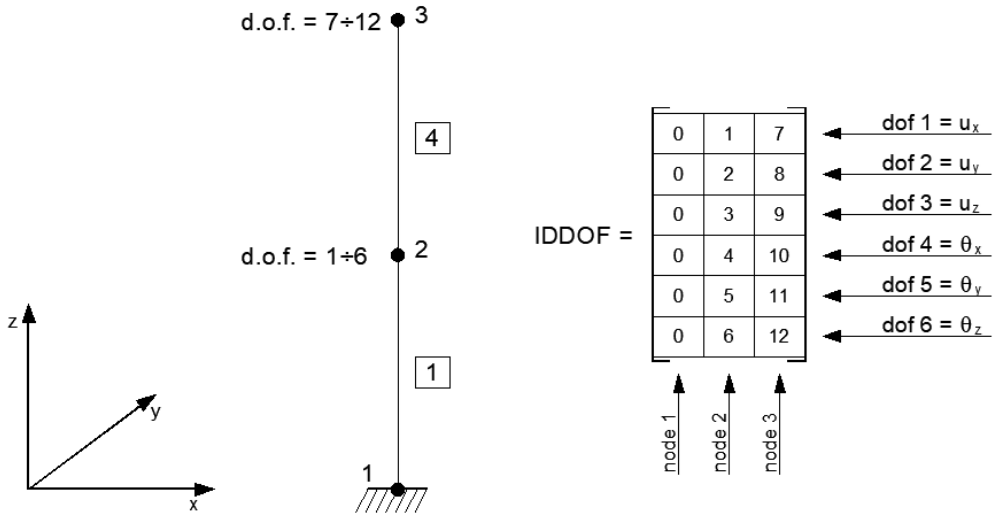
In the MdFem program the spring values are stored in the array ***SPRIN****(Npoin, Ndofn),* where *Npoin* is the number of nodes in the model and *Ndofn* is the number of degrees of freedom (6) at each node.

The effect of the springs is considered by simply adding, in the global stiffness matrix, their stiffness to the degree of freedom on which they act. From [9] we know that the degrees of freedom of the structure are stored in the array ***IDDOF***(*Ndofn, Npoin*), while the addresses of the diagonal elements of the global stiffness matrix are stored in the array ***MAXAD***(*Ndoft*), where *Ndoft* is the total number of dof in the structure.

The global stiffness matrix of the structure $[K_G]$ is stored in compact form in the array ***GLOBK***(*Index*), with the active columns scheme as described in [1], [9].

2.1 Example 2.1

Consider the structure in the next figure, with 2 elements and 3 nodes. Node 1 is fixed and has no degrees of freedom, while at nodes 2, 3 there are 6 + 6 degrees od freedom, 3 displacements and 3 rotations each. After some preliminary elaborations, the array ***IDDOF***(*Ndofn, Npoin*) will contain the values listed below.

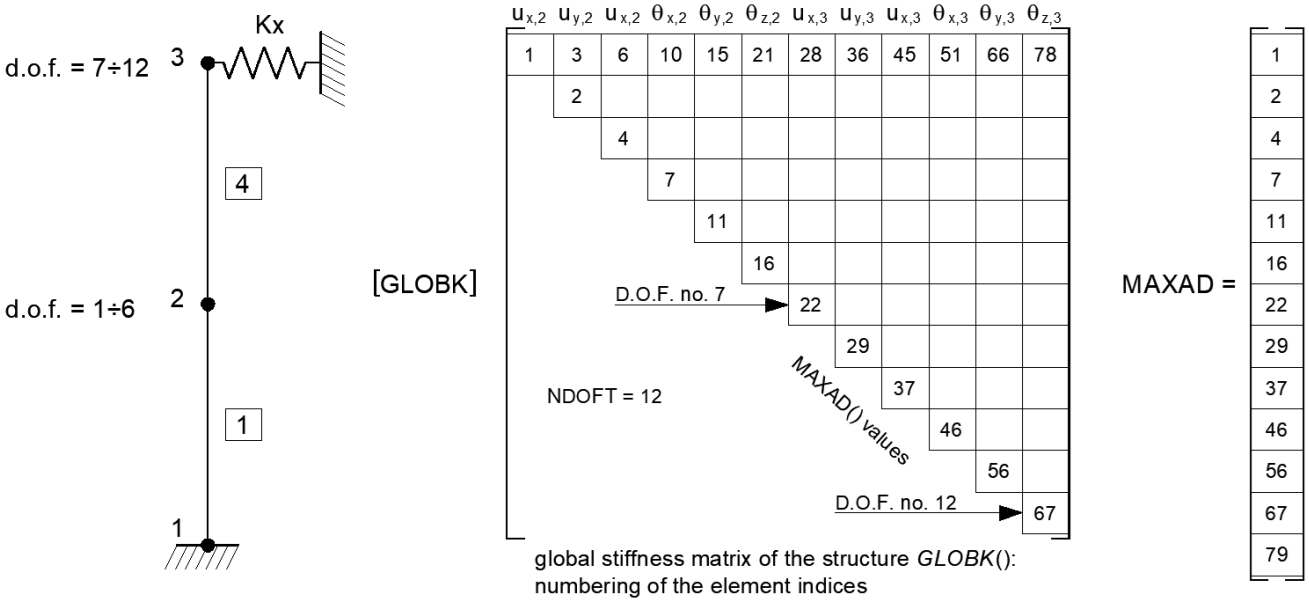


At node 2 there are the 6 dof ($u_{x,2}$, $u_{y,2}$, $u_{z,2}$, $\theta_{x,2}$, $\theta_{y,2}$, $\theta_{z,2}$), and at node 3 there are the 6 dof ($u_{x,3}$, $u_{y,3}$, $u_{z,3}$, $\theta_{x,3}$, $\theta_{y,3}$, $\theta_{z,3}$).

The following figure shows the ***MAXAD()*** array referred to the previous figure, and the numbering of the element indices in the global stiffness matrix ***GLOBK()***.

Let's add a spring with intensity Kx acting in the global x direction, as shown in the next figure. The value Kx is stored in the array ***SPRIN()*** at the position corresponding to node 3 and dof 1 (u_x): ***SPRIN***(*3, 1)* = Kx. The array ***IDDOF***(*1, 3)* indicates that the dof corresponding to dof 1 and node 3 is number 7. The corresponding value of ***MAXAD***(*7)* = 22 and therefore the value Kx will be added to ***GLOBK***(*22)*:

GLOBK(*22*) = ***GLOBK***(*22*) + Kx

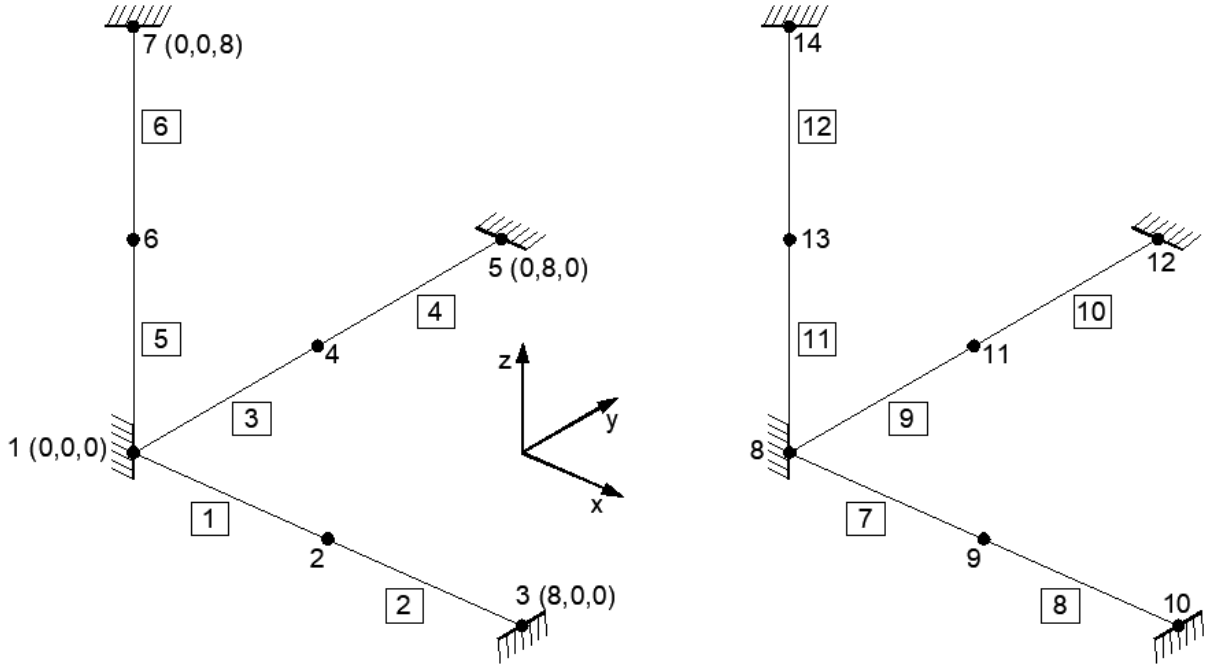


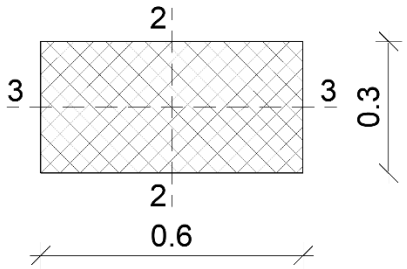
The subroutine used in the program for the assemblage of the springs' stiffnesses is presented below.

```
    Sub ADDSPR()
        Dim Ldofn, Lkglo As Long

        For Ipoin = 1 To Npoin
            For Idofn = 1 To NDOFN
                Ldofn = IDDOF(Idofn, Ipoin)
                Lkglo = MAXAD(Ldofn)
                GLOBK(Lkglo) = GLOBK(Lkglo) + SPRIN(Ipoin, Idofn)
            Next Idofn
        Next Ipoin

    End Sub
```

2.2 Validation example 2.2

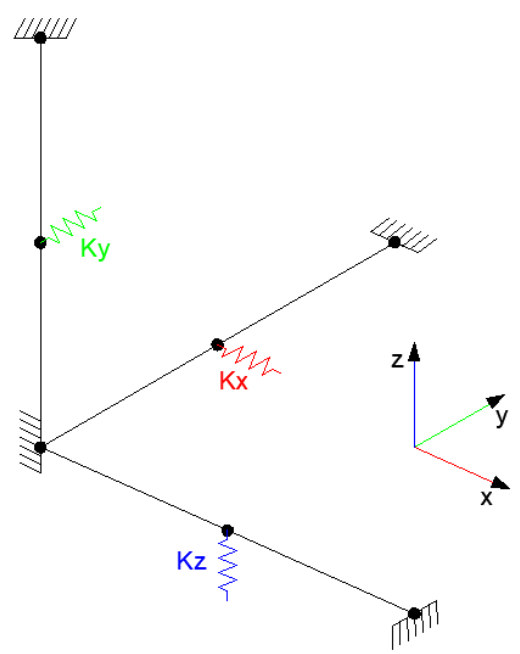
In order to check all the possible spring types and directions, the model shown in the next figures has been calculated. The adopted units are kN, m, rad respectively for forces, lengths and angles. Materials are supposed to be weightless.

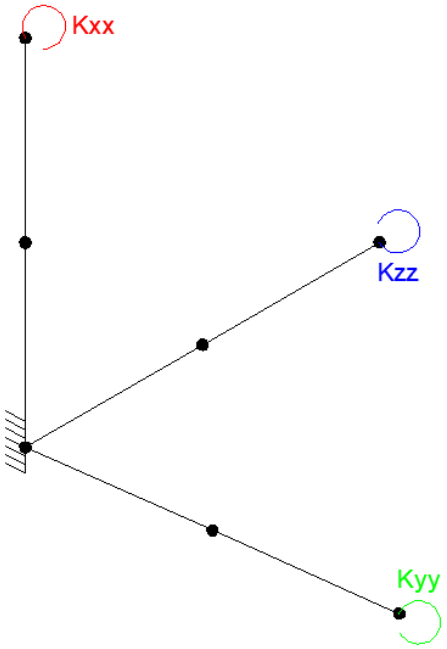


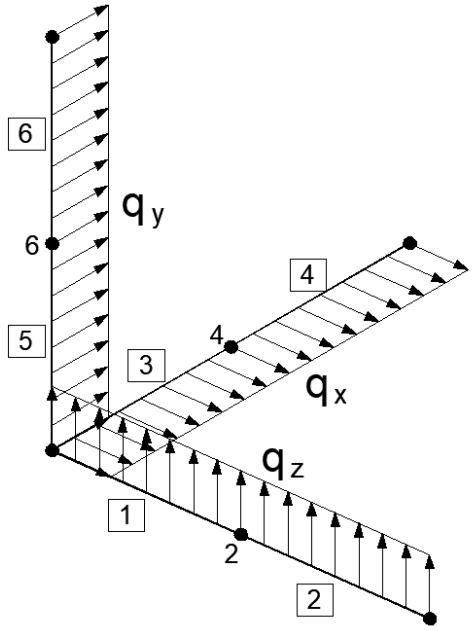


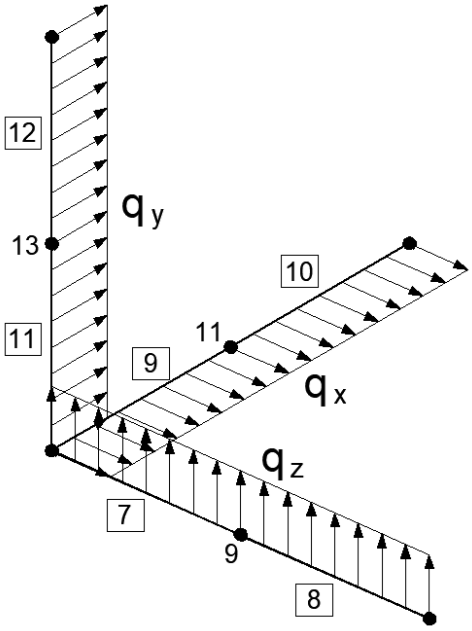
C25/30 concrete – section 0.6x0.3
A = 0.18 m^2
J_{33} = 0.00135 m^4
J_{22} = 0.0054 m^4
J_t = 0.003699 m^4
E = 31447161 kN/m^2
ν = 0.1

In the following figures the springs and the loads are represented. Their values are listed in the next tables.





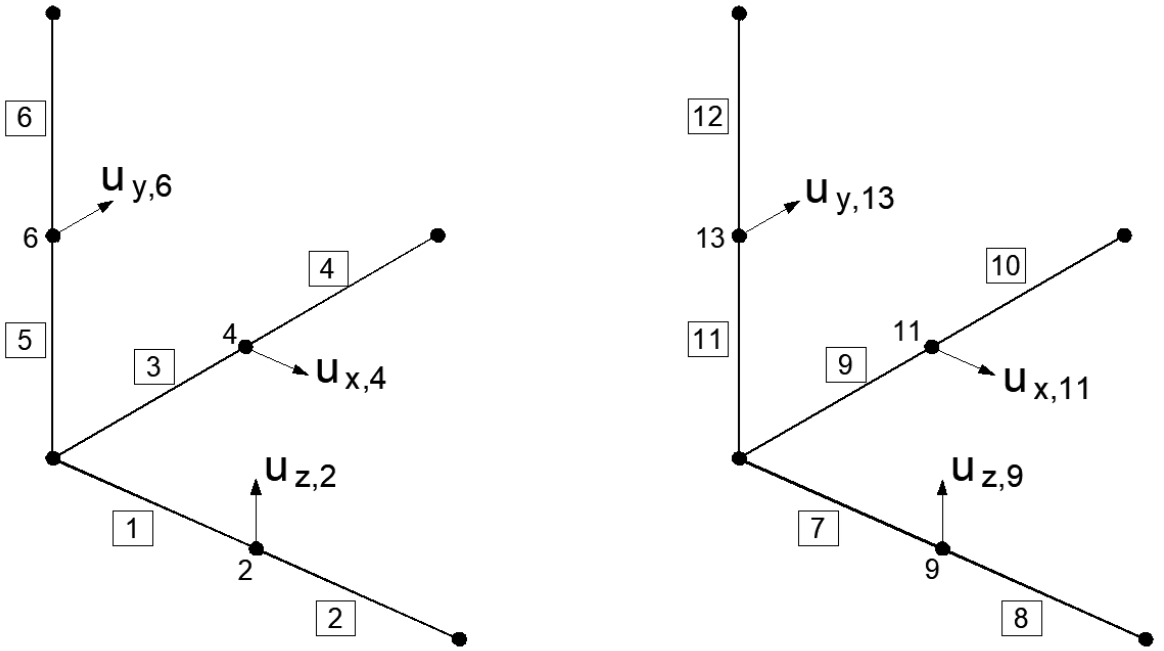


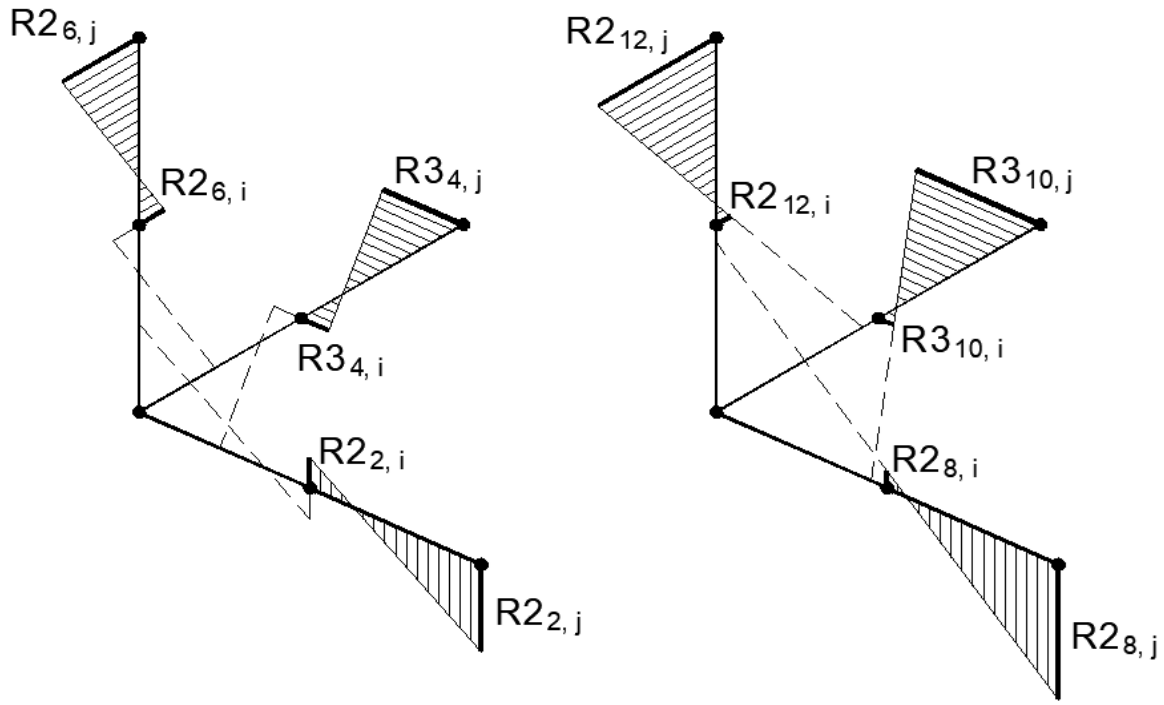


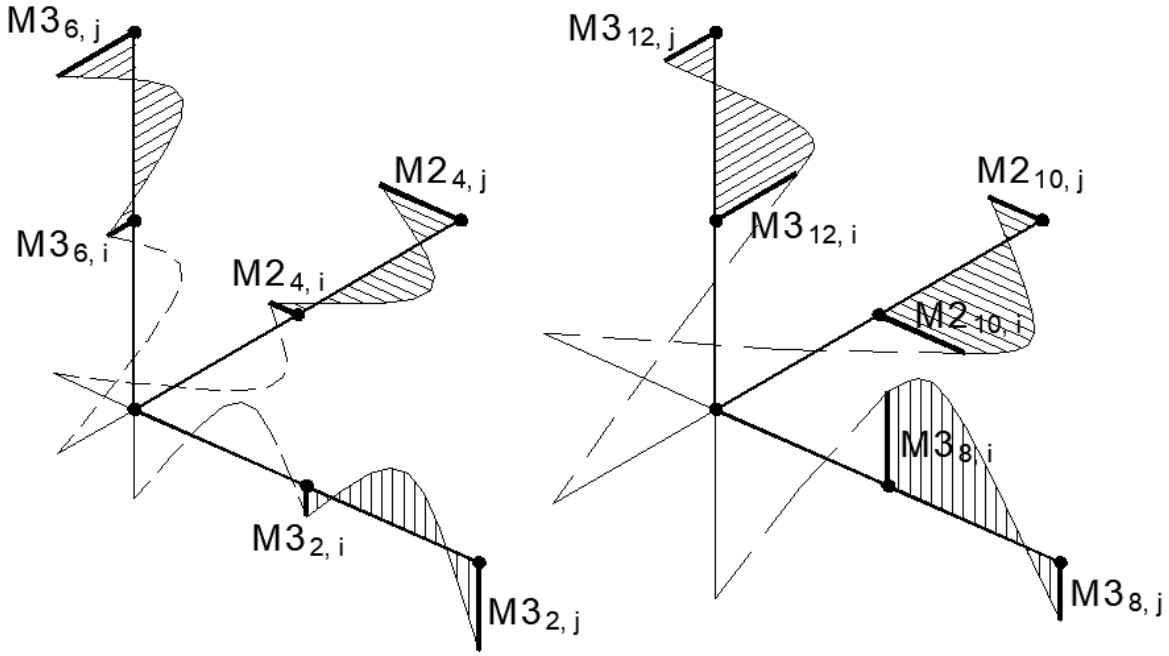
Node	Kx (kN/m)	Ky (kN/m)	Kz (kN/m)	Kxx (kNm/rad)	Kyy (kNm/rad)	Kzz (kNm/rad)
2	0	0	50000	0	0	0
4	50000	0	0	0	0	0
6	0	50000	0	0	0	0
10	0	0	0	0	15000	0
12	0	0	0	0	0	15000
14	0	0	0	15000	0	0

Elements	qx (kN/m)	qy (kN/m)	qz (kN/m)
1, 2, 7, 8	0	0	30
3, 4, 9, 10	30	0	0
5, 6, 11, 12	0	30	0

In the next three figures, some of the main results are highlighted: these results are compared with those obtained with the program SismiCad, for the validation of MdFem.







The geometry, the spring values and the load values are organized in order to obtain the same results in the three global directions. The comparison of the results follows: signs will be omitted in the tables.

Displacements (m)	SismiCad	MdFem	error
$U_{z,2}=U_{x,4}=U_{y,6}$	0.001827	0.0018204	0.36%
$U_{z,9}=U_{x,11}=U_{y,13}$	0.012073	0.011954	0.99%

Shear Forces (kN)	SismiCad	MdFem	error
$R2_{2,i}=R3_{4,i}=R2_{6,i}$	45.67	45.51	0.35%
$R2_{2,j}=R3_{4,j}=R2_{6,j}$	74.33	74.49	0.22%
$R2_{8,i}=R3_{10,i}=R2_{12,i}$	17.54	17.578	0.22%
$R2_{8,j}=R3_{10,j}=R2_{12,j}$	102.46	102.42	0.04%

Bending Moments (kNm)	SismiCad	MdFem	error
$M3_{2,i}= M2_{4,i}= M3_{6,i}$	11.342	11.019	2.8%
$M3_{2,j}= M2_{4,j}= M3_{6,j}$	68.658	68.981	0.47%
$M3_{8,i}= M2_{10,i}= M3_{12,i}$	103.4759	103.440	0.03%
$M3_{8,j}= M2_{10,j}= M3_{12,j}$	66.3571	66.249	0.16%

3 Moments release at the end of the elements

A release in moment is the same as a release in rotation: the only way to ensure rotation continuity from one element to another is to transfer moment between them. Hence a moment end release implies different rotations at the end of concurring elements.

The approach followed by MdFem program to release the moments at the end of the elements, is to add a new dof where required. This technique allows the calculation of the released rotations at the ends of the elements, where the moments are zero.

The possible releases could be technically in any degree of freedom, but the choice has been made to allow only **global rotation releases**. Some kind of translational releases is allowed by means of elements capable of only tensile or only compression forces: these are described further.

Another approach is not to assemble the stiffnesses related to the released rotations, referred to the local coordinate system. This is the technique used by SAP4, however it does not allow to calculate the rotations of the released ends. On the other hand, this technique deals with rotations and moments referred to the local axes, that could be very useful for elements not parallel to the global reference system.

In the MdFem program the released dof are stored in the array ***Releases****(Nelem, NEVAB=2xNdofn),* where *Nelem* is the number of elements in the model and *Ndofn* is the number of degrees of freedom at each node (6 degrees of freedom).

Since only rotational releases are allowed, as a choice, the ***Releases()*** array will refer only to $\theta_{x,i}$, $\theta_{y,i}$, $\theta_{z,i}$, $\theta_{x,j}$, $\theta_{y,j}$, $\theta_{z,j}$. The corresponding indexes will be therefore: ***Releases****(Ielem, 1÷3)* for the three rotations of the first node, and ***Releases****(Ielem, 10÷12)* for the three rotations of the second node.

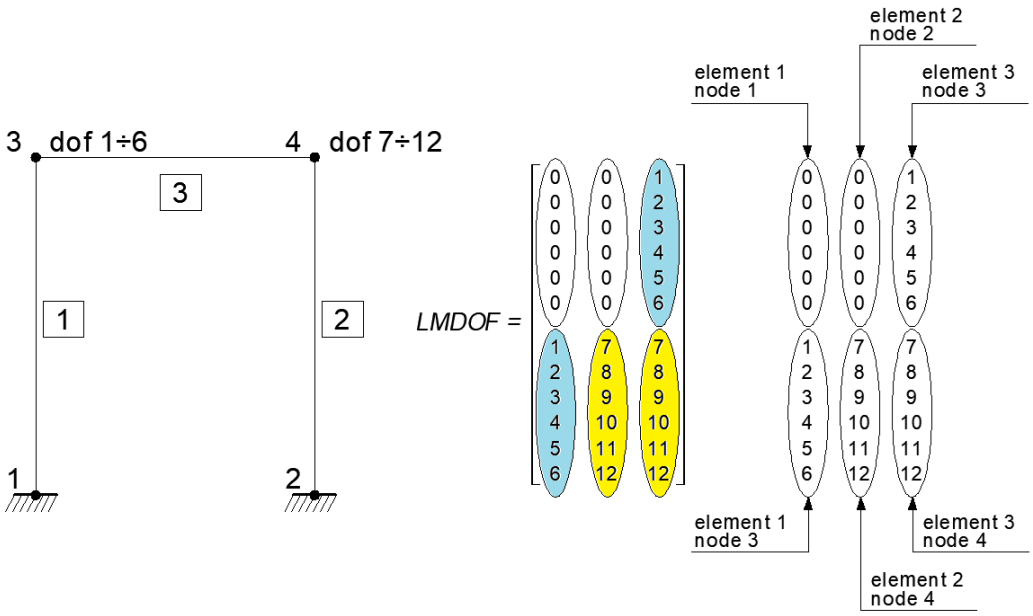
The degrees of freedom of the structure are associated to the elements via the array ***LMDOF****(NEVAB, Nelem)*, which is constructed with the indications contained in the array ***IDDOF()*** introduced in the previous paragraph and in [9]. To insert a free rotation at the end of an element, it is sufficient to define a new dof, in addition to those already identified by the geometry and the constraints of the structure.

Technically the problem is solved like this, but in practice it is also necessary to check that not all the rotations of the elements that converge to a node are released: if it happens, there is a d.o.f. with no stiffness at the node, and the structure is unstable. This check is obtained with the subroutine `Check_Release()` shown later.

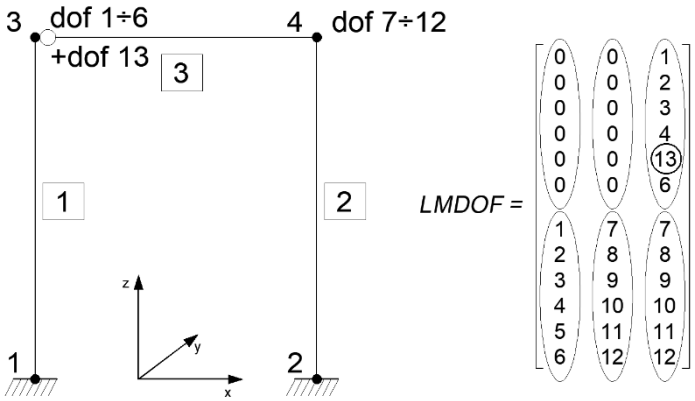
Another precaution is also needed: it is necessary to store the rotations of the free elements in an array, in order to output the calculation results, and also to allow the correct drawing of the deformed structure. The extra rotations are stored in the array ***ElemRotat****(Ncase, Nelem, 1÷6)*, where *Ncase* is the number of load conditions + load combinations, *Nelem* is the number of elements, and the third dimension refers to the six rotations $\theta_{x,i}$, $\theta_{y,i}$, $\theta_{z,i}$, $\theta_{x,j}$, $\theta_{y,j}$, $\theta_{z,j}$.

3.1 Example 3.1

Consider the structure in the next figure, with 3 elements and 4 nodes. There are 6 dof at node no. 3 and 6 dof at node no. 4. The ***LMDOF()*** array contains the indexes of the degrees of freedom at the ends of the elements, and the colors highlight which dof are common between the elements.



By inserting a θ_y hinge at the first (left) end of element no. 3, as shown in the next figure, an extra dof is created, associated to ***LMDOF****(5, 3)*. Therefore the rotations of the elements converging to node no. 3 are no more the same.



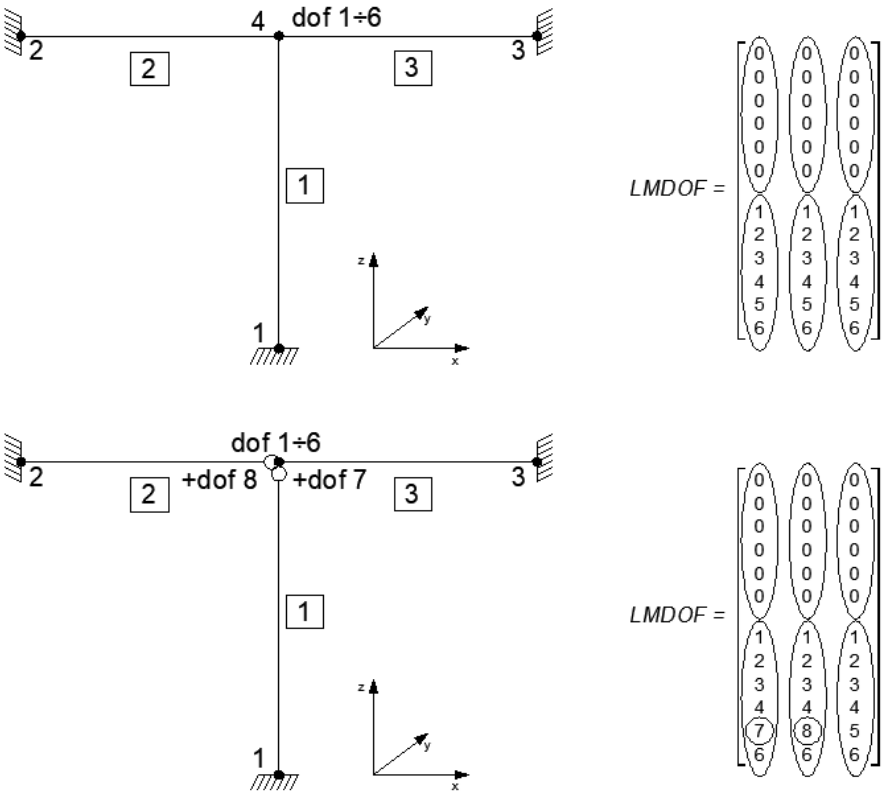
The subroutine used in the program to add the dof corresponding to the released rotations is presented below.

```
Sub MOME_RELEASE()

    ReDim LMDOF(NEVAB, Nelem + Ngaps)

    'Add new d.o.f. if there is some release, and set their number in LMDOF array
    'LMDOF will be completed with normally restrained d.o.f. in INPELE subroutine

    For Ielem = 1 To Nelem
        If Releases(Ielem, 4) = 1 Then
            NDOFT += 1
            LMDOF(4, Ielem) = NDOFT   '4th dof is xxi-rotation
        End If
        If Releases(Ielem, 5) = 1 Then
            NDOFT += 1
            LMDOF(5, Ielem) = NDOFT   '5th dof is yyi-rotation
        End If
        If Releases(Ielem, 6) = 1 Then
            NDOFT += 1
            LMDOF(6, Ielem) = NDOFT   '6th dof is zzi-rotation
```

```
        End If

        If Releases(Ielem, 10) = 1 Then
            NDOFT += 1
            LMDOF(10, Ielem) = NDOFT   '10th dof is xxi-rotation
        End If
        If Releases(Ielem, 11) = 1 Then
            NDOFT += 1
            LMDOF(11, Ielem) = NDOFT   '11th dof is yyi-rotation
        End If
        If Releases(Ielem, 12) = 1 Then
            NDOFT += 1
            LMDOF(12, Ielem) = NDOFT   '12th dof is zzi-rotation
        End If
    Next Ielem

    ReDim Preserve MCOLH(NDOFT)

End Sub
```

3.2 Example 3.2

Consider the structure in the next figure, with 3 elements and 4 nodes. At first there are no hinges (figure above), and the second ends of all the elements refer to the same dof. Then a θ_y hinge is inserted at the second ends of elements no. 1, 2 (figure below): the ***LMDOF()*** array changes as shown in the figure. Dof no. 5 refers to the θ_y rotation of node 4, and therefore after the insertion of the hinges, only to the 11th dof of element 3.



It is not possible to insert another θ_y hinge at the second end of element 3, because in this case equation 5 would have no correlated stiffness, producing an unstable structure. The ***LMDOF()*** array is constructed with the data from the ***IDDOF***() array: to check that there is no instability, it is necessary to check that the dof present in ***IDDOF***() are present at least once in ***LMDOF()***.

The code of the subroutine Check_Release() is presented below: it checks wether the same dof is released at all the ends of the elements convergent to a node.

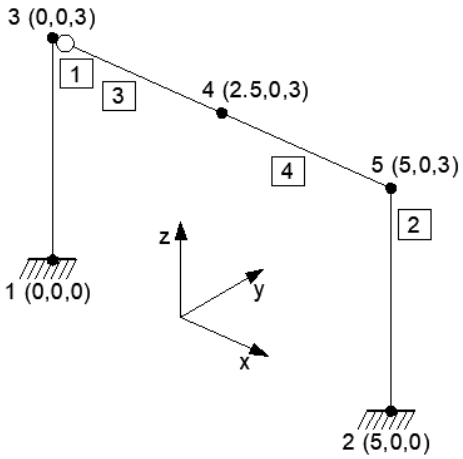
```
 Sub Check_Release(File1 As StreamWriter)

     'check if the same d.o.f. is released in all the elements convergent to a node.
     'If this happens, there is a d.o.f. with no stiffness at the node,
     'and the solver finds the structure unstable.
     'in practice: all the d.o.f. in IDDOF() must be found in LMDOF() at least once

     Dim Unstable, Found As Boolean
     Dim Kdofn, Idof1 As Integer
     Dim Message As String
     Dim Ipoin As Integer = 1

     Unstable = False : Errors = False

     'loop on d.o.f. found in IDDOF(Ndofn, Ipoin)
      For Idofn = 1 To NDOFN
         If Unstable Then Exit For
         For Ipoin = 1 To Npoin
             If Unstable Then Exit For
             Kdofn = IDDOF(Idofn, Ipoin)
              Idof1 = Idofn + NDOFN

             Found = False
             'loop on LMDOF(Idofn, Ielem) elements
             For Ielem = 1 To Nelem
                 If LMDOF(Idofn, Ielem) = Kdofn Then
                     Found = True
                     Exit For
                 End If
                 If LMDOF(Idof1, Ielem) = Kdofn Then
                     Found = True
                     Exit For
                 End If
             Next Ielem
             If Not Found Then
                 If Kdofn <> 0 Then Unstable = True
             End If
         Next Ipoin
     Next Idofn

     If Unstable Then
         Message = "At least one element convergent to node "
         Message += Str$(Ipoin - 1) + "." + vbCrLf
         Message += "must have unreleased rotations." + vbCrLf
         Message += "Data correction necessary to proceed." + vbCrLf
         MsgBox(Message, vbExclamation, "Warning")

         PrtString = "" + vbCrLf
         PrtString += Message
         File1.WriteLine(PrtString)

         Errors = True
     End If

 End Sub
```

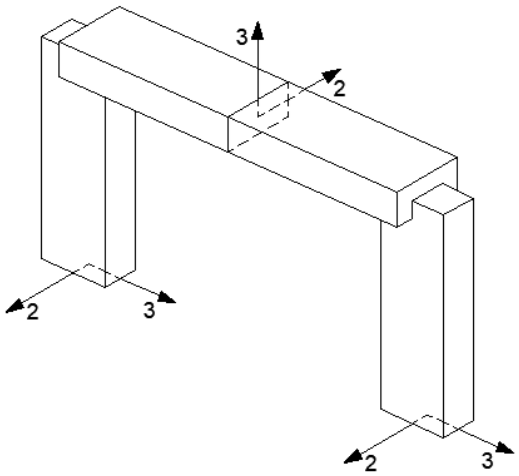
The following code shows the storage of the rotations at the ends of the elements, after the solution of the equilibrium equations in subroutine BACKSU(): the array ***Rload***() contains the calculated displacements (and rotations) of the nodes.

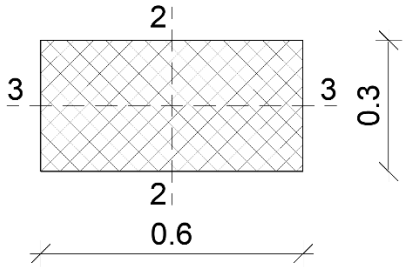
```
        'for BEAM or WINK elements store rotations at the ends of the elements (needed
for deformed shape drawing).
        If Ntype = 2 Or Ntype = 3 Then
            For Ielem = 1 To Nelem
                ElemRotat(Icase, Ielem, 1) = Rload(LMDOF(4, Ielem))
                ElemRotat(Icase, Ielem, 2) = Rload(LMDOF(5, Ielem))
                ElemRotat(Icase, Ielem, 3) = Rload(LMDOF(6, Ielem))
                ElemRotat(Icase, Ielem, 4) = Rload(LMDOF(10, Ielem))
                ElemRotat(Icase, Ielem, 5) = Rload(LMDOF(11, Ielem))
                ElemRotat(Icase, Ielem, 6) = Rload(LMDOF(12, Ielem))
            Next Ielem
        End If
```

3.3 Validation example 3.3

The validation is made with the model shown in the next figures. The cross section is the same of Validation example 2.2. The adopted units are kN, m, rad respectively for forces, lengths and angles. Materials are supposed to be weightless.

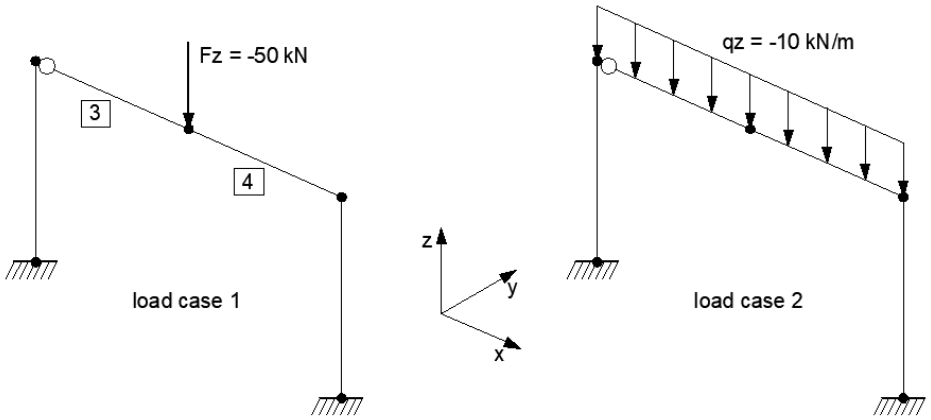




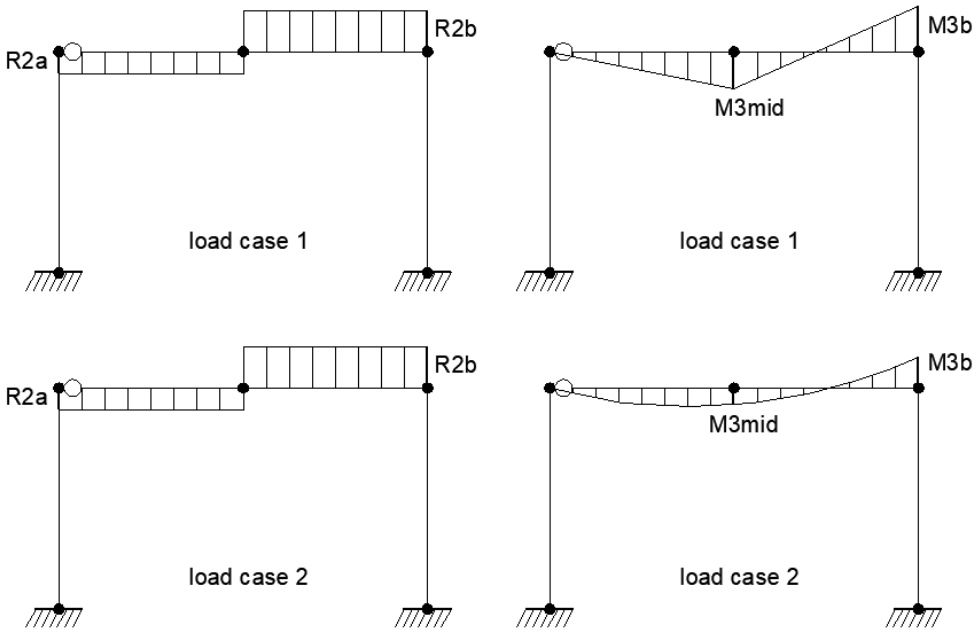


C25/30 concrete – section 0.6x0.3

$A = 0.18\ m^2$
$J_{33} = 0.00135\ m^4$
$J_{22} = 0.0054\ m^4$
$J_t = 0.003699\ m^4$
$E = 31447161\ kN/m^2$
$\nu = 0.1$



The released moment is θ_y at the initial node 3 of element 3. To avoid excessive complexity in the presentation of the comparison of the results, only the example of a θ_y hinge is presented. However, the other possibilities have also been verified, with positive results.
The following figures show the shear forces R2 and the bending moments M3 in the two load cases. The comparison of the results is made with the values indicated in the figures, and is listed below.



Load case 1

Shear Forces (kN)	SismiCad	Sap4	MdFem
R2a	-17.09	-17.7	-17.7
R2b	32.91	-32.3	32.3

Bending moments (kNm)	SismiCad	Sap4	MdFem
M3mid	44.18	44.24	44.24
M3b	-36.64	36.51	-36.51

Load case 2

Shear Forces (kN)	SismiCad	MdFem
R2a	-20.11	-20.13
R2b	29.89	29.87

Bending moments (kNm)	SismiCad	MdFem
M3mid	19.036	19.08
M3b	-24.426	24.34

The results obtained with Sap4 are identical to those of MdFem, and in any case the results obtained with SismiCad are practically the same. Sap 4 doesn't allow the explicit definition of linearly distributed loads, therefore in load case 2 the comparison was limited to SismiCad only.

3.4 Releases in the local coordinate system

As already mentioned at the beginning of the chapter, working in the local coordinate system is an interesting method to apply releases to the ends of an element.

This is the technique used by SAP4, which however has the defect of not calculating the released rotations at the beam ends.

The advantage of this method is the ease of treating beams oriented in space in any way.

The application of this method involves eliminating the contribution of the released degree of freedom, both in the local stiffness matrix $[K_L]$ and in the local equivalent loads vector $[f_L]$.

The processing required to have a generalized force (force or moment) equal to zero, is reported below with reference to the dof *m*.

For any row *i* of the local stiffness matrix:

$$K_{i,j} = K_{i,j} - \frac{K_{m,i}}{K_{m,m}} K_{m,j} \qquad \text{with } j = 1 \div 12 \tag{3.1}$$

The load vector is processed with the following expression:

$$f_i = f_i - \frac{K_{m,i}}{K_{m,m}} f_m \tag{3.2}$$

Although this method is not currently implemented in the MdFem program, the subroutines *ModStiff()* and *ModLoa()* used to (positively) check the above expressions are reported below.

```
    Sub ModStif(ByRef Kmat(,) As Double)
        'Modify stiffness to account for known zero member end forces

        Dim Code, Kd, I1, I2 As Integer
        Dim Sii, Krow(NEVAB) As Double

        'this routine is created, at the moment, only for releasing M3 moment at first node i

        For Inode = 1 To 2
            If Inode = 1 Then
                Code = 1 'that is Code=000001 - here is fixed, only to try the method
            Else
                Code = 0 'that is Code=000000 - no zero end force at node j
                Exit Sub
            End If
            Kd = 100000
            I1 = 6 * (Inode - 1) + 1
            I2 = I1 + 5
            For Idofn = I1 To I2
                If Code < Kd Then
                    Kd /= 10
                Else
```

```
                    Sii = Kmat(Idofn, Idofn)
                    For Jdofn = 1 To NEVAB
                        Krow(Jdofn) = Kmat(Idofn, Jdofn) 'preserve row Idofn
                        'store the ratios between row Idofn values and diagonal element
                        Ratio(Inode, Jdofn) = Kmat(Idofn, Jdofn) / Sii
                    Next Jdofn
                    For Jdofn = 1 To NEVAB 'loop over the matrix rows
                        If Ratio(Inode, Jdofn) <> 0 Then
                            For Ldofn = 1 To NEVAB 'elaborate row Jdofn
                                Kmat(Jdofn, Ldofn) -= Ratio(Inode, Jdofn) * Krow(Ldofn)
                            Next Ldofn
                        End If
                    Next Jdofn
                    Code -= Kd
                    Kd /= 10
                End If
            Next Idofn
        Next Inode

    End Sub
```

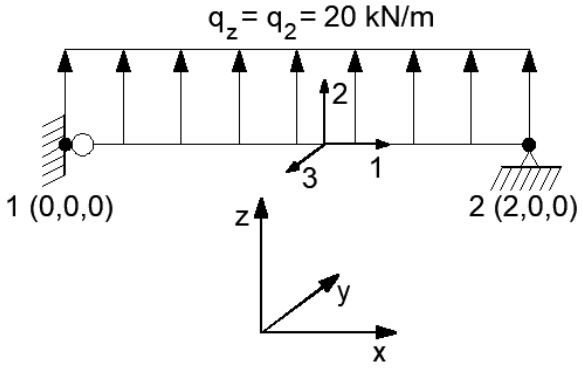
```
    Sub ModLoa(ByRef LocLoa() As Double)
        'Modify local loads to account for known zero member end forces

        Dim Code, Kd, I1, I2 As Integer
        Dim Sfi As Double

        'this routine is created, at the moment, only for releasing M3 moment at first node i

        For Inode = 1 To 2
            If Inode = 1 Then
                Code = 1 'that is Code=000001 - here is fixed, only to try the method
            Else
                Code = 0 'that is Code=000000 - no zero end force at node j
                Exit Sub
            End If
            Kd = 100000
            I1 = 6 * (Inode - 1) + 1
            I2 = I1 + 5
            For Idofn = I1 To I2
                If Code < Kd Then
                    Kd /= 10
                Else
                    Sfi = LocLoa(Idofn)
                    For Jdofn = 1 To NEVAB 'loop over the local load array rows
                        If Ratio(Inode, Jdofn) <> 0 Then
                            'elaborate row Jdofn
                            LocLoa(Jdofn) -= Ratio(Inode, Jdofn) * Sfi
                        End If
                    Next Jdofn
                    Code -= Kd
                    Kd /= 10
                End If
            Next Idofn
        Next Inode

    End Sub
```

3.5 Example 3.4

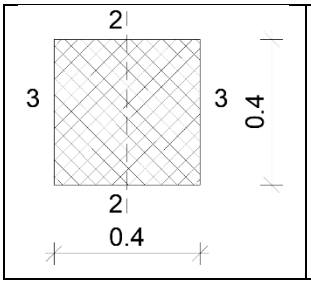
A simple application example of the presented method is given below. It is a beam with a fixed left end and a pinned right end.

The only degree of freedom in this model is θ_y rotation at node 2. If we insert a θ_3 hinge at the first end of the element, as shown in the figure below, with this method the ***LMDOF()*** array doesn't change. What changes is the local stiffness matrix and the equivalent load array of the element.

For this very simple case it is possible to perform the entire calculation by hand, to verify the proposed method. The adopted units are kN, m respectively for forces and lengths. Materials are supposed to be weightless.



$$LMDOF = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} u_{x,1} \\ u_{y,1} \\ u_{z,1} \\ \theta_{x,1} \\ \theta_{y,1} \\ \theta_{z,1} \\ u_{x,2} \\ u_{y,2} \\ u_{z,2} \\ \theta_{x,2} \\ \theta_{y,2} \\ \theta_{z,2} \end{matrix}$$



C25/30 concrete – section 0.4x0.4
A = 0.16 m^2
J_{33} = 0.002133 m^4
J_{22} = 0.002133 m^4
J_t = 0.00315733 m^4
E = 31447161 kN/m^2
ν = 0.1

As shown in [9], the stiffness matrix of the element is:

$$[K_L] = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EJ_3}{L^3} & 0 & 0 & 0 & \frac{6EJ_3}{L^2} & 0 & -\frac{12EJ_3}{L^3} & 0 & 0 & 0 & \frac{6EJ_3}{L^2} \\ 0 & 0 & \frac{12EJ_2}{L^3} & 0 & -\frac{6EJ_2}{L^2} & 0 & 0 & 0 & -\frac{12EJ_2}{L^3} & 0 & -\frac{6EJ_2}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ_1}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ_1}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EJ_2}{L^2} & 0 & \frac{4EJ_2}{L} & 0 & 0 & 0 & \frac{6EJ_2}{L^2} & 0 & \frac{2EJ_2}{L} & 0 \\ 0 & \frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{4EJ_3}{L} & 0 & -\frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{2EJ_3}{L} \\ -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EJ_3}{L^3} & 0 & 0 & 0 & -\frac{6EJ_3}{L^2} & 0 & \frac{12EJ_3}{L^3} & 0 & 0 & 0 & -\frac{6EJ_3}{L^2} \\ 0 & 0 & -\frac{12EJ_2}{L^3} & 0 & \frac{6EJ_2}{L^2} & 0 & 0 & 0 & \frac{12EJ_2}{L^3} & 0 & \frac{6EJ_2}{L^2} & 0 \\ 0 & 0 & 0 & -\frac{GJ_1}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ_1}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EJ_2}{L^2} & 0 & \frac{2EJ_2}{L} & 0 & 0 & 0 & \frac{6EJ_2}{L^2} & 0 & \frac{4EJ_2}{L} & 0 \\ 0 & \frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{2EJ_3}{L} & 0 & -\frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{4EJ_3}{L} \end{bmatrix}$$

Using the data shown above, and neglecting the decimal places, the matrix becomes:

$$\begin{bmatrix} 2515772 & 0 & 0 & 0 & 0 & 0 & -2515772 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100615 & 0 & 0 & 0 & 100615 & 0 & -100615 & 0 & 0 & 0 & 100615 \\ 0 & 0 & 100615 & 0 & -100615 & 0 & 0 & 0 & -100615 & 0 & -100615 & 0 \\ 0 & 0 & 0 & 22565 & 0 & 0 & 0 & 0 & 0 & -22565 & 0 & 0 \\ 0 & 0 & -100615 & 0 & 134153 & 0 & 0 & 0 & 100615 & 0 & 67076 & 0 \\ 0 & 100615 & 0 & 0 & 0 & 134153 & 0 & -100615 & 0 & 0 & 0 & 67076 \\ -2515772 & 0 & 0 & 0 & 0 & 0 & 2515772 & 0 & 0 & 0 & 0 & 0 \\ 0 & -100615 & 0 & 0 & 0 & -100615 & 0 & 100615 & 0 & 0 & 0 & -100615 \\ 0 & 0 & -100615 & 0 & 100615 & 0 & 0 & 0 & 100615 & 0 & 100615 & 0 \\ 0 & 0 & 0 & -22565 & 0 & 0 & 0 & 0 & 0 & 22565 & 0 & 0 \\ 0 & 0 & -100615 & 0 & 67076 & 0 & 0 & 0 & 100615 & 0 & 134153 & 0 \\ 0 & 100615 & 0 & 0 & 0 & 67076 & 0 & -100615 & 0 & 0 & 0 & 134153 \end{bmatrix}$$

As shown in [9], the equivalent loads in local coordinates are:

$$[f_L]^T = \left[\frac{q_1 l}{2} \quad \frac{q_2 l}{2} \quad \frac{q_3 l}{2} \quad 0 \quad -\frac{q_3 l^2}{12} \quad \frac{q_2 l^2}{12} \quad \frac{q_1 l}{2} \quad \frac{q_2 l}{2} \quad \frac{q_3 l}{2} \quad 0 \quad \frac{q_3 l^2}{12} \quad -\frac{q_2 l^2}{12}\right]$$

And, inserting the data, the array becomes:

$$[0 \quad 20 \quad 0 \quad 0 \quad 0 \quad 6.666 \quad 0 \quad 20 \quad 0 \quad 0 \quad 0 \quad -6.666]$$

The release of θ_3 rotation at the first end of the element, corresponds to the release of the 6th dof of the element. In this case the expressions (3.1) and (3.2) become:

$$K_{i,j} = K_{i,j} - \frac{K_{6,i}}{K_{6,6}} K_{6,j} \qquad \text{with } j = 1 \div 12 \qquad (3.3)$$

$$f_i = f_i - \frac{K_{6,i}}{K_{6,6}} f_6 \qquad (3.4)$$

Let's calculate the transformed stiffness matrix of the element, starting from row 1 to row 12.

All the rows *i* where $K_{6,i} = 0$ remain unchanged. The only rows involved in the transformation are those where $K_{6,i}/K_{6,6} \neq 0$: these are rows 2, 6, 8, 12. Furthermore, in each row, the columns involved in the transformations are only those where $K_{6,j} \neq 0$: these are (again) columns 2, 6, 8, 12.

row = 2: $K_{6,2}/K_{6,6} = 100615/134153 = 0.75 = R$

- **j=2** (column 2) $K_{2,2} = K_{2,2} - R \cdot K_{6,2} = 100615 - 0.75 \cdot 100615 = 25153$
- **j=6** (column 6) $K_{2,6} = K_{2,6} - R \cdot K_{6,6} = 100615 - 0.75 \cdot 134153 = 0$
- **j=8** (column 8) $K_{2,8} = K_{2,8} - R \cdot K_{6,8} = -100615 + 0.75 \cdot 100615 = -25153$
- **j=12** (column 12) $K_{2,12} = K_{2,12} - R \cdot K_{6,12} = 100615 - 0.75 \cdot 67076 = 50307$

row = 6: $K_{6,6}/K_{6,6} = 1 = R$

- **j=2** (column 2) $K_{6,2} = K_{6,2} - R \cdot K_{6,2} = 100615 - 1 \cdot 100615 = 0$
- **j=6** (column 6) $K_{6,6} = K_{6,6} - R \cdot K_{6,6} = 134153 - 1 \cdot 134153 = 0$
- **j=8** (column 8) $K_{6,8} = K_{6,8} - R \cdot K_{6,8} = -100615 + 1 \cdot 100615 = 0$
- **j=12** (column 12) $K_{6,12} = K_{6,12} - R \cdot K_{6,12} = 67076 - 1 \cdot 67076 = 0$

The row corresponding to the dof to be released contains now all zeroes, as expected.

row = 8: $K_{6,8}/K_{6,6} = -100615/134153 = -0.75 = R$

- **j=2** (column 2) $K_{8,2} = K_{8,2} - R \cdot K_{6,2} = -100615 + 0.75 \cdot 100615 = -25153$
- **j=6** (column 6) $K_{8,6} = K_{8,6} - R \cdot K_{6,6} = -100615 + 0.75 \cdot 134153 = 0$
- **j=8** (column 8) $K_{8,8} = K_{8,8} - R \cdot K_{6,8} = 100615 - 0.75 \cdot 100615 = 25153$
- **j=12** (column 12) $K_{8,12} = K_{8,12} - R \cdot K_{6,12} = -100615 + 0.75 \cdot 67076 = -50307$

row = 12: $K_{6,12}/K_{6,6} = 67076/134153 = 0.5 = R$

j=2 (column 2) $K_{12,2} = K_{12,2} - R \cdot K_{6,2} = 100615 - 0.5 \cdot 100615 = 50307$

j=6 (column 6) $K_{12,6} = K_{12,6} - R \cdot K_{6,6} = 67076 - 0.5 \cdot 134153 = 0$

j=8 (column 8) $K_{12,8} = K_{12,8} - R \cdot K_{6,8} = -100615 + 0.5 \cdot 100615 = -50307$

j=12 (column 12) $K_{12,12} = K_{12,12} - R \cdot K_{6,12} = 134153 - 0.5 \cdot 67076 = 100615$

The transformed local stiffness matrix becomes:

$$[K_L] = \begin{bmatrix} 2515772 & 0 & 0 & 0 & 0 & 0 & -2515772 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25153 & 0 & 0 & 0 & 0 & 0 & -25153 & 0 & 0 & 0 & 50307 \\ 0 & 0 & 100615 & 0 & -100615 & 0 & 0 & 0 & -100615 & 0 & -100615 & 0 \\ 0 & 0 & 0 & 22565 & 0 & 0 & 0 & 0 & 0 & -22565 & 0 & 0 \\ 0 & 0 & -100615 & 0 & 134153 & 0 & 0 & 0 & 100615 & 0 & 67076 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2515772 & 0 & 0 & 0 & 0 & 0 & 2515772 & 0 & 0 & 0 & 0 & 0 \\ 0 & -25153 & 0 & 0 & 0 & 0 & 0 & 25153 & 0 & 0 & 0 & -50307 \\ 0 & 0 & -100615 & 0 & 100615 & 0 & 0 & 0 & 100615 & 0 & 100615 & 0 \\ 0 & 0 & 0 & -22565 & 0 & 0 & 0 & 0 & 0 & 22565 & 0 & 0 \\ 0 & 0 & -100615 & 0 & 67076 & 0 & 0 & 0 & 100615 & 0 & 134153 & 0 \\ 0 & 50307 & 0 & 0 & 0 & 0 & 0 & -50307 & 0 & 0 & 0 & 100615 \end{bmatrix}$$

Now, applying expression (3.4), we calculate the transformation of the equivalent load vector, starting from row 1 to row 12. In this case too, the rows *i* for which $K_{6,i} = 0$ remain unchanged. The only rows involved in the transformation are those where $K_{6,i}/K_{6,6} \neq 0$: these are rows 2, 6, 8, 12.

row = 2: $K_{6,2}/K_{6,6} = 100615/134153 = 0.75 = R$

$f_2 = f_2 - R \cdot \mathrm{f}_6 = 20 - 0.75 \cdot 6.666 = 15$

row = 6: $K_{6,6}/K_{6,6} = 1 = R$

$f_6 = f_6 - R \cdot \mathrm{f}_6 = 6.666 - 1 \cdot 6.666 = 0$

The row corresponding to the dof to be released contains now a zero, as expected.

row = 8: $K_{6,8}/K_{6,6} = -100615/134153 = -0.75 = R$

$f_8 = f_8 - R \cdot \mathrm{f}_6 = 20 + 0.75 \cdot 6.666 = 25$

row = 12: $K_{6,12}/K_{6,6} = 67076/134153 = 0.5 = R$

$f_{12} = f_{12} - R \cdot \mathrm{f}_6 = -6.666 - 0.5 \cdot 6.666 = -10$

The equivalent loads array in local coordinates becomes:

$$[f_L]^T = [0 \quad 15 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 25 \quad 0 \quad 0 \quad 0 \quad -10] \tag{3.5}$$

The next step consists in transforming the stiffness matrix and the equivalent load vector from the local to the global reference system.

In practice, with reference to the previous figure: the elements referred to the local axis 1 become referred to the global x-axis; the elements referred to the local axis 2 become referred to the global z-axis; the elements referred to the local axis 3 become referred to the global y-axis with a change of sign. Without performing all the steps, the stiffness matrix in the global system becomes:

$$[K_G] = \begin{bmatrix} 2515772 & 0 & 0 & 0 & 0 & 0 & -2515772 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100615 & 0 & 0 & 0 & 100615 & 0 & -100615 & 0 & 0 & 0 & 100615 \\ 0 & 0 & 25153 & 0 & 0 & 0 & 0 & 0 & -25153 & 0 & -50307 & 0 \\ 0 & 0 & 0 & 22565 & 0 & 0 & 0 & 0 & 0 & -22565 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100615 & 0 & 0 & 0 & 134153 & 0 & -100615 & 0 & 0 & 0 & 67076 \\ -2515772 & 0 & 0 & 0 & 0 & 0 & 2515772 & 0 & 0 & 0 & 0 & 0 \\ 0 & -100615 & 0 & 0 & 0 & -100615 & 0 & 100615 & 0 & 0 & 0 & -100615 \\ 0 & 0 & -25153 & 0 & 0 & 0 & 0 & 0 & 25153 & 0 & 50307 & 0 \\ 0 & 0 & 0 & -22565 & 0 & 0 & 0 & 0 & 0 & 22565 & 0 & 0 \\ 0 & 0 & -50307 & 0 & 0 & 0 & 0 & 0 & 50307 & 0 & 100615 & 0 \\ 0 & 100615 & 0 & 0 & 0 & 67076 & 0 & -100615 & 0 & 0 & 0 & 134153 \end{bmatrix}$$

The vector of equivalent loads must also be transported into the global reference system, and becomes:

$$[f_G]^T = [0 \quad 0 \quad 15 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 25 \quad 0 \quad 10 \quad 0]$$

The only dof of the structure is the rotation $\theta_{y,2}$ of node 2 around the y-axis, so the assembly of the stiffness matrix is reduced to the single element $K(11,11)$. Similarly, the assembly of the load vector is reduced to the single element $f(11)$.

The system of equations to be solved is simply:

$$100615 \cdot \theta_{y,2} = 10$$

From which:

$$\theta_{y,2} = \frac{10}{100615} = 9.9389^{-5}$$

This value of $\theta_{y,2}$ in the global coordinate system corresponds to $-\theta_{3,j}$ in the local coordinate system.

By transforming the displacements (they are all equal to zero except $\theta_{3,j}$) from the global to the local system, we obtain:

$$[u_L]^T = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -9.9389^{-5}]$$

Now let's calculate the internal forces $[f_L]$ of the element with the fundamental expression:

$$[K_L][u_L] = [f_L]$$

The product is simple because in the vector $[u_L]$ the only element different from zero is the 12th. Then, for each row *i*, the forces result from the product of the element $K_L(i,12)$ by the rotation $u_L(12)$.

$f_L(1) = K_L(1,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(2) = K_L(2,12) \cdot u_L(12) = 50307 \cdot -9.9389^{-5} = -5$

$f_L(3) = K_L(3,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(4) = K_L(4,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(5) = K_L(5,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(6) = K_L(6,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(7) = K_L(7,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(8) = K_L(8,12) \cdot u_L(12) = -50307 \cdot -9.9389^{-5} = 5$

$f_L(9) = K_L(9,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(10) = K_L(10,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(11) = K_L(11,12) \cdot u_L(12) = 0 \cdot -9.9389^{-5} = 0$

$f_L(12) = K_L(12,12) \cdot u_L(12) = 100615 \cdot -9.9389^{-5} = -10$

The last steps, as explained in [9], are the subtraction of the equivalent loads in (3.5), and the change of the signs of the forces at node *i*.

$f_L(1) = -(0-0) = 0$
$f_L(2) = -(-5-15) = 20$
$f_L(3) = -(0-0) = 0$
$f_L(4) = -(0-0) = 0$
$f_L(5) = -(0-0) = 0$
$f_L(6) = -(0-0) = 0$
$f_L(7) = 0-0 = 0$
$f_L(8) = 5-25 = -20$
$f_L(9) = 0-0 = 0$
$f_L(10) = 0-0 = 0$
$f_L(11) = 0-0 = 0$
$f_L(12) = -10-(-10) = 0$

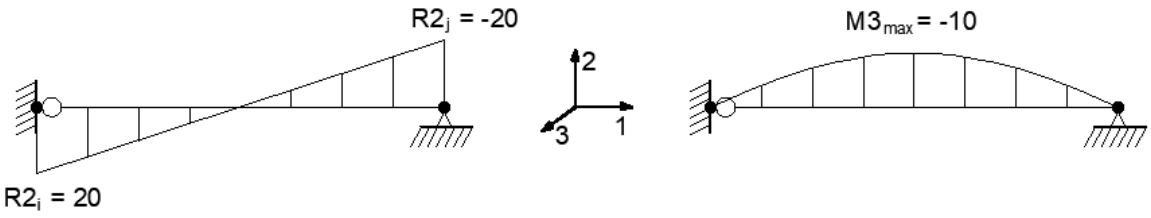
For a better reading, the vector of internal forces referred to the local system is reported below:

$$[f_L]^T = [0 \quad 20 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -20 \quad 0 \quad 0 \quad 0 \quad 0]$$

The calculated values correspond to the R2 forces at the initial and final nodes of the element. All the other forces are zero at the ends of the element.

It's easy to calculate the bending moment M3 at the middle of the element:

$$M3 = -20 \cdot \frac{L}{2} + q \cdot \frac{L^2}{2} = -20 \cdot 1 + 20 \cdot \frac{1^2}{2} = -10$$



The values of shear forces and bending moment are those expected.

With this method, no information is obtained about the amount of the released displacement.

4 Elements capable of only tensile stresses

The elements that can react only to tensile stresses are hereinafter called "tension elements".

The management of "tension elements" requires an iterative procedure: if one of these elements is subject to compressive stresses, it is eliminated by assigning it a material with zero area and very low flexural characteristics.

The iterations continue until there is no "tension element" with compressive stresses.

The direct elimination of the "tension elements" subject to compression could perhaps lead in some cases to incorrect results, even if in the tested examples it never happened.

A more conservative approach, adopted in the MdFem program for gap elements, would consist in the reduction of the geometric-mechanical characteristics of the tension elements subjected to compression, at each iteration.

This change could be easily implemented, similarly to what will be presented later, regarding the gap elements.

Note that in the MdFem program the load combinations are treated as load conditions, without exploiting the superposition of effects: this allows to solve nonlinear structures as in the case of tension elements or gap elements.

Tension elements are identified in the program by the Boolean variable ***OnlyTen****(Nelem)*, where Nelem is the total number of elements in the model.

In the main procedure of the program, called MDFEM(), inside the load conditions/combinations loop, there is the iterative process of checking for any unwanted compressions on the tension elements. The loop on load conditions/combinations is presented below

```
        For Icase = 1 To Ncase + NCOMB
            'restore material characteristics, possibly changed for Tension or Gap elements
            For Ielem = 1 To Nelem
                If OriginalMater(Ielem) > 0 Then
                    Mater(Ielem) = OriginalMater(Ielem)
                End If
            Next Ielem

            ' *** CALCULATE FIRST (ELASTIC) STIFFNESS MATRIX
            'always start the load case with a new Stiffness Matrix,
            ‘to avoid problems with iterative modifications of the matrix
            Call CreateStiffnessMatrix(FileWork1, FileWork3)

            Niteration = 0

            'initialize boolean variables in order to perform the first iteration
            TensionOnEl = True : CompressionOnEl = True

           'iteration loop for non linear elements (GAP or Tension elements)
            Do While TensionOnEl Or CompressionOnEl
                Niteration += 1

                Call LOADS(Icase, Niteration, File1)

                ' *** EQUATION SOLUTION
                Errors = False
                Call COLSOL(File1)
                If Errors Then
                    FileClose() : Exit Sub
                End If

                Call BACKSU(Icase)
                Call ComputeReactions(Icase, FileWork1)
                Call ComputeStresses(Icase, FileWork3)
```

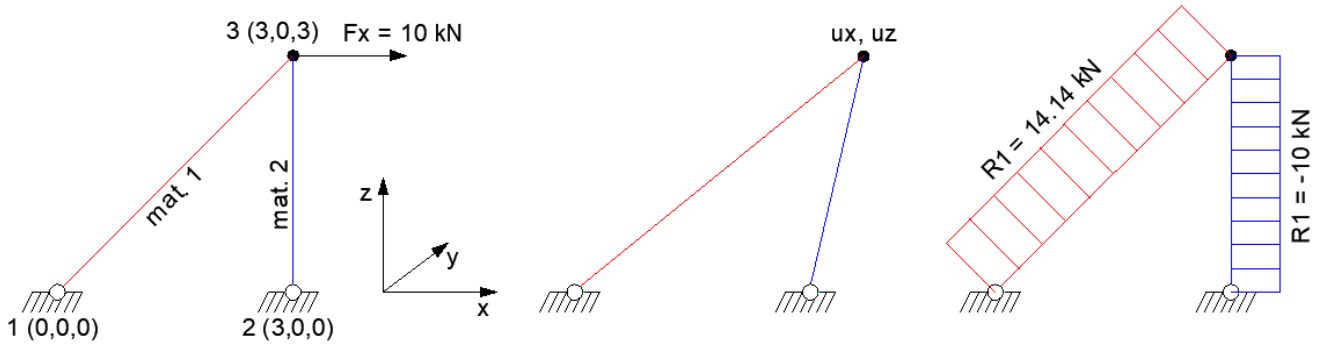
```
            'BEGIN NON LINEAR CONTROLS for GAP or TENSION elements ------------
            'checks for GAP elements (they can't have tension forces)
            TensionOnEl = False
            If Ngaps > 0 Then
                'if tension is found in GAP elements, their stiffness is reduced
                Call CheckGaps(Icase, TensionOnEl)
            End If

            'checks for Tension elements (they can't have compression forces)
            CompressionOnEl = False
            If NelOnlyTens > 0 Then
                'if compression is found in Tension elements,
                'a new material with zero area is assigned
                Call CheckTensElements(Icase, CompressionOnEl)
            End If

            If TensionOnEl Or CompressionOnEl Then 'update stiffness matrix
                Call CreateStiffnessMatrix(FileWork1, FileWork3)
            End If
            'END OF NON LINEAR CONTROLS -------------------------------------
        Loop

        If Ntype = 3 Then Call CalculateSoilStress()

        Call WriteResults(Icase, File1, File2)
    Next Icase
```

The subroutine CheckTensElements() is presented below.

```
Sub CheckTensElements(ByVal Icase As Integer, ByRef CompressionOnEl As Boolean)

    Dim OldMat, NewMat, Imats As Integer

    'define a new material with zero area and very small flexural inertia
    NewMat = Nmats + Ngaps + 1
    Imats = FindMatMax(1) : PROPS(NewMat, 1) = PROPS(Imats, 1) * 1.0E-18   'E
    PROPS(NewMat, 2) = 0   'area
    Imats = FindMatMax(3) : PROPS(NewMat, 3) = PROPS(Imats, 3) * 1.0E-18   'Jxx
    Imats = FindMatMax(4) : PROPS(NewMat, 4) = PROPS(Imats, 4) * 1.0E-18   'Jyy
    Imats = FindMatMax(8) : PROPS(NewMat, 8) = PROPS(Imats, 8) * 1.0E-18   'Jt

    For Ielem = 1 To Nelem - Ngaps
        If OnlyTens(Ielem) = True Then
            If Ntype = 2 Or Ntype = 3 Then
                If Stre1(Icase, Ielem, 1) < 0 Then
                    CompressionOnEl = True
                    'assign the element a material with zero area
                    OldMat = Mater(Ielem)
                    'G modulus
                    PROPS(NewMat, 10) = 0.5 * PROPS(NewMat, 1) / (1 + PROPS(OldMat, 9))
                    Mater(Ielem) = NewMat
                End If
            End If
        End If
    Next Ielem

End Sub
```

4.1 Validation example 4.1

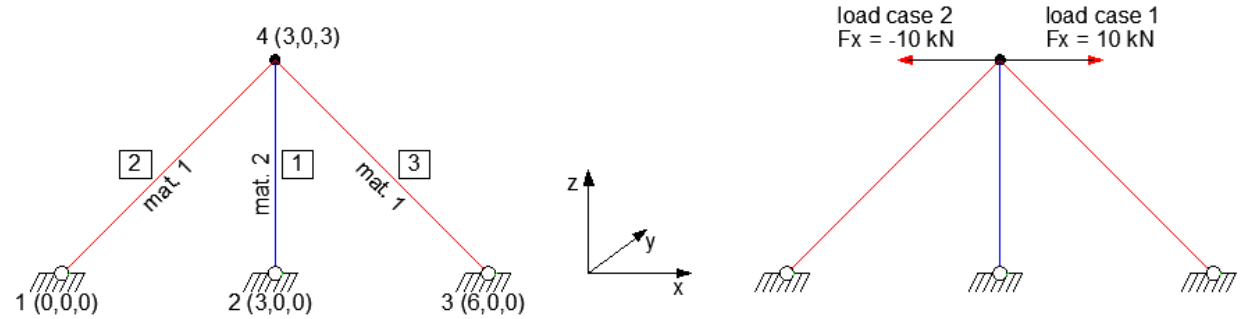
Consider the structure shown in the following figure. It is a simple isostatic structure, for which stresses and strains can also be calculated by hand. The adopted units are kN, m respectively for forces and lengths.



The following table shows the geometric and mechanical characteristics of the elements. Materials are supposed to be weightless.

Mat. 1 – ϕ20 – S235 steel bar	Mat. 2 – 0.4x0.4 – C25/30 concrete
A = 0.000314 m^2 J_{33} = 7.9E-9 m^4 J_{22} = 7.9E-9 m^4 J_t = 1.57E-8 m^4 E = 2.1E8 kN/m^2 ν = 0.3	A = 0.16 m^2 J_{33} = 0.002133 m^4 J_{22} = 0.002133 m^4 J_t = 0.00315733 m^4 E = 31447161 kN/m^2 ν = 0.1

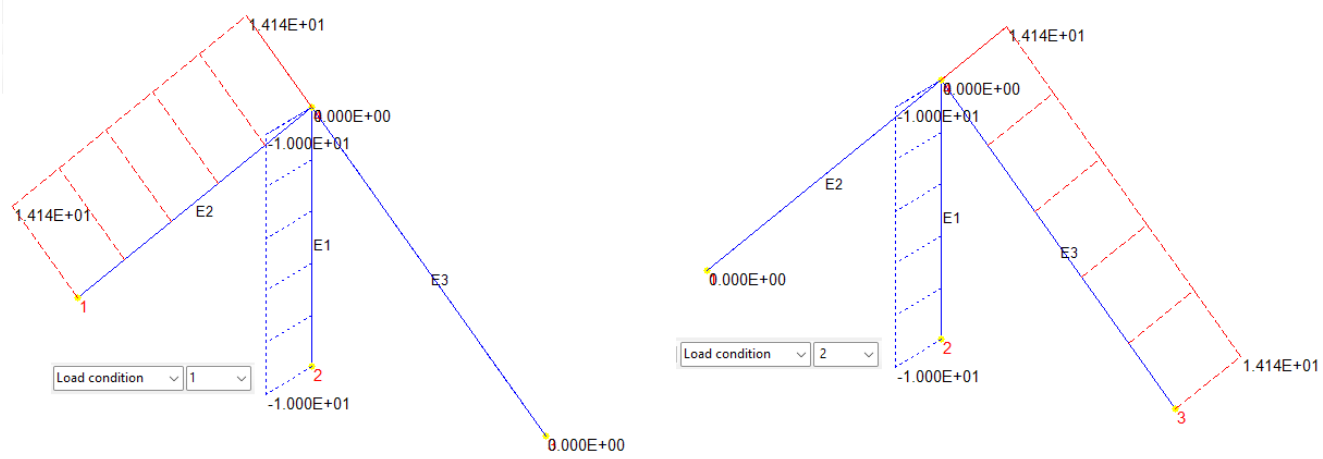
The results check was performed using the model of the previous figure with the SismiCad program, while with the MdFem program the model of the following figure was used: elements 2 and 3 are declared as "tension elements".



The comparison of the results is listed below.

	ux (m)	uz (m)	R1 el. 2, 3 (kN)	R1 el. 1 (kN)
SismiCad	1.2921E-3	-5.96E-6	14.1414	-10
MdFem	1.2928E-3	-5.96E-6	14.1414	-10

The results obtained with the two programs are practically the same. In the following image the results of MdFem are shown as they are presented by the program.



In this very simple case, only one iteration was necessary. After solving the original structure, the tension elements subject to compression were eliminated: at the first iteration the solution was already definitive since no tension element was subject to compression anymore.

5 Elements capable of only compressive stresses (gap elements)

This type of element can also be called “contact element”: it can be: 1) closed, with compressive force transmission; 2) open, without any force transmission. Contact forces are in the global directions x, y, z. They must be thought as mono-directional fixity codes in the x, y, z directions.

As already seen for tension elements, the management of gap elements requires an iterative procedure: if one of these elements is subject to tension stresses, its geometric-mechanical characteristics are reduced by a factor of 1000. Different reduction factors could be used, to prioritize the speed of execution (larger factors) or the precision of the results (smaller factors).

This is a more conservative approach than the one used for tension elements, but it involves a significantly higher number of iterations.

In any case, the approach can be easily modified, adopting the more drastic method presented for tension elements.

As noted at §4, the MdFem program the load combinations are treated as load conditions, without exploiting the superposition of effects, which is valid only for linear elastic behaviour and not for nonlinear structures.

Gap elements are identified in the program by the variable ***Gaps****(Npoin, NDIME)*, where *Npoin* is the total number of nodes in the model, and *NDIME* = 3 is the number of dimensions. At the nodes where a gap element is applied, the direction of action of this element must not be constrained. The value of the variable ***Gaps****(Npoin, NDIME)* can be +1 or -1 depending on whether the gap can react in the positive or negative direction parallel to one of the global reference axes. A gap element can act in only one direction.

From a practical point of view, a gap element is inserted into the model as a new element, with a length equal to 1/10 of the longest of the elements of the structure, with a very high area and with very low flexural stiffness.

The code used in MdFem for the attribution of the initial geometric-mechanical characteristics of the gap elements is listed below. For each characteristic, the maximum value present in the structure is first found, and then this value is multiplied or divided by the value 1E6. This is an arbitrary value, which can also be modified.

```
 'generate initial properties for gap elements
 For Igap = 1 To Ngaps
     Imats = FindMatMax(1) : PROPS(Nmats + Igap, 1) = PROPS(Imats, 1)             'E modulus
     Imats = FindMatMax(2) : PROPS(Nmats + Igap, 2) = PROPS(Imats, 2) * 1000000    'area
     If Ntype = 2 Or Ntype = 3 Then
         Imats = FindMatMax(3) : PROPS(Nmats + Igap, 3) = PROPS(Imats, 3) / 1000000 'Jxx
         Imats = FindMatMax(4) : PROPS(Nmats + Igap, 4) = PROPS(Imats, 4) / 1000000 'Jyy
         Imats = FindMatMax(8) : PROPS(Nmats + Igap, 8) = PROPS(Imats, 8) / 1000000 'Jt0
         PROPS(Nmats + Igap, 9) = 0                                              'Poisson ratio
         Imats = Nmats + Igap
         PROPS(Imats, 10) = 0.5 * PROPS(Imats, 1) / (1 + PROPS(Imats, 9))        'G modulus
     End If
 Next Igap
```

The subroutine that generates the gap elements is listed below.

```
    Sub GenerateGaps(File1 As StreamWriter)

        Dim Inew, Idim1, Idof1 As Integer
        Dim Delt1(3) As Single
        Dim Lmin, Lmax, Leng2, Lengt, Lgap As Single

        'find min, max lengths of elements in order to size the GAPS
        Lmin = 900000000000.0# : Lmax = -90000000000.0#
        For Ielem = 1 To Nelem
            Leng2 = 0
            For Idime = 1 To NDIME
                Delt1(Idime) = XYCOO(Idime + 3, Ielem) - XYCOO(Idime, Ielem)
                Leng2 = Leng2 + Delt1(Idime) * Delt1(Idime)
            Next Idime
            Lengt = Math.Sqrt(Leng2)

            If Lmin > Lengt Then Lmin = Lengt
            If Lmax < Lengt Then Lmax = Lengt
        Next Ielem
        Lgap = Lmax / 10 'gap element length

        'set nodal coordinates of gap elements
        PrtString = vbCrLf : PrtString += vbCrLf
        PrtString += "         GAP ELEMENTS DEFINITION" + vbCrLf

        PrtString += vbCrLf
        PrtString += "         ***  A D D E D   N O D A L   D A T A  ***" + vbCrLf
        ' *** WRITE NODAL DATA
        PrtString += vbCrLf
        PrtString += "         NODE         X-COORD.       Y-COORD.       Z-COORD."
        File1.WriteLine(PrtString)
        Inew = 0
        For Ipoin = 1 To Npoin
            For Idime = 1 To NDIME
                If GAPS(Ipoin, Idime) <> 0 Then
                    Inew = Inew + 1
                    For Jdime = 1 To NDIME
                        CORDS(Npoin + Inew, Jdime) = CORDS(Ipoin, Jdime) - GAPS(Ipoin, Jdime)
* Lgap
```

```
                    Next Jdime

                    'next 3 lines needed for local axes calculation
                    Xcoor(Npoin + Inew) = CORDS(Npoin + Inew, 1)
                    Ycoor(Npoin + Inew) = CORDS(Npoin + Inew, 2)
                    Zcoor(Npoin + Inew) = CORDS(Npoin + Inew, 3)

                    PrtString = String.Format("{0,13}", (Npoin + Inew).ToString("####0")) +
"   "
                    For Jdime = 1 To NDIME
                        PrtString   +=   String.Format("{0,15}",   CORDS(Npoin   +   Inew,
Jdime).ToString("0.0000E+00"))
                    Next Jdime
                    File1.WriteLine(PrtString)

                    IJINC(Nelem + Inew, 1) = Ipoin
                    IJINC(Nelem + Inew, 2) = Npoin + Inew

                    'next 2 lines needed for local axes calculation
                    Inc1(Nelem + Inew) = IJINC(Nelem + Inew, 1)
                    Inc2(Nelem + Inew) = IJINC(Nelem + Inew, 2)

                    Exit For
                End If
            Next Idime
        Next Ipoin

        'set restraint codes at new nodes
        For Ipoin = Npoin + 1 To Npoin + Ngaps
            For Idofn = 1 To NDOFN : IDDOF(Idofn, Ipoin) = 0 : Next Idofn
        Next Ipoin

        'generate elements
        PrtString = vbCrLf
        PrtString += "         ***  A D D E D   E L E M E N T S   ***" + vbCrLf
        PrtString += vbCrLf
        PrtString += "         TYPE  ELEMENT    NODE I   NODE J" '+ vbCrLf
        File1.WriteLine(PrtString)

        For Ielem = Nelem + 1 To Nelem + Ngaps
            'SET UP INCIDENCES
            For Idime = 1 To NDIME
                Idim1 = Idime + NDIME
                XYCOO(Idime, Ielem) = CORDS(IJINC(Ielem, 1), Idime)
                XYCOO(Idim1, Ielem) = CORDS(IJINC(Ielem, 2), Idime)
            Next Idime

            'incidences are already set above

            Mater(Ielem) = Nmats + Ielem - Nelem

            For Idofn = 1 To NDOFN
                Idof1 = Idofn + NDOFN
                'load degrees of freedom at element ends
                LMDOF(Idofn, Ielem) = IDDOF(Idofn, IJINC(Ielem, 1))
                LMDOF(Idof1, Ielem) = IDDOF(Idofn, IJINC(Ielem, 2))
            Next Idofn

            'CALCULATE COLUMN HEIGHTS
            Call COLHT(Ielem)

            PrtString = "          GAP  "
            PrtString += String.Format("{0,5}", Ielem.ToString("####0")) + "      "
            PrtString += String.Format("{0,5}", IJINC(Ielem, 1).ToString("####0")) + "    "
            PrtString += String.Format("{0,5}", IJINC(Ielem, 2).ToString("####0"))
            File1.WriteLine(PrtString)
        Next Ielem
```

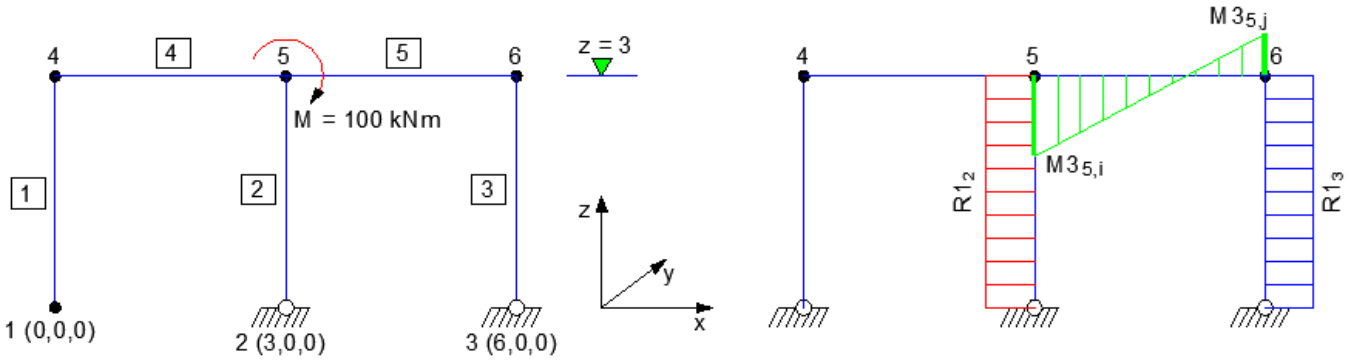
```
        Call CalculateTmat(Nelem + 1, Nelem + Ngaps) 'calculate transformation T matrix for
Gap elements: Tmat(,,)

        'update nodes and elements numbers
        OldNpoin = Npoin
        OldNelem = Nelem
        Npoin += Ngaps
        Nelem += Ngaps

    End Sub
```

The iterative process of checking for any unwanted tensile stresses on the gap elements is already shown in §4, while the actual checking subroutine is presented below.

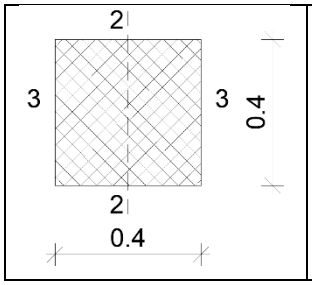
```
 Sub CheckGaps(ByVal Icase As Integer, ByRef TensionOnEl As Boolean)

     Dim Material As Integer

     For Ielem = Nelem - Ngaps + 1 To Nelem
         Material = Mater(Ielem)
         If Ntype = 2 Or Ntype = 3 Then
             If Stre1(Icase, Ielem, 1) > 0 Then
                 TensionOnEl = True
                 'reduce stiffness
                 PROPS(Material, 1) /= 1000 'Young
                 PROPS(Material, 2) /= 1000 'area
                 PROPS(Material, 3) /= 1000 'Jxx
                 PROPS(Material, 4) /= 1000 'Jyy
                 PROPS(Material, 8) /= 1000 'Jt
                 'G modulus
                 PROPS(Material, 10) = 0.5 * PROPS(Material, 1) / (1 + PROPS(Material, 9))
             End If
         End If
     Next Ielem

    End Sub
```

5.1 Validation example 5.1

Let's consider the structure shown in the following figure: it is the structure modelled with the Sap4 and SismiCad programs, in order to check the results obtained with gaps elements in MdFem. Due to the symmetry, only one load condition was considered with the control programs. Node 1 has no restraints, while node 2, 3 have a hinge restraint The adopted units are kN, m respectively for forces and lengths.

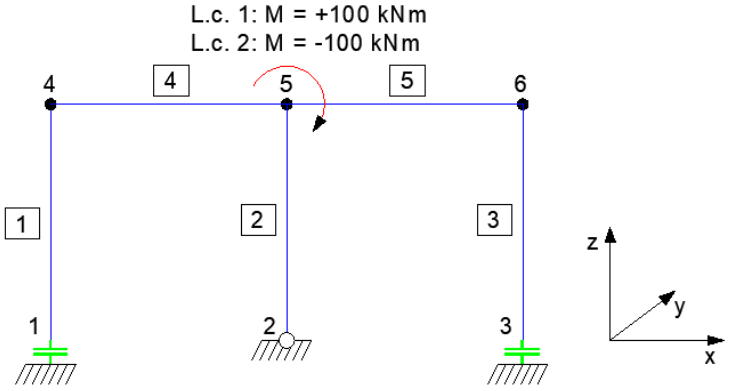


The following table shows the geometric and mechanical characteristics of the elements. Materials are supposed to be weightless.



C25/30 concrete – section 0.4x0.4
A = 0.16 m^2
J_{33} = 0.002133 m^4
J_{22} = 0.002133 m^4
J_t = 0.00315733 m^4
E = 31447161 kN/m^2
ν = 0.1

The check of gap elements in MdFem was performed using the model shown in the following figure: at nodes 2, 3 a gap element is inserted, acting in the +z direction.



The comparison of the results is listed below.

	SismiCad	Sap4	MdFem
$u_{z,4}$ (m)	0.001896	0.0018548	0.0018548
$R1_2$ (kN)	30.40	30.63	30.625
$R1_3$ (kN)	−30.40	−30.63	−30.625
$M3_{5,i}$ (kNm)	65.028	65.22	65.222
$M3_{5,j}$ (kNm)	−26.1692	26.65	−26.654

As in the example of §3.3, the results obtained with Sap4 are identical to those of MdFem, and in any case the results obtained with SismiCad are practically the same.

Despite the simplicity of the structure, with the conservative approach adopted for the management of the gap elements, 3 iterations were necessary in addition to the initial calculation.

6 Beams on elastic soil

The soil contribution to the stiffness of a beam element is calculated as indicated in [11]. The soil is modeled according to the Winkler hypothesis, i.e. with independent springs. This means that the soil has no cohesion, and that stresses cannot spread outside the loading area.

The stiffness K_w of the springs is defined by the ratio between the contact stress and the displacement of the point of application of the stress, calculated in the direction of the stress. The soil springs, by program convention, act in the local 2 direction of the element (see below for the definition of the local reference system). The soil stiffness is therefore:

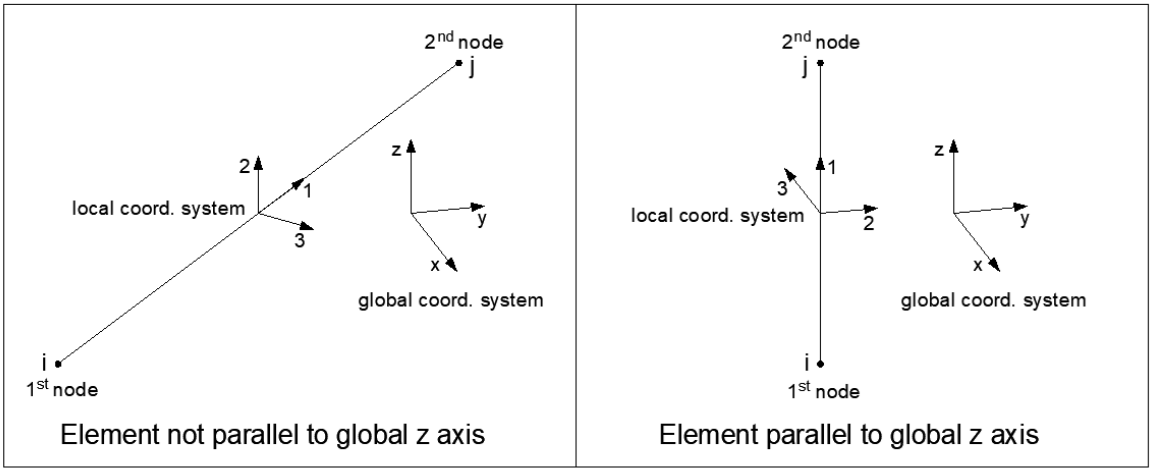
$$K_w = \frac{\sigma_2}{u_2}$$

In literature K_w is known as modulus or coefficient of subgrade reaction.

The local coordinate system of a beam is automatically defined as follows (see next figure):

- local axis 1 goes from first node i to second node j, along the beam axis;
- local axis 2, orthogonal to axis 1, is automatically set by the program: if the element direction is not parallel to the global z axis, axis 1 lays on the (1, z) plane; if the element is parallel to the global z axis, axis 2 is parallel to the global y axis;
- local axis 3 results from the cross (or vector) product of two versors in the direction of the axes 1 and 2. The right hand rule is respected.

Local Axes Definition



In reference [11] the contribution of the soil stiffness is calculated for a 2D beam, for which the only variables are the displacements orthogonal to the beam and the rotations at the ends of the beam itself. If we extend this contribution to a 3D beam, for which the soil acts only along the local direction 2, these variables are:

$$u_{2,i}, \theta_{3,i}, u_{2,j}, \theta_{3,j}$$

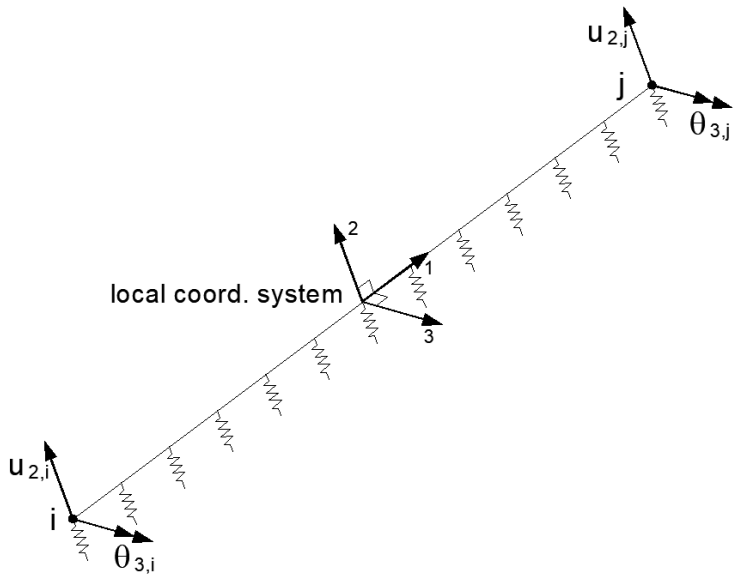
And the contribution of the soil stiffness is:

$$[K_{soil}] = K_w b\, L \begin{bmatrix} \frac{13}{35} & \frac{11}{210}L & \frac{9}{70} & -\frac{13}{420}L \\ & \frac{1}{105}L^2 & \frac{13}{420}L & -\frac{1}{140}L^2 \\ & & \frac{13}{35} & -\frac{11}{210}L \\ symm. & & & \frac{1}{105}L^2 \end{bmatrix}$$

Where: b = width of the beam resting on the soil

L = beam length

The next figure shows a 3D beam element with the variables involved by the soil stiffness.



The stiffness matrix of a beam is shown below, with the positions affected by the soil contribution highlighted.

$$
\begin{matrix}
[\,u_{1,i} & u_{2,i} & u_{3,i} & \theta_{1,i} & \theta_{2,i} & \theta_{3,i} & u_{1,j} & u_{2,j} & u_{3,j} & \theta_{1,j} & \theta_{2,j} & \theta_{3,j}\,]
\end{matrix}
$$

$$
[K_L] = \begin{bmatrix}
K_{1,1} & 0 & 0 & 0 & 0 & 0 & K_{1,7} & 0 & 0 & 0 & 0 & 0 \\
 & K_{2,2} & 0 & 0 & 0 & K_{2,6} & 0 & K_{2,8} & 0 & 0 & 0 & K_{2,12} \\
 & & K_{3,3} & 0 & K_{3,5} & 0 & 0 & 0 & K_{3,9} & 0 & K_{3,11} & 0 \\
 & & & K_{4,4} & 0 & 0 & 0 & 0 & 0 & K_{4,10} & 0 & 0 \\
 & & & & K_{5,5} & 0 & 0 & 0 & K_{5,9} & 0 & K_{5,11} & 0 \\
 & & & & & K_{6,6} & 0 & K_{6,8} & 0 & 0 & 0 & K_{6,12} \\
 & & & & & & K_{7,7} & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & K_{8,8} & 0 & 0 & 0 & K_{8,12} \\
 & & & & symm. & & & & K_{9,9} & 0 & K_{9,11} & 0 \\
 & & & & & & & & & K_{10,10} & 0 & 0 \\
 & & & & & & & & & & K_{11,11} & 0 \\
 & & & & & & & & & & & K_{12,12}
\end{bmatrix}
\begin{bmatrix} u_{1,i} \\ u_{2,i} \\ u_{3,i} \\ \theta_{1,i} \\ \theta_{2,i} \\ \theta_{3,i} \\ u_{1,j} \\ u_{2,j} \\ u_{3,j} \\ \theta_{1,j} \\ \theta_{2,j} \\ \theta_{3,j} \end{bmatrix}
$$

In conclusion, if the soil acts only along the local axis 2, calling $K^* = K_w b\, L$, the highlighted elements become:

$$K_{2,2} = \frac{12EJ_3}{L^3} + K^* \frac{13}{35}$$

$$K_{2,6} = \frac{6EJ_3}{L^2} + K^* \frac{11}{210} L$$

$$K_{2,8} = -\frac{12EJ_3}{L^3} + K^* \frac{9}{70}$$

$$K_{2,12} = \frac{6EJ_3}{L^2} - K^* \frac{13}{420} L$$

$$K_{6,6} = \frac{4EJ_3}{L} + K^* \frac{1}{105} L^2$$
$$K_{6,8} = -\frac{6EJ_3}{L^2} + K^* \frac{13}{420} L$$
$$K_{6,12} = \frac{2EJ_3}{L} - \frac{1}{140} L^2$$

$$K_{8,8} = \frac{12EJ_3}{L^3} + K^* \frac{13}{35}$$
$$K_{8,12} = -\frac{6EJ_3}{L^2} - \frac{11}{210} L$$

$$K_{12,12} = \frac{4EJ_3}{L} + K^* \frac{1}{105} L^2$$

In the MdFem program, the soil stiffness is added in the Beam3DstiffVaragnolo subroutine listed below, called after assembling the local stiffness matrix of the beam element.

```
    Sub AddWinkler(ByRef Kmat(,) As Double, Aleng As Double, Imats As Integer)

        Dim C3 As Double

        C3 = PROPS(Imats, 5) * PROPS(Imats, 11) * Aleng

        Kmat(2, 2) += C3 * (13 / 35)
        Kmat(2, 6) += C3 * (11 / 210) * Aleng
        Kmat(2, 8) += C3 * (9 / 70)
        Kmat(2, 12) += C3 * (-13 / 420) * Aleng

        Kmat(6, 6) += C3 * (1 / 105) * Aleng ^ 2
        Kmat(6, 8) += C3 * (13 / 420) * Aleng
        Kmat(6, 12) += C3 * (-1 / 140) * Aleng ^ 2

        Kmat(8, 8) += C3 * (13 / 35)
        Kmat(8, 12) += C3 * (-11 / 210) * Aleng

        Kmat(12, 12) += C3 * (1 / 105) * Aleng ^ 2

    End Sub
```

After solving the system of equations, which gives the displacements referred to the global reference system, we still have to calculate the soil stress, which is:

$$\sigma_{soil} = K_w \, u_2$$

The relationship between the displacements of node i in the global x, y, z directions, and the displacements of the same node in the local 1, 2, 3 directions is:

$$[u_i]_L^T = [t] \, [u_i]_G^T \qquad (6.1)$$

where: $[u_i]_L^T = \left[u_{1,i} \; u_{2,i} \; u_{3,i}\right]$ are the displacements at node i expressed in local coordinates;

$[u_i]_G^T = \left[u_{x,i} \; u_{y,i} \; u_{z,i}\right]$ are the displacements at node i expressed in global coordinates;

$[t] = \begin{bmatrix} \alpha_{1,x} & \alpha_{1,y} & \alpha_{1,z} \\ \alpha_{2,x} & \alpha_{2,y} & \alpha_{2,z} \\ \alpha_{3,x} & \alpha_{3,y} & \alpha_{3,z} \end{bmatrix}$ are the cosines of the angles between local and global axes

For node j, similar relations apply.

Since only the displacement in the local 2 direction is of interest, the application of relation (6.1) is reduced to:

$$u_{2,i} = \alpha_{2,x}\, u_{x,i} + \alpha_{2,y}\, u_{y,i} + \alpha_{2,z}\, u_{z,i}$$

$$u_{2,j} = \alpha_{2,x}\, u_{x,j} + \alpha_{2,y}\, u_{y,j} + \alpha_{2,z}\, u_{z,j}$$

Below is the listing of the *CalculateSoilStress()* subroutine that calculates the soil stresses at the nodes of the elements.

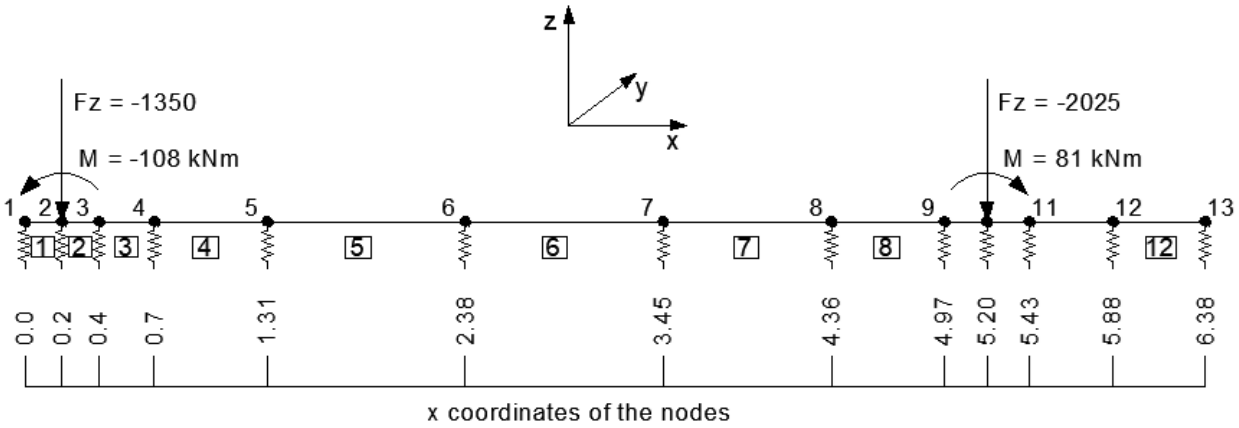
```
Sub CalculateSoilStress()
    'calculate soil stresses for WINK elements, parallel to local 2 axis

    Dim Mat, In1, In2 As Integer

    For Icase = 1 To Ncase + NCOMB
      For i% = 1 To Nelem
        Mat = Mater(i%)
        In1 = Inc1(i%)
        In2 = Inc2(i%)

        If PROPS(Mat, 5) * PROPS(Mat, 11) <> 0 Then
         'project global displacement at node i in local 2 direction
         SoilStress(Icase, In1)=Tmat(i%, 2, 1)*Displ(Icase, In1, 1)*PROPS(Mat, 11)
         SoilStress(Icase, In1)+=Tmat(i%, 2, 2)*Displ(Icase, In1, 2)*PROPS(Mat, 11)
         SoilStress(Icase, In1)+=Tmat(i%, 2, 3)*Displ(Icase, In1, 3)*PROPS(Mat, 11)

         'project global displacement at node j in local 2 direction
         SoilStress(Icase, In2)=Tmat(i%, 2, 1)*Displ(Icase, In2, 1)*PROPS(Mat, 11)
         SoilStress(Icase, In2)+=Tmat(i%, 2, 2)*Displ(Icase, In2, 2)*PROPS(Mat, 11)
         SoilStress(Icase, In2)+=Tmat(i%, 2, 3)*Displ(Icase, In2, 3)*PROPS(Mat, 11)
        End If
      Next i%
    Next Icase

End Sub
```

6.1 Validation example 6.1

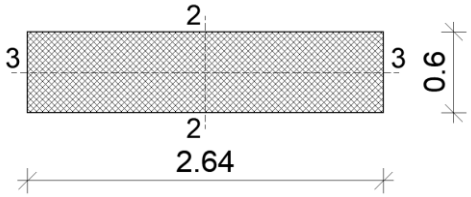
Let's consider the structure shown in the next figure: this problem is taken from [2], where it was solved by inserting concentrated springs of appropriate stiffness at the nodes. This may be a sufficiently approximate approach, and in any case this is a good method to easily check the goodness of the implementation of the Winkler beam finite element. Unfortunately, in the cited text there is an error in the calculation of the springs applied to the first and last node.

For this reason, for the validation of the MdFem program, the structure with concentrated springs was first solved using the values published in [2]. Once the results were checked, the structure was calculated with the correct spring values, then comparing the results with those obtained by MdFem with Winkler elements. At each node only u_z and θ_y dof are allowed.

The adopted units are kN, m respectively for forces and lengths.



The following table shows the geometric and mechanical characteristics of the elements. Materials are supposed to be weightless.



Concrete – section 2.64x0.6
A = 0.1.584 m^2
J_{33} = 0.04752 m^4
J_{22} = 0.9199872 m^4
J_t = 0.162864 m^4
E = 21700000 kN/m^2
ν = 0.1

The coefficient of subgrade reaction K_w has a value of 22000 kN/m^3. The spring value K_n at node *n* is given by the following product:

$$K_n = K_w \, b \, \frac{L_{n-1} L_n}{2}$$

where: b = 2.64 m is the width of the concrete section

L_{n-1} is the length of the element to the left of node n

L_{n-1} is the length of the element to the right of node n

The following table contains the calculation of the spring values and the values found in [2]. The first and last values are different because in [2] the entire length of the element has been used instead of half the length.

Node	x-coord. (m)	Element length (m)	Kz (kN/m) (correct value)	Kz (kN/m) (Bowles)
1	0		5808.00	11616
2	0.2	0.2	11616.00	11616
3	0.4	0.2	14520.00	14520
4	0.7	0.3	26426.40	26426.41
5	1.31	0.61	48787.20	48787.2
6	2.38	1.07	62145.60	62145.59
7	3.45	1.07	57499.20	58499.2
8	4.36	0.91	44140.80	44140.8
9	4.97	0.61	24393.60	24393.61
10	5.2	0.23	13358.40	13358.41
11	5.43	0.23	19747.20	19747.2
12	5.88	0.45	27588.00	27588
13	6.38	0.5	14520.00	29040.02

The first comparisons, between the results published in [2] and those of MdFem, is listed below. The comparison is made with the displacements and with the bending moments. The soil pressures are proportional to the displacements and therefore there is no need to control them.

Displacements (m)	Bowles	MdFem	Error (%)
$u_{z,1}$	0.01181	0.01179	0.17
$u_{z,2}$	0.01131	0.01129	0.18
$u_{z,10}$	0.00875	0.00875	0
$u_{z,13}$	0.00972	0.00972	0

Bending Moments (kNm)	Bowles	MdFem	Error (%)
$M3_{2,i}$	−83	−80.66	2.8
$M3_{2,j}$	296	−296.93	0.3
$M3_{max}$	1225.25	1224.4	0.07
$M3_{9,i}$	195.25	194.42	0.4
$M3_{9,j}$	−468.75	468.21	0.1

The results are practically the same, ignoring some strange signs indicated in [2] for the bending moments.

We then proceed to calculate the same structure with the MdFem program: 1) with the correct values of the concentrated springs; 2) again with concentrated spring, but dividing the structure into a greater number of elements; 3) using the Winkler elements presented in this chapter.

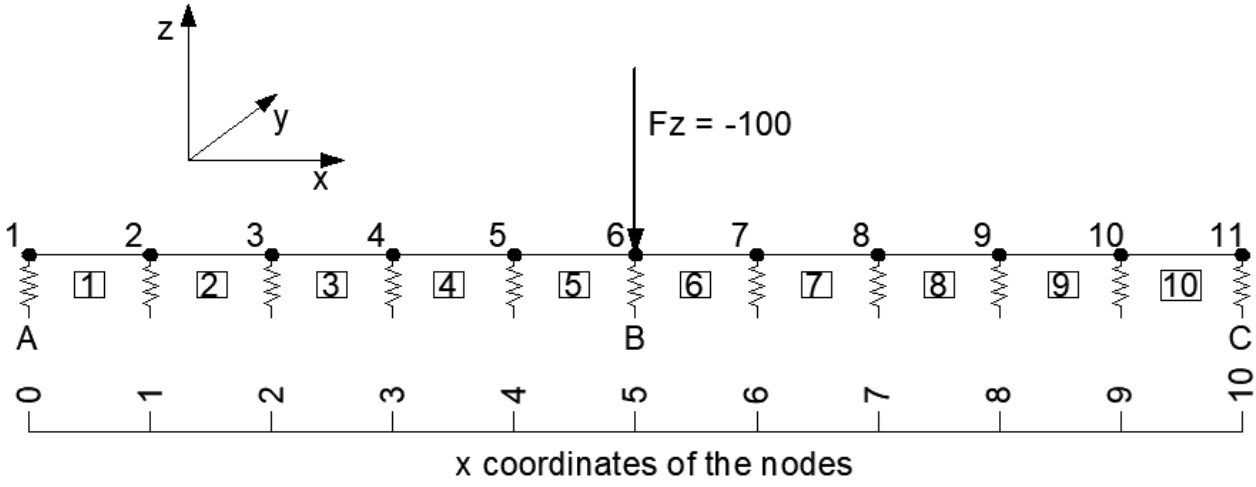
Displacements (m)	MdFem with correct spring values	MdFem with thickened mesh	MdFem with Winkler elements
$u_{z,x=0}$	0.01212	0.01228	0.1233
$u_{z,x=0.2}$	0.01161	0.01175	0.1179
$u_{z,x=5.2}$	0.00971	0.00974	0.00975
$u_{z,x=6.38}$	0.01125	0.01132	0.01134

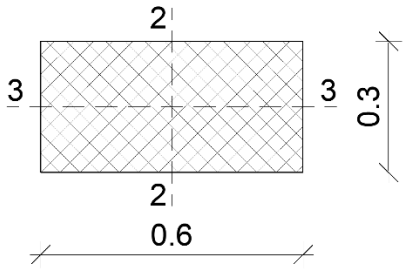
Bending Moments (kNm)	MdFem with correct spring values	MdFem with thickened mesh	MdFem with Winkler elements
$M3_{x=0.2}$	−93.9	−93.6	−93.9
$M3_{max}$	−1341.0	−1344.6	−1345.9
$M3_{x=5.2}$	356.6	356.7	356.7

By dividing the structure into a greater number of elements, closer values are obtained between the concentrated spring model and the one with Winkler elements. It can be seen that the displacements increase and tend to the values calculated with the continuous model of the Winkler element.

6.2 Validation example 6.2

Another validation example was made with the structure represented in the next figure, using the SismiCad and MdFem programs, with the (continuous) Winkler elements presented in this chapter. The adopted units are kN, m, rad respectively for forces, lengths and angles. Materials are supposed to be weightless.





C25/30 concrete – section 0.6x0.3

$A = 0.18\ m^2$
$J_{33} = 0.00135\ m^4$
$J_{22} = 0.0054\ m^4$
$J_t = 0.003699\ m^4$
$E = 31447161\ kN/m^2$
$\nu = 0.1$

The coefficient of subgrade reaction K_w has a value of 15000 kN/m^3.

The SismiCad program proposes a mesh with 6 elements by default. The comparison with MdFem is conducted with three discretizations: one with only 2 elements, one with 6 elements and one with 10 elements. In the following table the results are listed, referred to the end points A, C and the central point B.

	SismiCad (6 elements)	MdFem (2 elements)	MdFem (6 elements)	MdFem (10 elements)
$u_{z\,A} = u_{z\,C}$	7.31E-4	7.2E-4	7.31E-4	7.31E-4
$u_{z\,B}$	-2.804E-3	-2.758E-3	-2.804E-3	-2.804E-3
$M3_B$	52.91	53.78	52.88	52.90

The results are positive, with an error of 1.6% on the bending moment already with only two elements. With 6 elements and with 10 elements the results are practically the same.

A check was also done with the elements arranged along the z-axis, to check the correctness of the implementation: the check gave a positive result.

7 Program MdFem

The MdFem program has been published in [9] and [10], and adding the subroutines presented in this paper does not require complex work. For this reason I have chosen not to republish the entire program. In any case, anyone interested can request the entire program at *info@studioingegneriavaragnolo.com*.

The complete program also includes a preprocessor with a user manual and a graphical interface for the visualization of the model and the results.

Anyone wishing to use the program must remember that numerous checks have been carried out, but the responsibility always remains of the user as set out below.

8 Final Remarks

With the features presented in this article, the MdFem program reaches a decent level, and is able to analyze fairly complex structures.

A lot of time has been spent to achieve this result, although of course the MdFem program is still not comparable to many commercial programs available on the market.

In any case, I think that the practical tips and insights on the techniques adopted, with detailed and step-by-step explanations of some examples, can be useful to many readers.

I was happy to write this article and the previous ones [9] and [10], so I could put the work done in order.

9 Bibliography

[1] Bathe K. J., *Finite element procedures in engineering analysis*; Prentice-Hall, New Jersey; (1982).

[2] Bowles J. E., *Foundation Analysis and Design*; McGraw-Hill International Editions; (1988).

[3] Capurso M*., Introduzione al calcolo automatico delle strutture*; E.S.A.C. - Roma; (1977).

[4] Cesari F., *Introduzione al metodo degli elementi finiti*; Pitagora Editrice Bologna; (1982).

[5] Clarke D., *Computer e strutture*; BE-MA Editrice - Milano; (1979).

[6] Gugliotta, A.: "*Elementi Finiti – Parte I"*. Otto editore, (2002).

[7] Hinton E., Owen D.R.J., *An introduction to finite element computations*; Pineridge Press Limited - Swansea, U.K.; (1979).

[8] Schrefler B., Vitaliani R., *Calcolo automatico dei telai spaziali*; CLEUP - Padova; (1978).

[9] Varagnolo P*., 3-D Beam Finite Element Programming -A Practical Guide: Part 1 – Static Analysis*; ResearchGate, (2021).

[10] Varagnolo P*., 3-D Beam Finite Element Programming - A Practical Guide Part 2 – Dynamic Modal Analysis (Software Included)*; ResearchGate, (2024).

[11] Vitaliani R., Martini L., *Lezioni di calcolo automatico - 1ª parte*; CUSL NUOVA VITA - Padova; (1987).

[12] Zienkiewicz O.C., *The finite element method - third edition*; McGRAW-HILL, U.K.; (1977).