Programma per l'analisi FEM di Elementi Beam 3-D
Guida Pratica
Parte 1 – Analisi Statica
(Software Incluso)

Giugno 2021

Paolo Varagnolo: Ingegnere libero professionista, info@studioingegneriavaragnolo.com

Rev. 1 - aprile 2024

Indice

1 Generalità

Viene presentato e commentato il programma agli elementi finiti MdFem (Mono dimensional Finite Element Method), derivato dal programma STAP presentato da K.J. Bathe in [1]. L'impostazione generale di questo programma è la stessa che si trova in [1] e nei classici programmi SAP e ADINA.

Il linguaggio di programmazione utilizzato è invece vb.net di Microsoft.

L'obiettivo di questo lavoro è quello di fornire degli approfondimenti pratici sui metodi e sulle procedure utilizzate, presentando degli esempi molto dettagliati. Si pensa così di fornire un contributo a quanti vogliano avvicinarsi al metodo degli elementi finiti (FEM) dal punto di vista dello sviluppatore di software.

Gli esempi svolti possono anche aiutare gli studenti di ingegneria che studiano il metodo degli elementi finiti.

La parte teorica e matematica non verrà invece affrontata, in quanto già ampiamente disponibile in numerosi testi e articoli.

Il programma MdFem è nato in 2-D nel 1988 ed è stato registrato a nome di Paolo Varagnolo Ingegneria nel 1994. Nell'ultima versione, che viene descritta nel seguito, è stato implementato l'elemento Beam 3-D con le seguenti caratteristiche: si tratta di un elemento rettilineo a 2 nodi, con 3 gradi di libertà traslazionali e 3 rotazionali ad ogni nodo, atto a trasmettere forze assiali, taglianti, momenti flettenti e torcenti.

Vengono descritte le funzionalità di base del programma per l'analisi statica, rimandando ad eventuali futuri approfondimenti altre funzionalità come lo svincoli di alcuni gradi di libertà ai nodi, la possibilità che gli elementi possano reagire solo a trazione, l'inserimento di molle, l'inserimento di elementi gap, la possibilità di considerare gli elementi su un letto di molle alla Winkler.

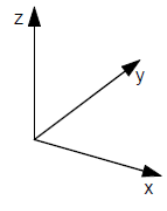
Di seguito viene descritta l'impostazione del programma.

Il problema statico può essere descritto con la seguente espressione:

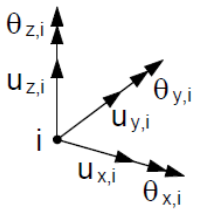
$$[K_G]\,[u_G] = [f_G] \qquad (1.1)$$

Dove $[K_G]$ è la matrice di rigidezza, $[u_G]$ il vettore degli spostamenti generalizzati (cioè spostamenti e rotazioni), $[f_G]$ il vettore delle forze generalizzate (cioè forze e momenti). **Questo sistema di equazioni è riferito al sistema di riferimento globale (x, y, z)**: gli spostamenti generalizzati e le forze generalizzate ad un nodo generico i sono indicati nella figura seguente, e in notazione matriciale sono indicati come:

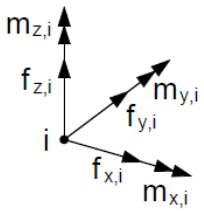
$$[u_G]^T = [u_{x,i} \quad u_{y,i} \quad u_{z,i} \quad \theta_{x,i} \quad \theta_{y,i} \quad \theta_{z,i}]$$
$$[f_G]^T = [f_{x,i} \quad f_{y,i} \quad f_{z,i} \quad m_{x,i} \quad m_{y,i} \quad m_{z,i}]$$



sistema di riferimento globale



spostamenti globali al nodo i



forze globali al nodo i

1.1 Definizione del problema statico nel sistema di riferimento locale

Risulta comodo impostare il problema nel riferimento locale, in quanto è semplice distinguere le caratteristiche geometrico-statiche della sezione dell'elemento nelle direzioni principali, e altrettanto semplice risulta definire i carichi generalizzati.

Nella figura seguente è rappresentato un elemento che va dal nodo i al nodo j, con evidenziati gli spostamenti e i carichi riferiti al sistema di riferimento locale (1, 2, 3).

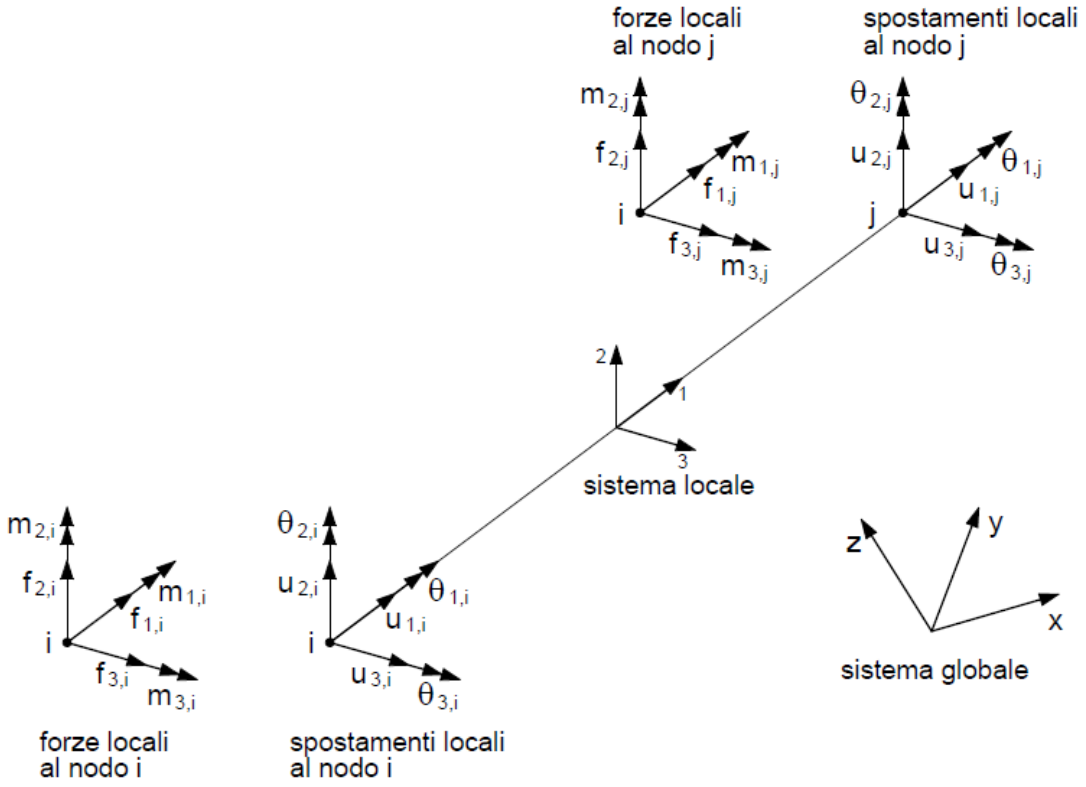
Il sistema di riferimento locale viene così definito:

- l'asse locale 1 va dal nodo i al nodo j;
- l'asse locale 2 viene scelto dal programma nel modo seguente: se l'elemento non è parallelo all'asse z globale l'asse 2 è ortogonale all'asse 1 e sta nel piano (1, z); se l'elemento è verticale l'asse 2 è parallelo all'asse y globale;
- l'asse locale 3 risulta dal prodotto vettoriale dei versori paralleli agli assi (1, 2).

Gli spostamenti generalizzati e le forze generalizzate dell'elemento sono indicati nella figura seguente, e in notazione matriciale sono indicati come:

$$[u_L]^T = [u_{1,i} \quad u_{2,i} \quad u_{3,i} \quad \theta_{1,i} \quad \theta_{2,i} \quad \theta_{3,i} \quad u_{1,j} \quad u_{2,j} \quad u_{3,j} \quad \theta_{1,j} \quad \theta_{2,j} \quad \theta_{3,j}]$$

$$[f_L]^T = [f_{1,i} \quad f_{2,i} \quad f_{3,i} \quad m_{1,i} \quad m_{2,i} \quad m_{3,i} \quad f_{1,j} \quad f_{2,j} \quad f_{3,j} \quad m_{1,j} \quad m_{2,j} \quad m_{3,j}]$$



La matrice di rigidezza nel sistema locale è la seguente:

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

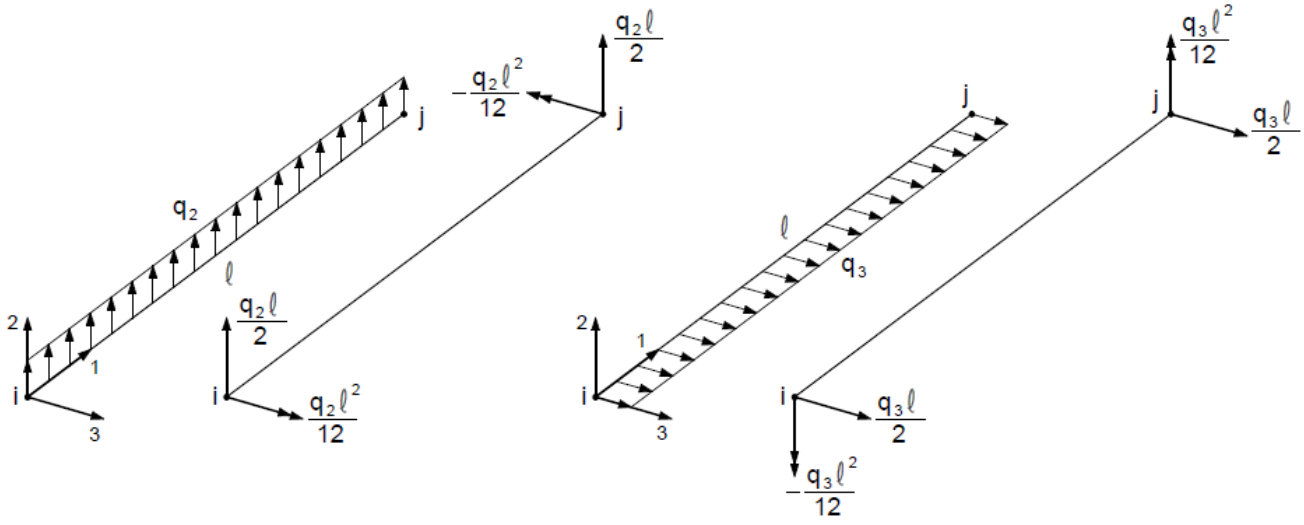
Gli spostamenti generalizzati ai nodi i, j sono legati agli elementi della matrice di rigidezza locale come evidenziato di seguito.

$$\begin{bmatrix} u_{1,i} & u_{2,i} & u_{3,i} & \theta_{1,i} & \theta_{2,i} & \theta_{3,i} & u_{1,j} & u_{2,j} & u_{3,j} & \theta_{1,j} & \theta_{2,j} & \theta_{3,j} \end{bmatrix}$$

$$\begin{bmatrix}
\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12EJ_3}{L^3} & 0 & 0 & 0 & \frac{6EJ_3}{L^2} & 0 & -\frac{12EJ_3}{L^3} & 0 & 0 & 0 & \frac{6EJ_3}{L^2} \\
0 & 0 & \frac{12EJ_2}{L^3} & 0 & -\frac{6EJ_2}{L^2} & 0 & 0 & 0 & -\frac{12EJ_2}{L^3} & 0 & -\frac{6EJ_2}{L^2} & 0 \\
0 & 0 & 0 & \frac{GJ_1}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ_1}{L} & 0 & 0 \\
0 & 0 & -\frac{6EJ_2}{L^2} & 0 & \frac{4EJ_2}{L} & 0 & 0 & 0 & \frac{6EJ_2}{L^2} & 0 & \frac{2EJ_2}{L} & 0 \\
0 & \frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{4EJ_3}{L} & 0 & -\frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{2EJ_3}{L} \\
-\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12EJ_3}{L^3} & 0 & 0 & 0 & -\frac{6EJ_3}{L^2} & 0 & \frac{12EJ_3}{L^3} & 0 & 0 & 0 & -\frac{6EJ_3}{L^2} \\
0 & 0 & -\frac{12EJ_2}{L^3} & 0 & \frac{6EJ_2}{L^2} & 0 & 0 & 0 & \frac{12EJ_2}{L^3} & 0 & \frac{6EJ_2}{L^2} & 0 \\
0 & 0 & 0 & -\frac{GJ_1}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ_1}{L} & 0 & 0 \\
0 & 0 & -\frac{6EJ_2}{L^2} & 0 & \frac{2EJ_2}{L} & 0 & 0 & 0 & \frac{6EJ_2}{L^2} & 0 & \frac{4EJ_2}{L} & 0 \\
0 & \frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{2EJ_3}{L} & 0 & -\frac{6EJ_3}{L^2} & 0 & 0 & 0 & \frac{4EJ_3}{L}
\end{bmatrix}
\begin{bmatrix} u_{1,i} \\ u_{2,i} \\ u_{3,i} \\ \theta_{1,i} \\ \theta_{2,i} \\ \theta_{3,i} \\ u_{1,j} \\ u_{2,j} \\ u_{3,j} \\ \theta_{1,j} \\ \theta_{2,j} \\ \theta_{3,j} \end{bmatrix}$$

1.2 Carichi nodali equivalenti

Vengono considerati dei carichi uniformemente distribuiti q_2, q_3, rispettivamente lungo gli assi locali 2, 3. Nella seguente figura sono indicati i carichi equivalenti, e di seguito il vettore delle forze locali.



carico distribuito lungo l'asse 2 carico distribuito lungo l'asse 3

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

Se ci fosse anche un carico uniformemente distribuito lungo la direzione locale 1, non rappresentato nella figura precedente, il vettore delle forze locali diventerebbe:

$$[f_L]^T = \left[\frac{q_1 l}{2} \quad \frac{q_2 l}{2} \quad \frac{q_3 l}{2} \quad 0 \quad -\frac{q_3 l^2}{2} \quad \frac{q_2 l^2}{2} \quad \frac{q_1 l}{2} \quad \frac{q_2 l}{2} \quad \frac{q_3 l}{2} \quad 0 \quad \frac{q_3 l^2}{2} \quad -\frac{q_2 l^2}{2}\right] \qquad (1.2)$$

Eventuali carichi distribuiti in modo non uniforme possono essere trattati seguendo la stessa logica, e cioè trasportandoli ai nodi in modo equivalente.

Si noti che q_2, q_3 producono sempre momenti di segno opposto, come si può vedere nella figura precedente.

1.3 Trasformazioni fra coordinate locali e globali

Le matrici di rigidezza e il vettore dei carichi equivalenti devono essere espressi in coordinate globali per essere assemblati nel sistema espresso dalla relazione (1.1). Le trasformazioni dal sistema locale a quello globale vengono fatte utilizzando i coseni degli angoli compresi fra gli assi locali e gli assi globali, attraverso una matrice di trasformazione $[T]$.

$$[T] = \begin{bmatrix} [t] & & & \\ & [t] & & \\ & & [t] & \\ & & & [t] \end{bmatrix}$$

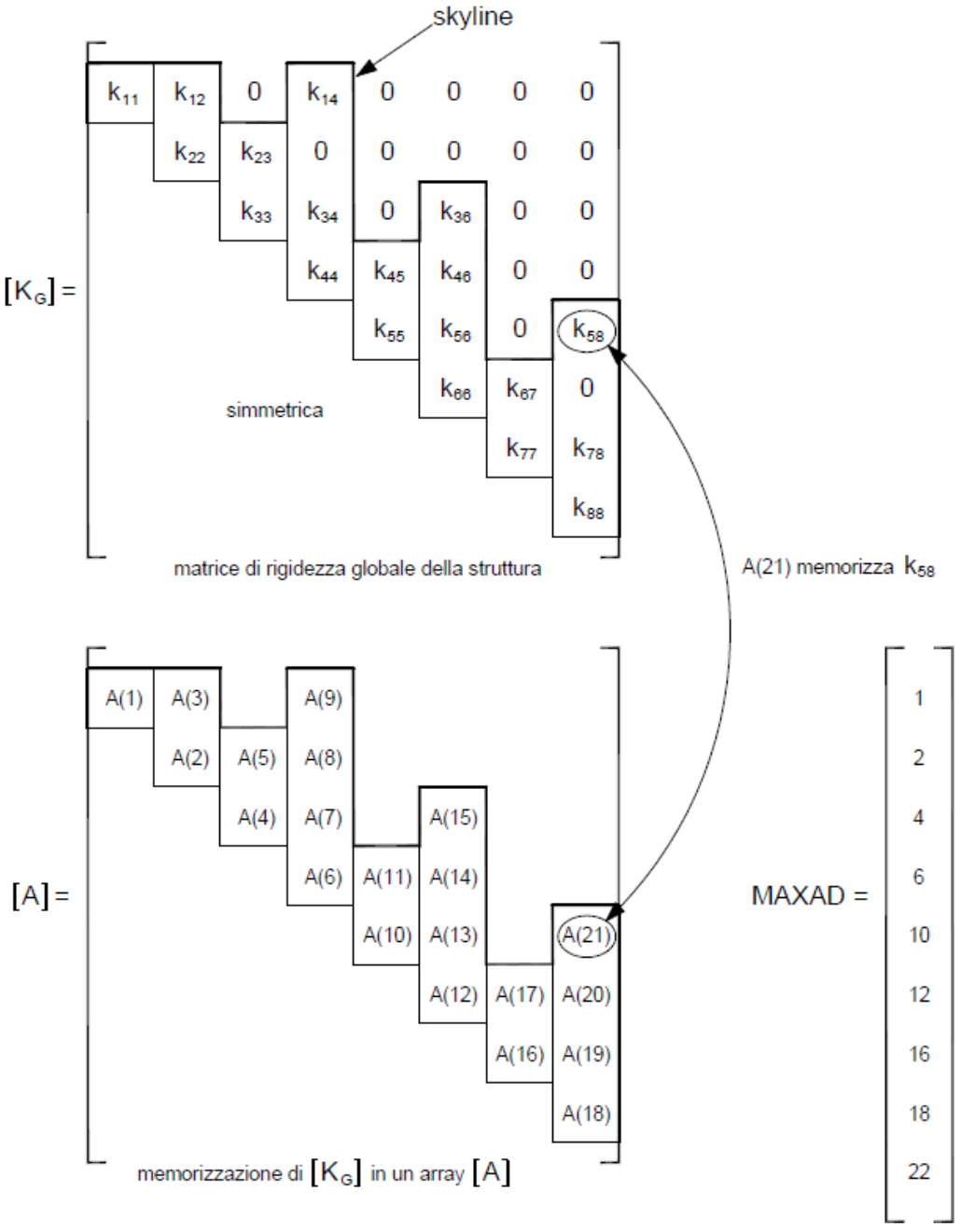
dove

$$[t] = \begin{bmatrix} \alpha_{1,x} & \alpha_{1,y} & \alpha_{1,z} \\ \alpha_{2,x} & \alpha_{2,y} & \alpha_{2,z} \\ \alpha_{3,x} & \alpha_{3,y} & \alpha_{3,z} \end{bmatrix}$$

in cui $\alpha_{i,j}$ è il coseno dell'angolo formato dall'asse locale i con l'asse globale j. Gli elementi di $[T]$ non indicati sono nulli.

1.4 Memorizzazione della matrice di rigidezza globale della struttura

La matrice di rigidezza globale della struttura $[K_G]$ viene memorizzata in forma compatta secondo l'algoritmo delle colonne attive descritto in [1]. Vengono cioè memorizzati in un array monodimensionale $[A]$ i soli elementi al disopra della diagonale principale e al disotto della skyline (questi elementi formano quelle che sono definite colonne attive). Per conoscere la posizione degli elementi della matrice così memorizzata, è necessario un vettore ausiliario, denominato ***MAXAD***(), che contiene le posizioni degli elementi della diagonale principale. Nella figura seguente è schematizzato il meccanismo di memorizzazione della matrice di rigidezza globale.



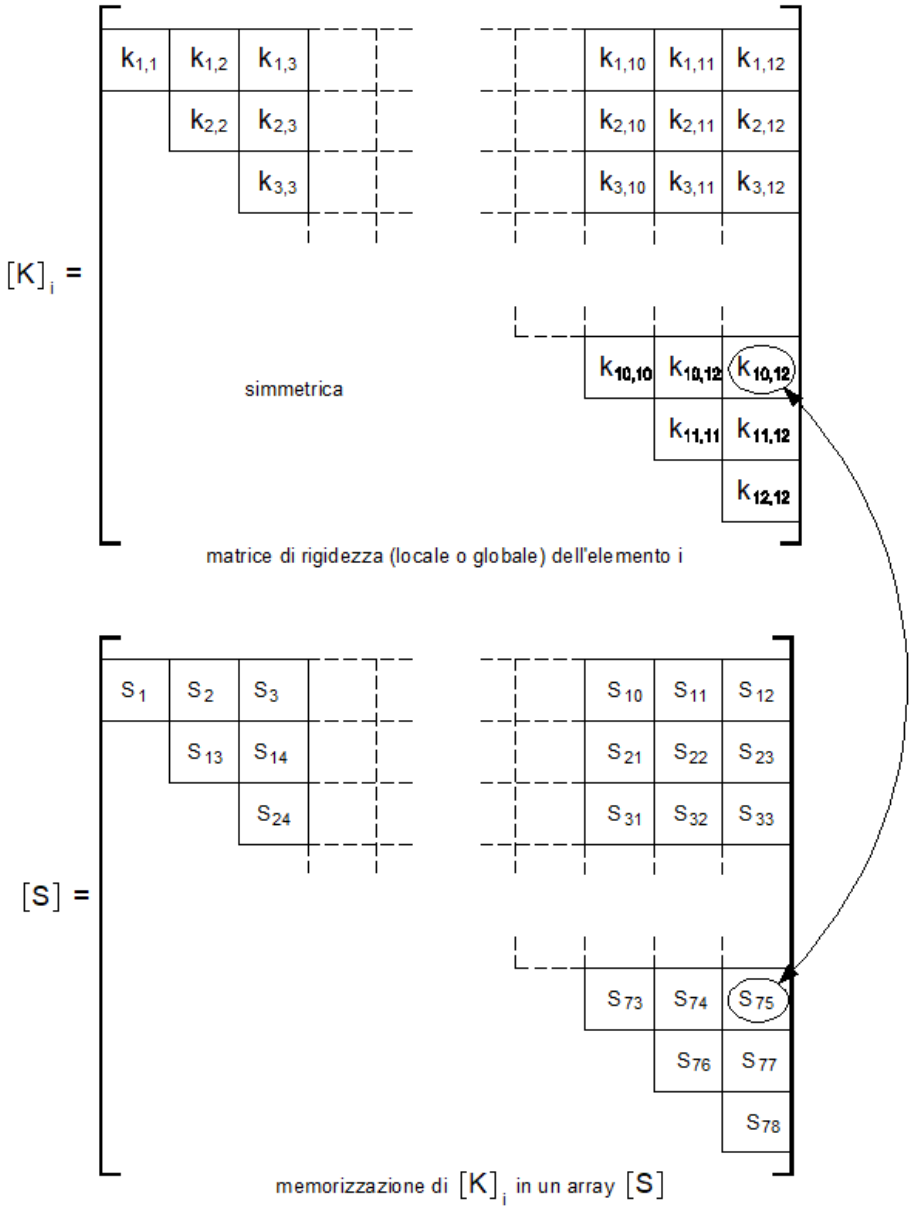
Nel programma il numero di gradi di libertà della struttura viene denominato ***Ndoft***, mentre il numero di elementi al disotto della skyline viene chiamato ***Nkglo***.

Il vettore $[A]$ viene chiamato ***GLOBK***(*Index*), con *Index* = 1 ÷ *Nkglo*.

L'individuazione degli elementi di questa matrice in funzione degli indici di riga e colonna viene presentata nel §2.3 all'interno della subroutine ADDBAN.

1.5 Memorizzazione delle matrici di rigidezza degli elementi

Le matrici di rigidezza di ciascun elemento, sia nel sistema di riferimento locale, sia nel sistema di riferimento globale, vengono memorizzate in forma compatta in un array monodimensionale $[S]$ secondo l'algoritmo del programma STAP in [1]: nell'array vengono conservati i soli elementi al disopra della diagonale principale, come indicato nello schema seguente. La matrice di 12 x 12 elementi diventa un vettore di 78 elementi.



Nel programma il vettore $[S]$ viene chiamato ***Stiff***(*Index*), con *Index* = 1 ÷ 78.

L'individuazione degli elementi di questa matrice in funzione degli indici di riga i e colonna j avviene così: dati (i, j), con j>i, l'indice m del vettore $[S]$ vale:

$$m = [(j - i) + 1] + \sum_{k=0}^{i-2} (12 - k)$$

2 Dettagli dell'analisi F.E.M.

2.1 Numerazione dei gradi di libertà

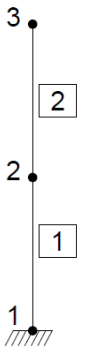
Le informazioni sui gradi di libertà della struttura, di seguito abbreviati in dof (degrees of freedom), vengono memorizzate in una matrice ***IDDOF***(*Idofn*, *Ipoin*), dove *Idofn* è l'indice dei gradi di libertà presenti in un nodo e *Ipoin* è l'indice del nodo della struttura.

Idofn = 1 ÷ *Ndofn* dove ***Ndofn*** = 6 è il n° di gradi di libertà presenti in un nodo

Ipoin = 1 ÷ *Npoin* dove ***Npoin*** è il n° totale di nodi della struttura

Inizialmente la matrice contiene il numero 1 in corrispondenza dei dof vincolati. In queste posizioni viene successivamente sostituito il numero zero, ad indicare che non è necessaria la soluzione del corrispondente dof in quanto lo spostamento corrispondente è noto e pari a 0. Nelle posizioni inizialmente nulle vengono invece numerati in ordine crescente i dof per i quali deve essere risolto lo spostamento incognito.

Esempio 1: nella figura a lato è rappresentata una struttura costituita da 2 elementi e da 3 nodi. Il nodo 1 è incastrato e non ha gradi di libertà, nei nodi 2, 3 ci sono 6 + 6 gradi di libertà, 3 spostamenti e 3 rotazioni ciascuno.



$$IDDOF = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

matrice *IDDOF* dopo l'input dei dati

$$IDDOF = \begin{bmatrix} 0 & 1 & 7 \\ 0 & 2 & 8 \\ 0 & 3 & 9 \\ 0 & 4 & 10 \\ 0 & 5 & 11 \\ 0 & 6 & 12 \end{bmatrix}$$

matrice *IDDOF* dopo la numerazione dei dof

Il numero totale di gradi di libertà della struttura *Ndoft* in questo esempio è pari a 12.

Nel caso siano definiti degli svincoli alle estremità degli elementi, devono essere inseriti ulteriori gradi di libertà.

2.2 Calcolo degli indirizzi degli elementi della diagonale principale di $[K_G]$

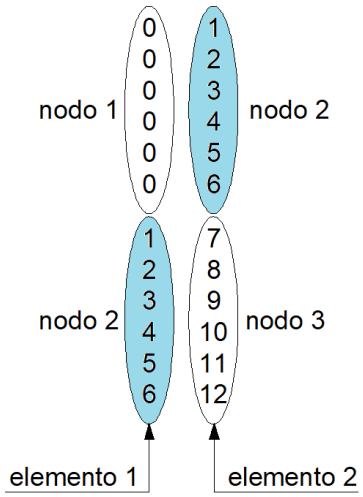
Quando vengono definite le incidenze degli elementi, e cioè i nodi iniziali i e finali j degli elementi, i dof alle estremità vengono conservati in una matrice ***LMDOF***(*Idofn*, *Ielem*), dove *Idofn* è l'indice dei gradi di libertà ai nodi dell'elemento e *Ielem* è l'indice dell'elemento.

Idofn = 1 ÷ *Nevab* dove ***Nevab*** = 2 · *Ndofn* = 12 è il n° di dof in un elemento

Ielem = 1 ÷ *Nelem* dove ***Nelem*** è il n° totale di elementi della struttura

Facendo riferimento all'esempio del paragrafo precedente, l'elemento 1 va dal nodo 1 al nodo 2, l'elemento 2 va dal nodo 2 al nodo 3. I dof associati agli elementi sono pertanto i seguenti:

$$LMDOF = \begin{bmatrix} 0 & 1 \\ 0 & 2 \\ 0 & 3 \\ 0 & 4 \\ 0 & 5 \\ 0 & 6 \\ 1 & 7 \\ 2 & 8 \\ 3 & 9 \\ 4 & 10 \\ 5 & 11 \\ 6 & 12 \end{bmatrix}$$



Nel programma le informazioni sulle incidenze degli elementi sono conservate nella matrice ***IJINC***(*Ielem*, *Inode*), dove *Ielem* è l'indice dell'elemento e *Inode* indica il primo o il secondo nodo dell'elemento. Nel caso in esame questa matrice assume i seguenti valori:

$$IJINC = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$$

A partire dalle informazioni organizzate come appena esposto, è possibile calcolare le altezze delle colonne attive definite nel §1.4: le altezze vengono memorizzate nel vettore ***MCOLH***(*Idofn*), con *Idofn* che indica il dof globale. La massima altezza delle colonne (**M**aximum **COL**umn **H**eight) viene calcolata come segue: per ogni elemento viene dapprima determinato il n° del minore dei dof dei nodi di estremità (chiamato ***Ldofn***); successivamente, per ognuno dei dof alle estremità dell'elemento, si calcola la massima altezza della colonna, pari alla differenza fra il dof corrente *Idofn* e *Ldofn*.

Con queste informazioni è infine possibile calcolare gli indirizzi degli elementi della diagonale principale dei coefficienti della matrice di rigidezza globale. Tali indirizzi sono memorizzati nel vettore *MAXAD*(*Idofn*), come già detto nel §1.4, con *Idofn* = 1 ÷ *Ndoft*.

I primi 2 valori di *MAXAD*() sono sempre uguali a 1 e 2: *MAXAD*(1) = 1; *MAXAD*(2) = 2. Per tutti i successivi dof vale la seguente espressione:

MAXAD(*Idofn* + 1) = *MAXAD*(*Idofn*) +*MCOLH*(*Idofn*) + 1

Nell'esempio corrente i valori di *MAXAD* sono riportati nell'immagine seguente, dove sono evidenziati due valori di *MCOLH*() calcolati di seguito a titolo di esempio.

<u>calcolo dell'altezza della colonna attiva per il dof n° 10 dell'elemento 1</u>

Ldofn=1

LMDOF(1, 10) = **4** (il 10° dof dell'elemento è il **4° dof della struttura**)

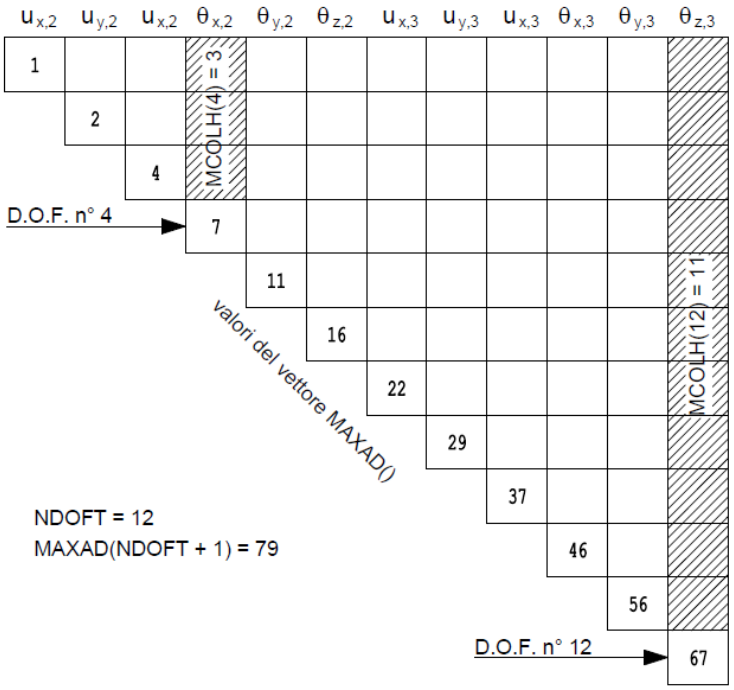
MCOLH(4) = **4** – 1 = 3

<u>calcolo dell'altezza della colonna attiva per il dof n° 12 dell'elemento 2</u>

Ldofn=1

LMDOF(2, 12) = **12** (il 12° dof dell'elemento è il **12° dof della struttura**)

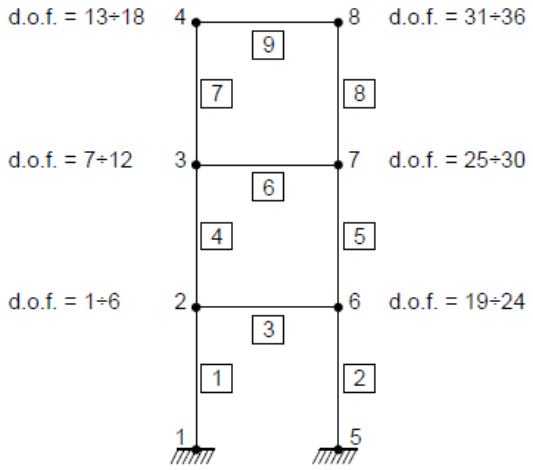
MCOLH(12) = **12** – 1 = 11



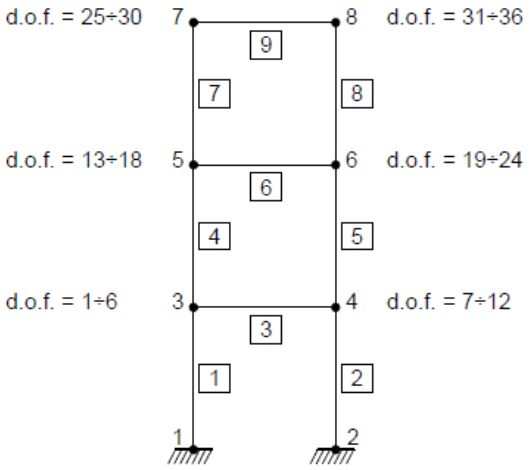
La semplicità della struttura dell'esempio 1 non consente di apprezzare i vantaggi del metodo delle colonne attive, infatti la matrice dei coefficienti del sistema risulta completamente riempita e non ci sono vantaggi in termini di occupazione di memoria.

D'altra parte non è facile rappresentare graficamente esempi di una complessità tale da rendere evidenti i vantaggi del metodo. Di seguito viene presentato l'esempio di 2 strutture per le quali si possono apprezzare non solo i vantaggi del metodo delle colonne attive, ma anche le differenze che originano da diverse numerazioni dei nodi.

Esempio 2



BadNumb



GoodNumb

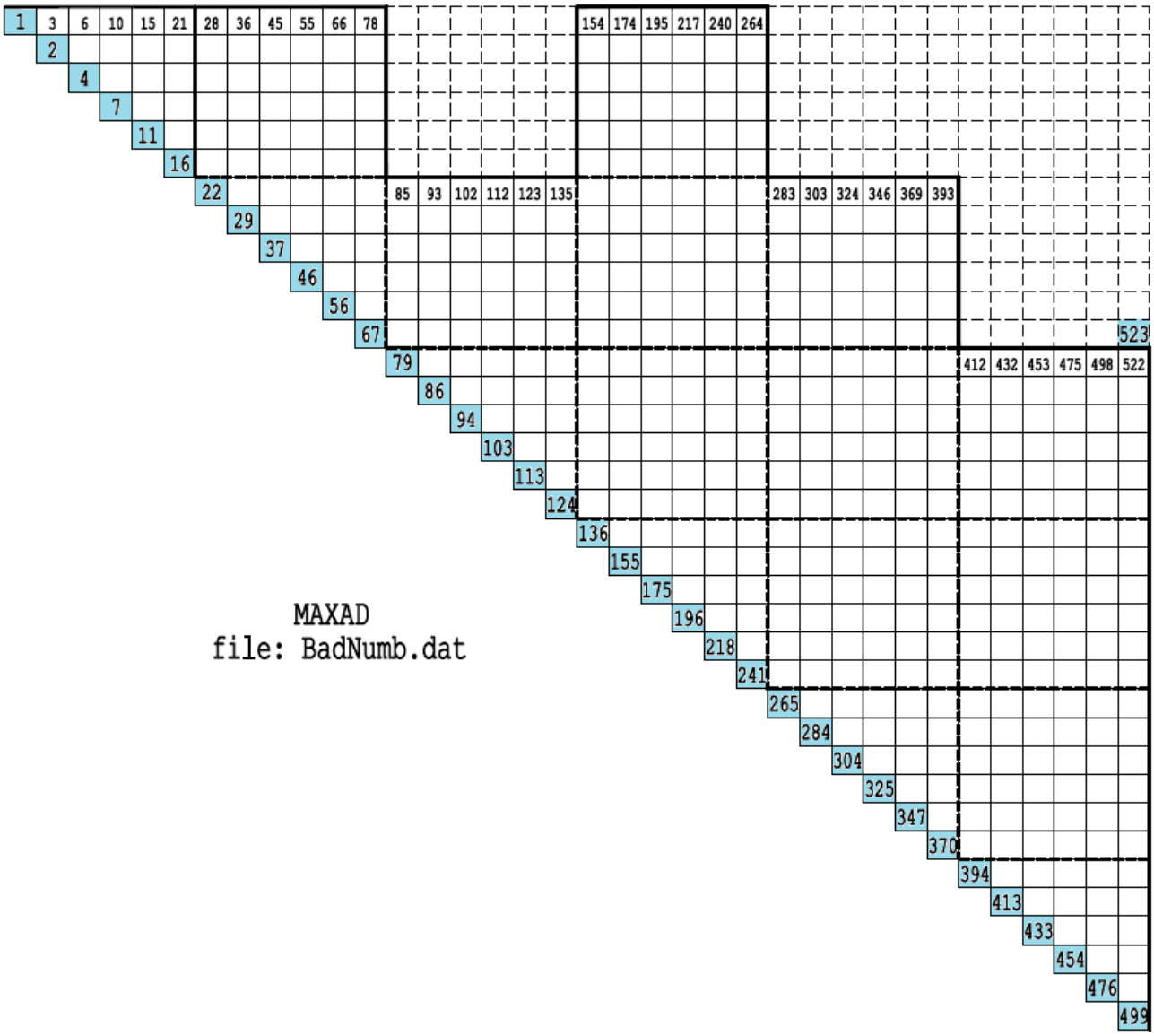
Vengono riportate direttamente le immagini schematiche delle matrici risultanti nei due casi, nelle quali sono evidenziate la skyline e i valori di *MAXAD*.

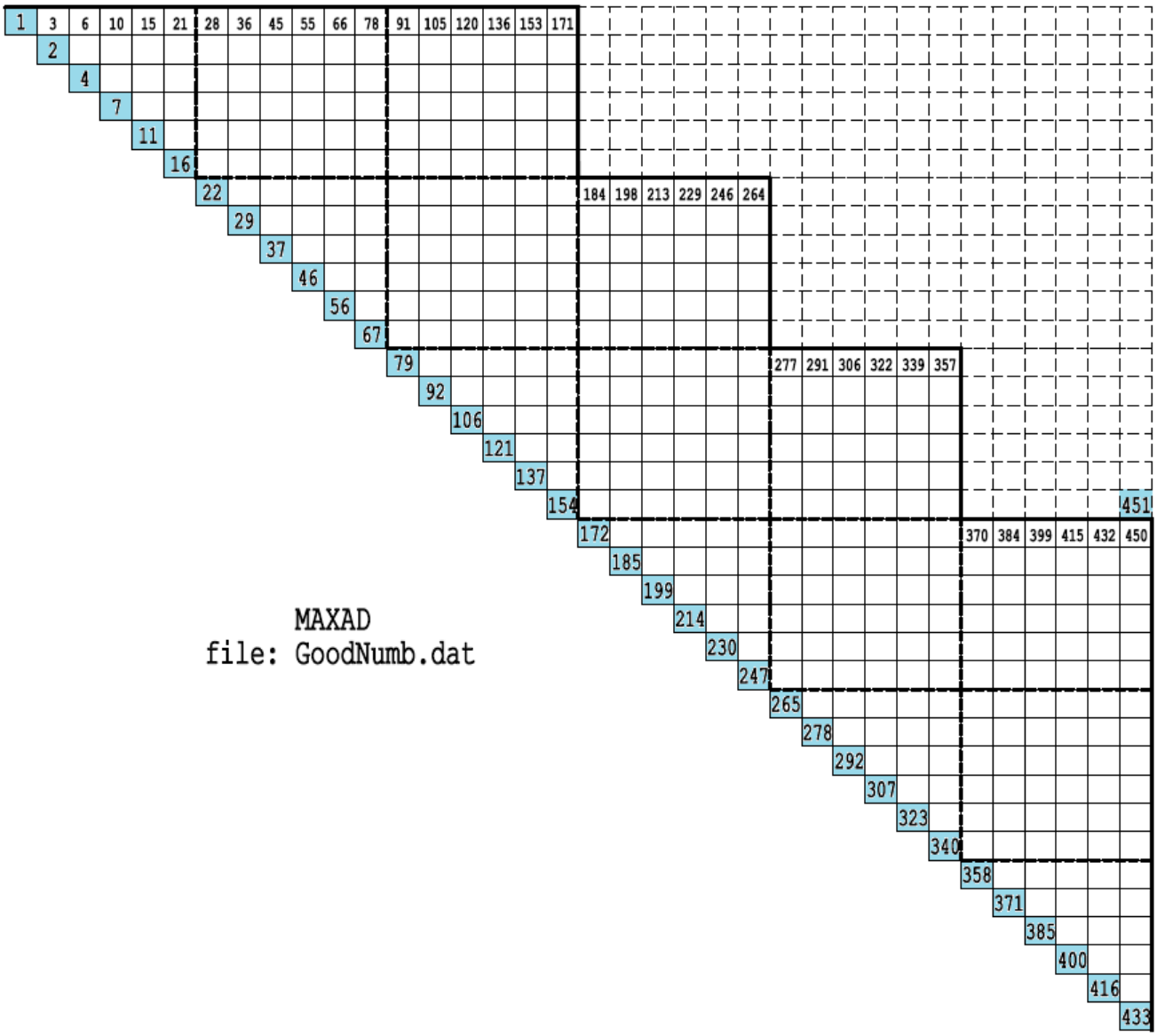
In entrambi i casi, nonostante la semplicità delle strutture, si comincia a vedere il vantaggio offerto dal metodo delle colonne attive (o skyline).

Con la numerazione della struttura di sinistra, denominata BadNumb, i termini da memorizzare e da gestire sono 522.

Con la numerazione della struttura di destra, denominata GoodNumb, i termini da memorizzare e da gestire sono solo 450.

La matrice completa, che dovrebbe essere gestita nel caso di metodi di memorizzazione tradizionali, conterrebbe 36 · 36 = 1296 elementi.



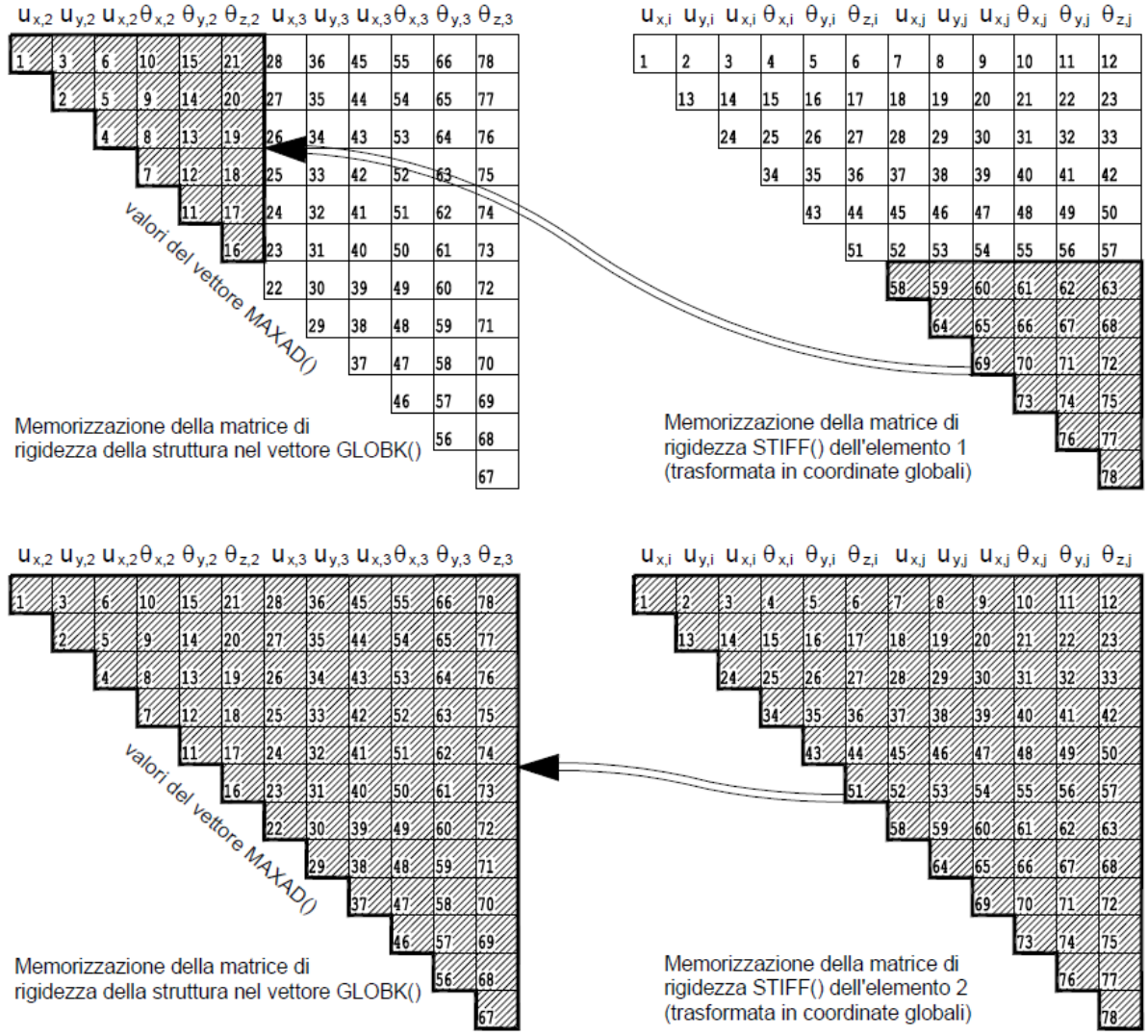

1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 153 171
2
4
7
11
16
22
29
37
46
56
67
79
92
106
121
137
154
172
185
199
214
230
247
265
278
292
307
323
340
358
371
385
400
416
433
184 198 213 229 246 264
277 291 306 322 339 357
370 384 399 415 432 450
451
MAXAD
file: GoodNumb.dat

2.3 Assemblaggio delle matrici di rigidezza

Poiché le matrici di rigidezza vengono create nel sistema di riferimento locale, mentre le incognite del sistema da risolvere sono riferite al sistema di riferimento globale, è innanzitutto necessario procedere alle seguenti trasformazioni per le matrici di ciascun elemento.

matrice di rigidezza da locale a globale	$[K_G] = [T]^T\ [K_L]\ [T]$	(2.1)

Con riferimento alla matrice *LMDOF*() dell'esempio 1 presentato nel §2.1, si ottiene l'assemblaggio riportato nella seguente figura.

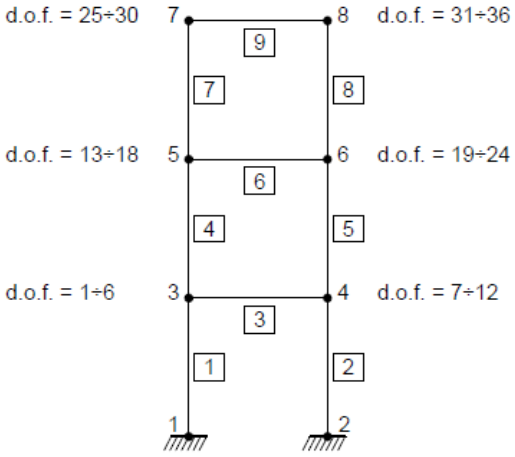


Infatti l'elemento 1 ha i dof 1÷6, che corrispondono a $[u_{x,2}\quad u_{y,2}\quad u_{z,2}\quad \theta_{x,2}\quad \theta_{y,2}\quad \theta_{z,2}]$, nel nodo j che corrisponde al nodo 2; l'elemento 2 ha invece i dof 1÷12, che corrispondono a $[u_{x,2}\quad u_{y,2}\quad u_{z,2}\quad \theta_{x,2}\quad \theta_{y,2}\quad \theta_{z,2}\quad u_{x,3}\quad u_{y,3}\quad u_{z,3}\quad \theta_{x,3}\quad \theta_{y,3}\quad \theta_{z,3}]$, nei nodi i, j che corrispondono ai nodi 2, 3.

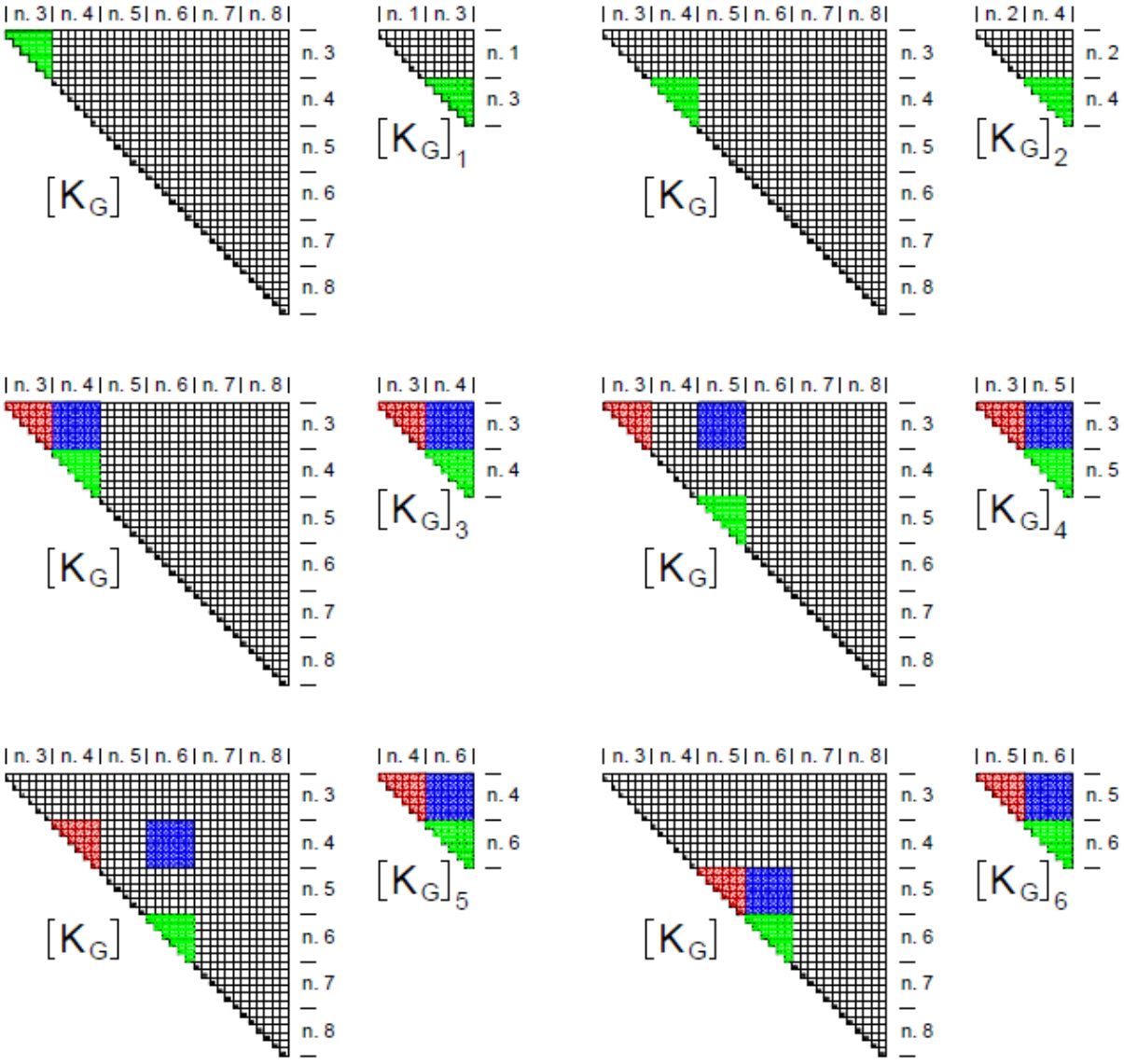
Il fatto che la matrice di rigidezza globale della struttura sia di ordine pari a quella dei singoli elementi è un caso particolare legato alle caratteristiche dell'esempio. La matrice di rigidezza degli elementi è sempre 12x12, mentre quella della struttura ha ordine pari al numero di gradi di libertà della struttura (*Ndoft*).

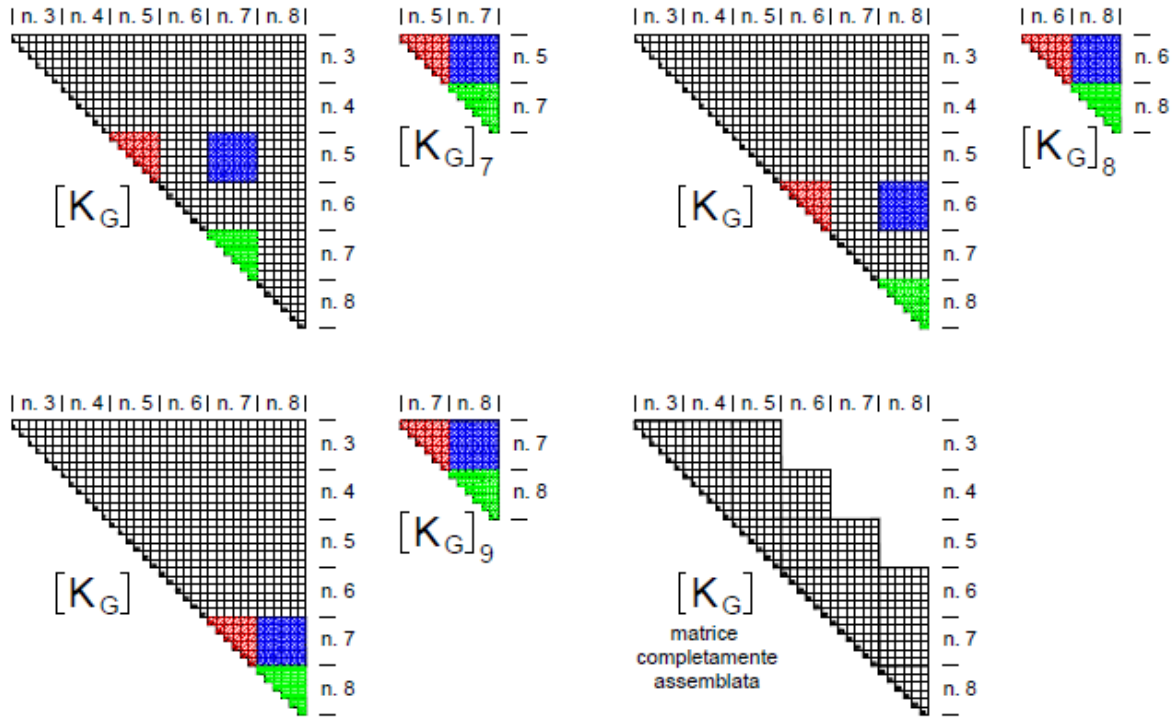
Per comprendere meglio il processo di assemblaggio delle matrici di rigidezza viene ripreso l'esempio 2 del §2.2, considerando la struttura denominata GoodNumb. Nelle figure seguenti sono

evidenziate le destinazioni delle matrici dei singoli elementi (naturalmente già trasformate in coordinate globali). Le destinazioni sono legate ai dof dei nodi di estremità degli elementi.



GoodNumb





In questo caso i dof dell'intera struttura sono 36 e la matrice $[K_G]$ ha dimensioni 36x36. Le matrici $[K_G]_i$ dei singoli elementi sono sempre 12x12. Nelle figure sono evidenziati i numeri dei nodi per indicare sinteticamente i 6 dof che vi afferiscono.

L'ultima immagine conferma la forma della skyline e le posizioni con coefficienti diversi da zero della matrice presentata nel §2.2.

Di seguito viene presentata la subroutine utilizzata nel programma per l'assemblaggio delle matrici di rigidezza di un elemento *Ielem*: si chiama ADDBAN ed è praticamente la stessa presentata in [1]. Le variabili utilizzate sono già state precedentemente definite.

```
    Sub ADDBAN(Ielem)
        Dim N1, Idofn, Jdofn, Ievab, MAADD, Kevab, IJdif, Inde1, Inde2 As Integer

        N1 = 0

        For Ievab = 1 To NEVAB
            Idofn = LMDOF(Ievab, Ielem)

            If Idofn > 0 Then
                MAADD = MAXAD(Idofn)
                Kevab = Ievab

                For Jevab = 1 To NEVAB
                    Jdofn = LMDOF(Jevab, Ielem)
                    IJdif = Idofn - Jdofn

                    If Jdofn > 0 And IJdif >= 0 Then
                        Inde1 = MAADD + IJdif
                        Inde2 = Kevab
                        If Jevab >= Ievab Then Inde2 = Jevab + N1
                        GLOBK(Inde1) = GLOBK(Inde1) + Stiff(Inde2)
                    End If
                    Kevab = Kevab + NEVAB - Jevab
                Next Jevab
```

```
            End If
            N1 = N1 + NEVAB - Ievab
        Next Ievab

    End Sub
```

La comprensione di questa subroutine è complicata dal fatto che la matrice *GLOBK*() è memorizzata come indicato nel §1.4, mentre le matrici locali *Stiff*() sono memorizzate come indicato nel §1.5. Entrambe le tecniche sono tratte da [1].

2.4 Assemblaggio dei carichi

Nel §1.2 è stato definito il vettore dei carichi equivalenti $[f_L]$, a partire dai carichi distribuiti

$$[q_L]^T = [q_1 \quad q_2 \quad q_3 \quad 0 \quad 0 \quad 0 \quad q_1 \quad q_2 \quad q_3 \quad 0 \quad 0 \quad 0]$$

riferiti al sistema di riferimento locale. Se i carichi sono riferiti al sistema di riferimento globale, è necessario trasformarli prima nel sistema di riferimento locale. Nel sistema globale i carichi uniformemente distribuiti sono del tipo:

$$[q_G]^T = [q_{x,i} \quad q_{y,i} \quad q_{z,i} \quad 0 \quad 0 \quad 0 \quad q_{x,j} \quad q_{y,j} \quad q_{z,j} \quad 0 \quad 0 \quad 0]$$

Nelle precedenti espressioni dei vettori dei carichi distribuiti, gli zeri corrispondono ai momenti (legati ai gradi di libertà rotazionali): eventuali momenti distribuiti lungo l'asse dell'elemento possono essere inseriti in modo analogo a quanto fatto per i carichi.

Il passaggio al vettore dei carichi in coordinate locali avviene con la seguente trasformazione:

forze equivalenti da globale a locale	$[q_L]^T = [T]\,[q_G]$	(2.2)

Calcolate le forze equivalenti $[f_L]$ come indicato nel §1.2, prima dell'assemblaggio è necessario trasformarle nel sistema globale tramite la seguente espressione.

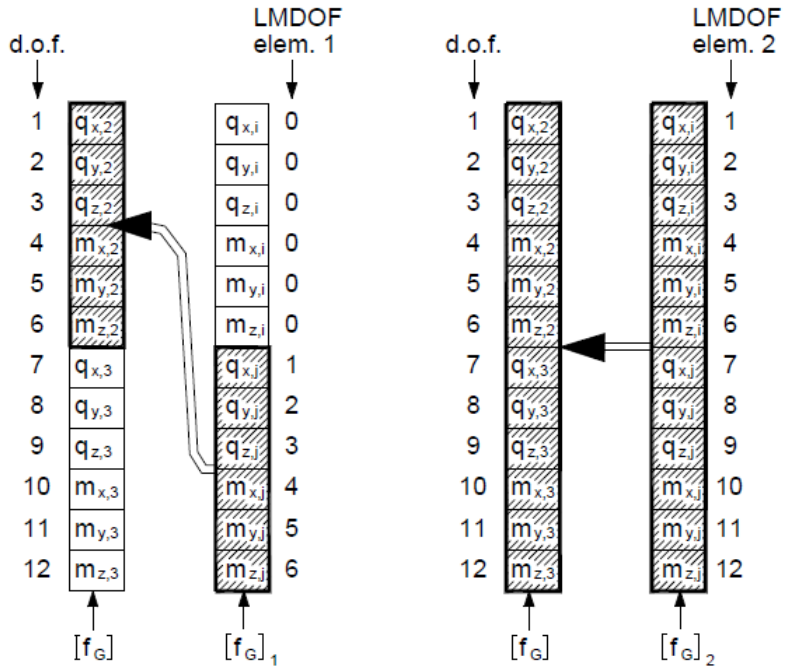
forze equivalenti da locale a globale	$[f_G]^T = [T]^T\,[f_L]$	(2.3)

Una volta trasformato in coordinate globali, il vettore delle forze generalizzate dell'elemento diventa del tipo:

$$[f_G]^T = [q_{x,i} \quad q_{y,i} \quad q_{z,i} \quad m_{x,i} \quad m_{y,i} \quad m_{z,i} \quad q_{x,j} \quad q_{y,j} \quad q_{z,j} \quad m_{x,j} \quad m_{y,j} \quad m_{z,j}]$$

I carichi concentrati applicati direttamente ai nodi, se riferiti al sistema di riferimento globale, non hanno bisogno di alcuna trasformazione e vengono inseriti direttamente nel vettore $[f_G]$. Infatti le forze generalizzate che compaiono nella precedente espressione di $[f_G]$ sono le stesse presentate nella figura del §1.1.

Con riferimento all'esempio 1, l'assemblaggio dei vettori delle forze generalizzate dei singoli elementi $[f_G]_i$ nel vettore $[f_G]$ del sistema risolvente, avviene utilizzando la matrice *LMDOF*() come esemplificato nell'immagine seguente.



Anche in questo caso naturalmente il fatto che il vettore globale dei carichi della struttura $[f_G]$ sia di ordine pari a quelli dei singoli elementi è un caso particolare legato alle caratteristiche dell'esempio. Il vettore $[f_G]_i$ dei carichi degli elementi è sempre di ordine 12, mentre quello della struttura ha ordine pari al numero totale di gradi di libertà (*Ndoft*).

Di seguito viene presentato l'assemblaggio presente all'interno del programma. Il vettore ***Rload***(*Idofn*), con *Idofn*=1÷*Ndoft*, è il vettore $[f_G]$ dei carichi generalizzati del sistema; il vettore ***FLOAD***(Ievab), con Ievab=1÷*NEVAB*, è il vettore $[f_G]_i$ dei carichi generalizzati dell'elemento i; *LMDOF*() è già stato precedentemente descritto.

```
For Ievab = 1 To NEVAB
    Ldofn = LMDOF(Ievab, Ielem)
    If Ldofn <> 0 Then Rload(Ldofn) += FLOAD(Ievab)
Next Ievab
```

2.5 Soluzione del sistema di equazioni

Il sistema di equazioni viene risolto con il metodo delle colonne attive o dello skyline, introdotto da K. J. Bathe in [1]. Si tratta di un metodo derivato dall'eliminazione di Gauss, come viene di seguito spiegato facendo ampio riferimento al testo originale. Rispetto a quanto riportato in [1] verranno sviluppati con maggior dettaglio alcuni passaggi degli esempi per consentire una migliore comprensione. Si sorvolerà invece su molti aspetti teorici, per i quali si rimanda al testo originale.

2.5.1 L'eliminazione di Gauss

Dato un sistema di equazioni lineari $[K][U] = [R]$, il metodo di Gauss prevede di combinare linearmente i coefficienti di $[K]$ e $[R]$ in modo da trasformare la matrice $[K]$ in una matrice triangolare superiore, cioè con elementi nulli al disotto della diagonale principale. In questo modo, procedendo dall'ultima equazione verso la prima, è possibile ottenere i valori di tutte le incognite.

In generale per eliminare l'elemento j della riga i si deve sottrarre alla riga i la riga i-1 moltiplicata per il rapporto

$$\frac{k_{i,j}}{k_{i-1,j}}$$

Con riferimento alle prime due righe della matrice $[K]$ riportata di seguito, per azzerare l'elemento k_{21} si deve sottrarre alla seconda riga la prima riga moltiplicata per k_{21}/k_{11}.

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} & . & . & k_{1n} \\ k_{21} & k_{22} & k_{23} & . & . & k_{2n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ & & & & & \end{bmatrix}$$

Risulterà infatti:

$$k_{21} = k_{21} - k_{11}\frac{k_{21}}{k_{11}} = 0$$

Gli altri coefficienti della riga 2, compreso il termine noto, assumeranno dei valori diversi da quelli originali.

Si risolve come esempio il seguente sistema, riportato nell'esempio 8.2.1 di [1].

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Per eliminare il termine k_{21} si sottrae dalla riga 2 la riga 1 moltiplicata per $-4/5$. I coefficienti della riga 1 moltiplicata per $-4/5$ risultano:

$$\begin{bmatrix} -4 & \frac{16}{5} & \frac{-4}{5} & 0 \end{bmatrix} [u_1] = [0]$$

Sottraendo i coefficienti così calcolati la riga 2 diventa:

$$\begin{bmatrix} 0 & \frac{14}{5} & \frac{-16}{5} & 1 \end{bmatrix} [u_2] = [1]$$

Si passa ora ad eliminare il termine k_{31} operando sempre sulla riga 1: si sottrae dalla riga 3 la riga 1 moltiplicata per $1/5$. I coefficienti della riga 1 moltiplicata per $1/5$ risultano:

$$\begin{bmatrix} 1 & \frac{-4}{5} & \frac{1}{5} & 0 \end{bmatrix} [u_1] = [0]$$

Sottraendo i coefficienti così calcolati la riga 3 diventa:

$$\begin{bmatrix} 0 & \frac{-16}{5} & \frac{29}{5} & -4 \end{bmatrix} [u_3] = [0]$$

Non è necessario eliminare il termine k_{41} che risulta già pari a zero.

Con queste operazioni sono stati eliminati i coefficienti della prima colonna dalla riga 2 all'ultima, e il sistema diventa:

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & 14/5 & -16/5 & 1 \\ 0 & -16/5 & 29/5 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Si procede eliminando la colonna 2 delle righe 3, 4 utilizzando la riga 2. Per eliminare il termine k_{32} si sottrae dalla riga 3 la riga 2 moltiplicata per $-16/5 \cdot 5/14 = -8/7$. I coefficienti della riga 2 moltiplicata per $-8/7$ risultano:

$$\begin{bmatrix} 0 & \frac{-16}{5} & \frac{128}{35} & \frac{-8}{7} \end{bmatrix} [u_2] = \begin{bmatrix} \frac{-8}{7} \end{bmatrix}$$

Sottraendo i coefficienti così calcolati la riga 3 diventa:

$$\begin{bmatrix} 0 & 0 & \frac{15}{7} & \frac{-20}{7} \end{bmatrix} [u_3] = \begin{bmatrix} \frac{8}{7} \end{bmatrix}$$

Si passa ora ad eliminare il termine k_{42} operando ancora sulla riga 2: si sottrae dalla riga 4 la riga 2 moltiplicata per $5/14$. I coefficienti della riga 2 moltiplicata per $5/14$ risultano:

$$\begin{bmatrix} 0 & 1 & \frac{-8}{7} & \frac{5}{14} \end{bmatrix} [u_2] = \begin{bmatrix} \frac{5}{14} \end{bmatrix}$$

Sottraendo i coefficienti così calcolati la riga 4 diventa:

$$\begin{bmatrix} 0 & 0 & \frac{-20}{7} & \frac{65}{14} \end{bmatrix} [u_4] = \begin{bmatrix} \frac{-5}{14} \end{bmatrix}$$

Con queste operazioni sono stati eliminati i coefficienti della seconda colonna dalle righe 3, 4, e il sistema diventa:

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & 14/5 & -16/5 & 1 \\ 0 & 0 & 15/7 & -20/7 \\ 0 & 0 & -20/7 & 65/14 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 8/7 \\ -5/14 \end{bmatrix}$$

Manca ora la sola eliminazione del termine k_{43}. Si sottrae dalla riga 4 la riga 3 moltiplicata per $-20/7 \cdot 7/15 = -4/3$. I coefficienti della riga 3 moltiplicata per $-4/3$ risultano:

$$\begin{bmatrix} 0 & 0 & \frac{-20}{7} & \frac{80}{21} \end{bmatrix} [u_3] = \begin{bmatrix} \frac{-32}{21} \end{bmatrix}$$

Sottraendo i coefficienti così calcolati la riga 4 diventa:

$$\begin{bmatrix} 0 & 0 & 0 & \frac{5}{6} \end{bmatrix} [u_4] = \begin{bmatrix} \frac{7}{6} \end{bmatrix}$$

Con queste operazioni sono stati eliminati i coefficienti della terza colonna dalla riga 4, e il sistema diventa:

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & 14/5 & -16/5 & 1 \\ 0 & 0 & 15/7 & -20/7 \\ 0 & 0 & 0 & 5/6 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 8/7 \\ 7/6 \end{bmatrix} \tag{2.4}$$

Ora è semplice calcolare le incognite partendo dall'ultima equazione, dove compare solo u_4, e procedendo con la sostituzione all'indietro (back-substitution).

$$u_4 = \frac{7}{6} \cdot \frac{6}{5} = \frac{7}{5}$$

$$u_3 = \frac{\frac{8}{7} + \frac{20}{7} \cdot u_4}{\frac{15}{7}} = \left(\frac{8}{7} + \frac{20}{7} \cdot \frac{7}{5}\right) \cdot \frac{7}{15} = \frac{12}{5}$$

$$u_2 = \frac{1 - u_4 + \frac{16}{5} \cdot u_3}{\frac{14}{5}} = \left(1 - \frac{7}{5} + \frac{16}{5} \cdot \frac{12}{5}\right) \cdot \frac{5}{14} = \frac{13}{5}$$

$$u_1 = \frac{-u_3 + 4\,u_2}{5} = \left(-\frac{12}{5} + 4 \cdot \frac{13}{5}\right) \cdot \frac{1}{5} = \frac{8}{5}$$

Quindi in sintesi:

$$[U] = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 8/5 \\ 13/5 \\ 12/5 \\ 7/5 \end{bmatrix} = \begin{bmatrix} 1.6 \\ 2.6 \\ 2.4 \\ 1.4 \end{bmatrix}$$

2.5.2 Formalizzazione matematica del metodo di Gauss

La riduzione della matrice $[K]$ in una matrice triangolare superiore può essere scritta come:

$$[L]_{n-1}^{-1} \cdots [L]_2^{-1} [L]_1^{-1} [K] = [S] \tag{2.5}$$

Dove i prodotti del primo termine devono essere condotti partendo da destra, cioè prima si esegue la $[L]_1^{-1}[K]$, poi si moltiplica $[L]_2^{-1}$ per il risultato della prima moltiplicazione e via di seguito.

Si ricorda la convenzione per cui nelle matrici gli elementi non indicati sono nulli.

La matrice $[L]_1^{-1}$ è quella che consente l'eliminazione dei coefficienti della colonna 1 a partire dalla riga 2 e vale:

$$[L]_1^{-1} = \begin{bmatrix} 1 & & & & & \\ -k_{21}/k_{11} & 1 & & & & \\ -k_{31}/k_{11} & & 1 & & & \\ \cdot & & & \cdot & & \\ \cdot & & & & \cdot & \\ \cdot & & & & & \cdot \\ -k_{n1}/k_{11} & & & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & & & \\ -l_{21} & 1 & & & & \\ -l_{31} & & 1 & & & \\ \cdot & & & \cdot & & \\ \cdot & & & & \cdot & \\ \cdot & & & & & \cdot \\ -l_{n1} & & & & & 1 \end{bmatrix}$$

La matrice $[L]_2^{-1}$ si calcola sui coefficienti k_{ij}^1 che risultano dalla prima moltiplicazione $[L]_1^{-1}[K]$, e vale:

$$[L]_2^{-1} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & -k_{32}^1/k_{22}^1 & 1 & & & \\ & -k_{42}^1/k_{22}^1 & & 1 & & \\ & \cdot & & & \cdot & \\ & \cdot & & & & \cdot \\ & -k_{n2}^1/k_{22}^1 & & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & -l_{32} & 1 & & & \\ & -l_{42} & & 1 & & \\ & \cdot & & & \cdot & \\ & \cdot & & & & \cdot \\ & -l_{n2} & & & & 1 \end{bmatrix}$$

La (2.5) può essere scritta in forma compatta come:

$$[L]^{-1}[K] = [S] \tag{2.6}$$

Poiché per matrici di questo tipo l'inversa si ottiene semplicemente invertendo i segni dei termini fuori dalla diagonale, la (2.6) si può scrivere come:

$$[K] = [L]_1[L]_2 \cdots [L]_{n-1}\,[S]$$

e cioè:

$$[K] = [L]\,[S] \tag{2.7}$$

dove:

$$[L] = \begin{bmatrix} 1 & & & & & \\ l_{21} & 1 & & & & \\ l_{31} & l_{32} & 1 & & & \\ \cdot & \cdot & & \cdot & & \\ \cdot & \cdot & & & \cdot & \\ \cdot & \cdot & & & & \cdot \\ l_{n1} & l_{n2} & \cdot & \cdot & \cdot & l_{n,n-1} \quad 1 \end{bmatrix}$$

Chiamando $[D]$ la matrice diagonale che contiene gli elementi diagonali di $[S]$, valgono le seguenti relazioni:

$$[K] = [L]\,[D]\,[L]^T \tag{2.8}$$

Questa decomposizione di $[K]$ viene sfruttata per risolvere il sistema con le seguenti espressioni:

$$[L]\,[V] = [R] \tag{2.9}$$

$$[D]\,[L]^T[U] = [V] \tag{2.10}$$

dove nella (2.9) il vettore $[R]$ dei carichi è ridotto per ottenere $[V]$ come:

$$[V] = [L]_{n-1}^{-1} \cdots [L]_2^{-1}[L]_1^{-1}[R] \tag{2.11}$$

E nella (2.10) la soluzione è ottenuta con una sostituzione all'indietro:

$$[L]^T[U] = [D]^{-1}[V] \tag{2.12}$$

Si riprende l'esempio 8.2.1 di [1], risolto nel paragrafo precedente: viene triangolarizzata la matrice $[K]$, trasformandola nella matrice $[S]$, utilizzando il metodo appena presentato.

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$[L]_1^{-1} = \begin{bmatrix} 1 & & & \\ 4/5 & 1 & & \\ -1/5 & & 1 & \\ 0 & & & 1 \end{bmatrix}$$

$$[L]_1^{-1}[K] = \begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & 14/5 & -16/5 & 1 \\ 0 & -16/5 & 29/5 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix}$$

$$[L]_2^{-1} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & (16\cdot 5)/(5\cdot 14) & 1 & \\ 0 & -5/14 & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 8/7 & 1 & \\ 0 & -5/14 & & 1 \end{bmatrix}$$

$$[L]_2^{-1}([L]_1^{-1}[K]) = \begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & 14/5 & -16/5 & 1 \\ 0 & 0 & 15/7 & -20/7 \\ 0 & 0 & -20/7 & 65/14 \end{bmatrix}$$

$$[L]_3^{-1} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & (20\cdot 7)/(7\cdot 15) & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & 4/3 & 1 \end{bmatrix}$$

$$[L]_3^{-1}\{[L]_2^{-1}([L]_1^{-1}[K])\} = \begin{bmatrix} 5 & -4 & 1 & 0 \\ 0 & 14/5 & -16/5 & 1 \\ 0 & 0 & 15/7 & -20/7 \\ 0 & 0 & 0 & 5/6 \end{bmatrix} = [S] \qquad (2.13)$$

La matrice $[S]$ è proprio la stessa matrice ottenuta in (2.4) con un diverso procedimento.

Combinando le (2.7) e (2.8) si ricava anche:

$$[L]\,[S] = [L]\,[D]\,[L]^T$$
$$[S] = [D]\,[L]^T \qquad (2.14)$$

In questo esempio $[D]$ e $[L]^T$ valgono:

$$[D] = \begin{bmatrix} 5 & & & \\ & 14/5 & & \\ & & 15/7 & \\ & & & 5/6 \end{bmatrix}$$

$$[L]^T = \begin{bmatrix} 1 & -4/5 & 1/5 & 0 \\ & 1 & -8/7 & 5/14 \\ & & 1 & -4/3 \\ & & & 1 \end{bmatrix}$$

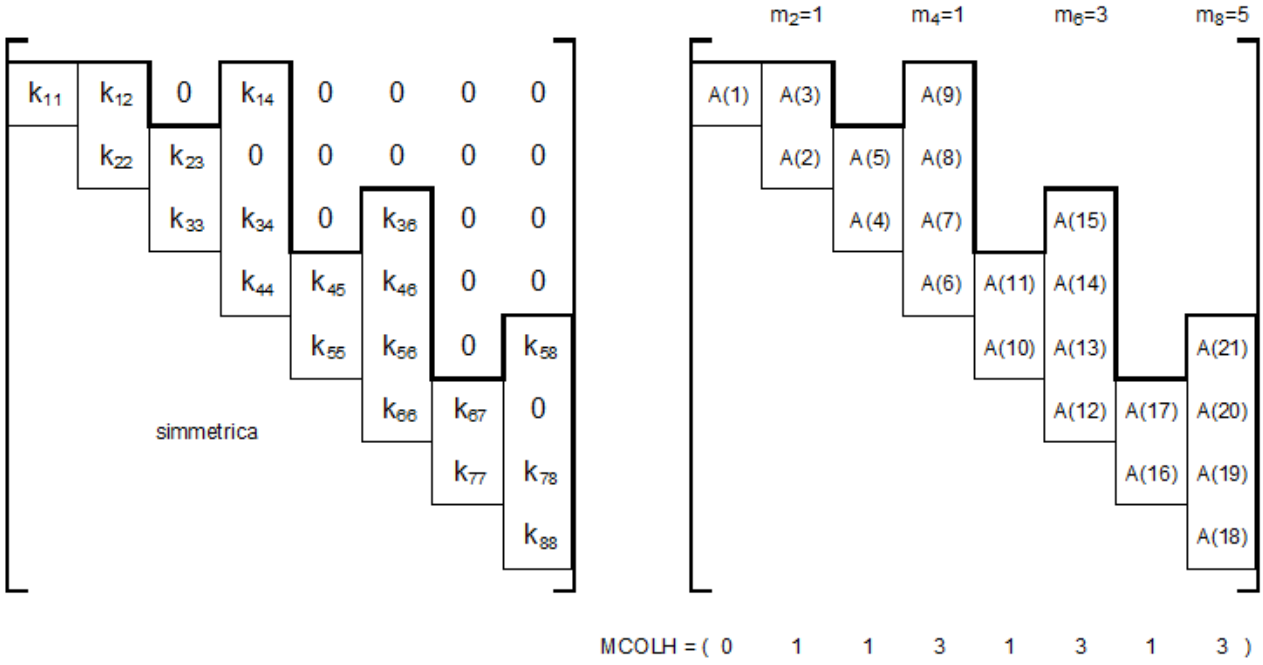
Gli elementi di $[L]^T$ sono quelli di $[L]_i^{-1}$ calcolati nei vari passaggi, cambiati di segno. Eseguendo il prodotto $[D]\,[L]^T$ si vede che la matrice $[S]$ è proprio quelle della (2.13). Di seguito si riporta come verifica il calcolo eseguito online nel sito https://www.youmath.it/.

$$\begin{pmatrix} 5 & -4 & 1 & 0 \\ 0 & \frac{14}{5} & -\frac{16}{5} & 1 \\ 0 & 0 & \frac{15}{7} & -\frac{20}{7} \\ 0 & 0 & 0 & \frac{5}{6} \end{pmatrix} = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & \frac{14}{5} & 0 & 0 \\ 0 & 0 & \frac{15}{7} & 0 \\ 0 & 0 & 0 & \frac{5}{6} \end{pmatrix} \cdot \begin{pmatrix} 1 & -\frac{4}{5} & \frac{1}{5} & 0 \\ 0 & 1 & -\frac{8}{7} & \frac{5}{14} \\ 0 & 0 & 1 & -\frac{4}{3} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.5.3 Il metodo skyline

Con questo metodo la decomposizione $[L]\,[D]\,[L]^T$ viene condotta per colonne anziché per righe, utilizzando il vettore $[A]$ presentato nel §1.4 e riportato anche nella figura seguente.

Nel §2.2 sono già stati definiti i vettori *MAXAD*() e *MCOLH*(), che contengono rispettivamente gli indirizzi degli elementi della diagonale principale di $[K]$ e le altezze delle colonne attive. Per utilizzare le formule della soluzione del sistema con il metodo skyline, è necessario introdurre una nuova grandezza m_j, definita come il numero di riga del primo elemento diverso da zero nella colonna j della matrice $[K]$. Nella figura seguente sono riportati i valori di m_j per le colonne j=2, 4, 6, 8. I valori m_j, con j=1÷n, definiscono la skyline.



Lavorando per colonne, invece di calcolare gli elementi di $[L]$ come nel paragrafo precedente, vengono calcolati gli elementi di $[L]^T$. Vengono di seguito riportate le formule per calcolare gli elementi di $[L]^T$ e di $[D]$. Si devono prima calcolare dei coefficienti intermedi g_{ij} con le seguenti espressioni:

$$\left.\begin{aligned} g_{mj,j} &= k_{mj,j} \\ g_{ij} &= k_{ij} - \sum_{r=m_m}^{i-1} l_{ri}\, g_{rj} \quad i = m_j + 1,\ \ldots, j-1 \end{aligned}\right\} \tag{2.15}$$

dove: $m_m = max(m_i, m_j)$

Con i valori di g_{ij} è possibile calcolare l_{ij} e d_{ij} con:

$$l_{ij} = \frac{g_{ij}}{d_{ii}} \qquad i = m_j,\ \ldots, j-1 \tag{2.16}$$

$$d_{jj} = k_{jj} - \sum_{r=m_j}^{j-1} l_{rj}\, g_{rj} \tag{2.17}$$

Nelle (2.15), (2.16), (2.17) **j = 2÷n**, dove n è l'ordine della matrice. Il valore di d_{11} è sempre uguale a k_{11}: $\boldsymbol{d_{11} = k_{11}}$.

I valori l_{ij} e d_{ij} così calcolati possono andare a sostituire direttamente i valori k_{ij} della matrice $[K]$ (e nella pratica del vettore $[A]$), in quanto ad ogni passo i valori coinvolti nelle operazioni sono il risultato delle operazioni precedenti. Nel paragrafo precedente si è visto infatti che $[L]_2^{-1}$ opera sul

risultato di $[L]_1^{-1}[K]$, cioè si esegue il prodotto: $[L]_2^{-1}([L]_1^{-1}[K])$. $[L]_3^{-1}$ opera sul risultato di $[L]_2^{-1}([L]_1^{-1}[K])$ e così via. Anche i termini intermedi g_{ij} possono andare direttamente ad occupare le posizioni k_{ij}, senza bisogno di occupare locazioni di memoria dedicate.

Per ottenere la riduzione (2.11) del vettore dei carichi $[R]$, **partendo dal primo valore fisso $\boldsymbol{V_1} = \boldsymbol{R_1}$**, si usa la seguente espressione:

$$V_i = R_i - \sum_{r=m_i}^{i-1} l_{ri} V_r \qquad i = 2 \div n \tag{2.18}$$

Gli elementi del vettore $[V]$ vanno direttamente a sostituire quelli del vettore $[R]$, senza bisogno di riservare apposite allocazioni di memoria.

Prima di eseguire la sostituzione all'indietro (back-substitution) si calcolano i termini $[\bar{V}] = [D]^{-1}[V]$ della (2.12) con la:

$$\bar{V}_i = \frac{V_i}{d_{ii}} \qquad i = 1 \div n \tag{2.19}$$

Infine si esegue la back-substitution con l'espressione (2.20), assegnando inizialmente al vettore degli spostamenti generalizzati i valori di $[\bar{V}]$: si pone cioè $[U] = [\bar{V}]$. L'ultima incognita è già stata determinata nel passo precedente, cioè $\boldsymbol{U_n} = \boldsymbol{\bar{V}_n}$. La (2.20) deve essere calcolata per **i = n÷2, con passo -1**.

$$U_r = U_r - l_{ri} U_i \qquad r = m_i \div i - 1 \tag{2.20}$$

Per ognuno dei valori dell'indice i, le (2.20) modificano i valori di U_j per j<i. Anche in questo processo i valori che vengono via via calcolati vanno a sostituire quelli corrispondenti nel vettore $[R]$. Alla fine del processo i carichi generalizzati del vettore $[R]$ risultano sostituiti dagli spostamenti generalizzati $[U]$, che sono le incognite del problema.

La riduzione del vettore $[R]$ può essere eseguita contemporaneamente alla decomposizione della matrice $[K]$, ovvero può anche essere eseguita dopo avere decomposto $[K]$. Infatti le (2.18), (2.19), (2.20) utilizzano i termini l_{ij} e d_{ii} che alla fine della decomposizione sono contenuti nella matrice $[K]$.

Si sottolinea il fatto che con il metodo skyline non solo si risparmiano posizioni di memoria, ma che le operazioni per la soluzione del sistema di equazioni vengono eseguite sui soli elementi della matrice di rigidezza al disotto della skyline.

Per comprendere il meccanismo presentato si risolve il seguente sistema, riportato nell'esempio 8.4 di [1]. Si tratta di un esempio che evidenzia l'efficacia del metodo, che prevede di eseguire operazioni sui soli termini al disotto della skyline. **La matrice è simmetrica** ma viene riportata e trattata la sola parte al disopra della diagonale principale.

$$\begin{bmatrix} 2 & -2 & & & -1 \\ & 3 & -2 & & 0 \\ & & 5 & -3 & 0 \\ & & & 10 & 4 \\ & & & & 10 \end{bmatrix}$$

La prima operazione è fissa: $\boldsymbol{d_{11}} = k_{11} = 2$.

Ora si deve eseguire un loop con j = 2÷n.

j=2 (operazioni sulla colonna 2)

Il primo elemento diverso da zero nella colonna j=2 si trova nella riga 1, pertanto **m_2=1**.

Dalla prima delle (2.15) risulta $g_{m_j,j} = k_{m_j,j}$, cioè $\boldsymbol{g_{12}} = k_{12} = -2$.

La seconda delle (2.15) va applicata per $i = m_j + 1 \div j - 1 = 2 \div 1$, e poiché il secondo indice è minore del primo non ci sono altri g_{ij}.

La (2.16) va applicata per $i = m_j \div j - 1 = 1 \div 1$, e cioè solo per $i = 1$.

Si ottiene $l_{ij} = \boldsymbol{l_{12}} = g_{12}/d_{11} = -2/2 = -1$

Dalla (2.17):

$$d_{jj} = \boldsymbol{d_{22}} = k_{22} - \sum_{r=m_j=1}^{j-1=1} l_{rj}\, g_{rj} = k_{22} - l_{12}\, g_{12} = 3 - (-1)(-2) = 1$$

Dopo le operazioni sulla seconda colonna la matrice diventa:

$$\begin{bmatrix} d_{11} & l_{12} & | & & & -1 \\ & d_{22} & | & -2 & & 0 \\ & & | & 5 & -3 & 0 \\ & & | & & 10 & 4 \\ & & | & & & 10 \end{bmatrix} = \begin{bmatrix} 2 & -1 & | & & & -1 \\ & 1 & | & -2 & & 0 \\ & & | & 5 & -3 & 0 \\ & & | & & 10 & 4 \\ & & | & & & 10 \end{bmatrix}$$

j=3 (operazioni sulla colonna 3)

Il primo elemento diverso da zero nella colonna j=3 si trova nella riga 2, pertanto $\mathbf{m_3=2}$.

Dalla prima delle (2.15) risulta $g_{m_j,j} = k_{m_j,j}$, cioè $\boldsymbol{g_{23}} = k_{23} = -2$.

La seconda delle (2.15) va applicata per $i = m_j + 1 \div j - 1 = 3 \div 2$, ma poiché il secondo indice è minore del primo non ci sono altri g_{ij}.

La (2.16) va applicata per $i = m_j \div j - 1 = 2 \div 2$, e cioè solo per $i = 2$.

Si ottiene $l_{ij} = \boldsymbol{l_{23}} = g_{23}/d_{22} = -2/1 = -2$

Dalla (2.17):

$$d_{jj} = \boldsymbol{d_{33}} = k_{33} - \sum_{r=m_j=2}^{j-1=2} l_{rj}\, g_{rj} = k_{33} - l_{23}\, g_{23} = 5 - (-2)(-2) = 1$$

Dopo le operazioni sulla terza colonna la matrice diventa:

$$\begin{bmatrix} d_{11} & l_{12} & & | & & -1 \\ & d_{22} & l_{23} & | & & 0 \\ & & d_{33} & | & -3 & 0 \\ & & & | & 10 & 4 \\ & & & | & & 10 \end{bmatrix} = \begin{bmatrix} 2 & -1 & & | & & -1 \\ & 1 & -2 & | & & 0 \\ & & 1 & | & -3 & 0 \\ & & & | & 10 & 4 \\ & & & | & & 10 \end{bmatrix}$$

j=4 (operazioni sulla colonna 4)

Il primo elemento diverso da zero nella colonna j=4 si trova nella riga 3, pertanto $\mathbf{m_4=3}$.

Dalla prima delle (2.15) risulta $g_{m_j,j} = k_{m_j,j}$, cioè $\boldsymbol{g_{34}} = k_{34} = -3$.

La seconda delle (2.15) va applicata per $i = m_j + 1 \div j - 1 = 4 \div 3$, e poiché il secondo indice è minore del primo non ci sono altri g_{ij}.

La (2.16) va applicata per $i = m_j \div j - 1 = 3 \div 3$, e cioè solo per $i = 3$.

Si ottiene $l_{ij} = \boldsymbol{l_{34}} = g_{34}/d_{33} = -3/1 = -3$

Dalla (2.17):

$$d_{jj} = \boldsymbol{d_{44}} = k_{44} - \sum_{r=m_j=3}^{j-1=3} l_{rj}\, g_{rj} = k_{44} - l_{34}\, g_{34} = 10 - (-3)(-3) = 1$$

Dopo le operazioni sulla quarta colonna la matrice diventa:

$$\left[\begin{array}{ccccc|c} d_{11} & l_{12} & & & & -1 \\ & d_{22} & l_{23} & & & 0 \\ & & d_{33} & l_{34} & & 0 \\ & & & d_{44} & & 4 \\ & & & & & 10 \end{array}\right] = \left[\begin{array}{ccccc|c} 2 & -1 & & & & -1 \\ & 1 & -2 & & & 0 \\ & & 1 & -3 & & 0 \\ & & & 1 & & 4 \\ & & & & & 10 \end{array}\right]$$

j=5 (operazioni sulla colonna 5)

Il primo elemento diverso da zero nella colonna j=5 si trova nella riga 1, pertanto $\mathbf{m_5=1}$.

Dalla prima delle (2.15) risulta $g_{m_j,j} = k_{m_j,j}$, cioè $\boldsymbol{g_{15}} = k_{15} = -1$.

La seconda delle (2.15) va applicata per $i = m_j + 1 \div j - 1 = 2 \div 4$. Per ridurre questa colonna è necessario calcolare alcuni valori intermedi g_{ij}: infatti ragionando sui soli elementi al disopra della diagonale principale, gli elementi della colonna 5 sono "collegati" agli elementi delle colonne precedenti. Procedendo con il calcolo degli altri g_{ij} si ottiene:

i=2 $m_i = m_2 = 1; \qquad m_j = m_5 = 1; \qquad m_m = max(m_i, m_j) = 1$

$$g_{ij} = \boldsymbol{g_{25}} = k_{ij} - \sum_{r=m_m=1}^{i-1=1} l_{ri}\, g_{rj} = k_{25} - l_{12}\, g_{15} = 0 - (-1)(-1) = -1$$

i=3 $m_i = m_3 = 2; \qquad m_j = m_5 = 1; \qquad m_m = max(m_i, m_j) = 2$

$$g_{ij} = \boldsymbol{g_{35}} = k_{ij} - \sum_{r=m_m=2}^{i-1=2} l_{ri}\, g_{rj} = k_{35} - l_{23}\, g_{25} = 0 - (-2)(-1) = -2$$

i=4 $m_i = m_4 = 3; \qquad m_j = m_5 = 1; \qquad m_m = max(m_i, m_j) = 3$

$$g_{ij} = \boldsymbol{g_{45}} = k_{ij} - \sum_{r=m_m=3}^{i-1=3} l_{ri}\, g_{rj} = k_{45} - l_{34}\, g_{35} = 4 - (-3)(-2) = -2$$

La (2.16) va applicata per $i = m_j \div j - 1 = 1 \div 4$, e quindi:

i=1 $l_{ij} = \boldsymbol{l_{15}} = g_{15}/d_{11} = -1/2 = -0.5$

i=2 $l_{ij} = \boldsymbol{l_{25}} = g_{25}/d_{22} = -1/1 = -1$

i=3 $l_{ij} = \boldsymbol{l_{35}} = g_{35}/d_{33} = -2/1 = -2$

i=4 $l_{ij} = \boldsymbol{l_{45}} = g_{45}/d_{44} = -2/1 = -2$

Dalla (2.17):

$$d_{jj} = \boldsymbol{d_{55}} = k_{55} - \sum_{r=m_j=1}^{j-1=4} l_{rj}\, g_{rj} = k_{55} - l_{15}\, g_{15} - l_{25}\, g_{25} - l_{35}\, g_{35} - l_{45}\, g_{45}$$

$$\boldsymbol{d_{55}} = 10 - (-0.5)(-1) - (-1)(-1) - (-2)(-2) - (-2)(-2) = 0.5$$

Dopo le operazioni sulla quinta colonna la matrice diventa:

$$\left[\begin{array}{ccccc} d_{11} & l_{12} & & & l_{15} \\ & d_{22} & l_{23} & & l_{25} \\ & & d_{33} & l_{34} & l_{35} \\ & & & d_{44} & l_{45} \\ & & & & d_{55} \end{array}\right] = \left[\begin{array}{ccccc} 2 & -1 & & & -0.5 \\ & 1 & -2 & & -1 \\ & & 1 & -3 & -2 \\ & & & 1 & -2 \\ & & & & 0.5 \end{array}\right] \qquad (2.21)$$

Per agevolare ulteriormente la comprensione del metodo, si riporta di seguito lo schema completo delle operazioni svolte sulle colonne. I coefficienti l_{ij} e d_{jj} vanno a sostituire gli elementi k_{ij} e k_{jj} senza bisogno quindi di riservare apposite posizioni di memoria.

j=1	***j=2***	***j=3***	***j=4***	***j=5***
	$g_{12} = k_{12}$	$g_{23} = k_{23}$	$g_{34} = k_{34}$	$g_{15} = k_{15}$ $g_{25} = k_{25} - l_{12}g_{15}$ $g_{35} = k_{35} - l_{23}g_{25}$ $g_{45} = k_{45} - l_{34}g_{35}$
$d_{11} = k_{11}$	$l_{12} = g_{12}/d_{11}$			$l_{15} = g_{15}/d_{11}$
	$d_{22} = k_{22} - l_{12}g_{12}$	$l_{23} = g_{23}/d_{22}$		$l_{25} = g_{25}/d_{22}$
		$d_{33} = k_{33} - l_{23}g_{23}$	$l_{34} = g_{34}/d_{33}$	$l_{35} = g_{35}/d_{33}$
			$d_{44} = k_{44} - l_{34}g_{34}$	$l_{45} = g_{45}/d_{44}$
				$d_{55} = k_{55} - \sum_{r=1}^{4} l_{r5}g_{r5}$

In questo esempio $[D]$ e $[L]^T$ valgono:

$$[D] = \begin{bmatrix} 2 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 0.5 \end{bmatrix} \qquad [L]^T = \begin{bmatrix} 1 & -1 & & & -0.5 \\ & 1 & -2 & & -1 \\ & & 1 & -3 & -2 \\ & & & 1 & -2 \\ & & & & 0.5 \end{bmatrix}$$

Con questo metodo vengono calcolati direttamente i coefficienti l_{ij} di $[L]^T$ e quindi non si devono cambiare i segni dei coefficienti calcolati (nel paragrafo precedente venivano calcolati i coefficienti di $[L]_i^{-1}$ e quindi si doveva cambiare il segno per invertire la matrice).

Eseguendo il prodotto $[D]\,[L]^T$ si ottiene la matrice $[S]$:

$$[S] = \begin{bmatrix} 2 & -2 & & & -1 \\ & 1 & -2 & & -1 \\ & & 1 & -3 & -2 \\ & & & 1 & -2 \\ & & & & 0.5 \end{bmatrix} \tag{2.22}$$

In questo caso la matrice che risulta dalle riduzioni sulle colonne (2.21) non coincide con la matrice $[S]$. Di seguito si riporta il calcolo $[S] = [D]\,[L]^T$ eseguito online nel sito https://www.youmath.it/.

$$\begin{pmatrix} 2 & -2 & 0 & 0 & -1 \\ 0 & 1 & -2 & 0 & -1 \\ 0 & 0 & 1 & -3 & -2 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 0 & 0 & -0.5 \\ 0 & 1 & -2 & 0 & -1 \\ 0 & 0 & 1 & -3 & -2 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Si può verificare che con l'eliminazione diretta di Gauss esposta nel §2.5.1 si ottiene proprio la matrice $[S]$ della (2.22). Di seguito vengono riportati sinteticamente i passaggi per un controllo di quanto eseguito.

$$\begin{bmatrix} 2 & -2 & & & -1 \\ -2 & 3 & -2 & & 0 \\ & -2 & 5 & -3 & 0 \\ & & -3 & 10 & 4 \\ -1 & & & 4 & 10 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

riga 1 · (− 2/2): $[-2 \quad 2 \quad 0 \quad 0 \quad 1] = [0]$ = riga1*

riga 2 – riga1*: $[\ 0 \quad 1 \quad -2 \quad 0 \quad -1] = [1]$ (eliminata la colonna 1 dalla riga 2)

riga 1 · (− 1/2): $[-1 \quad 1 \quad 0 \quad 0 \quad 0.5] = [0]$ = riga1*

riga 5 – riga1*: $[\ 0 \quad -1 \quad 0 \quad 4 \quad 9.5] = [-0.5]$ (eliminata la colonna 1 dalla riga 5)

ora la matrice risulta:

$$\begin{bmatrix} 2 & -2 & & & -1 \\ & 1 & -2 & & -1 \\ & -2 & 5 & -3 & 0 \\ & & -3 & 10 & 4 \\ & -1 & 0 & 4 & 9.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

riga 2 · (− 2/1): $[\ 0 \quad -2 \quad 4 \quad 0 \quad 2] = [-2]$ = riga2*

riga 3 – riga2*: $[\ 0 \quad 0 \quad 1 \quad -3 \quad -2] = [\ 2]$ (eliminata la colonna 2 dalla riga 3)

riga 2 · (− 1/1): $[\ 0 \quad -1 \quad 2 \quad 0 \quad 1] = [-1]$ = riga2*

riga 5 – riga2*: $[\ 0 \quad 0 \quad -2 \quad 4 \quad 8.5] = [\ 1]$ (eliminata la colonna 2 dalla riga 5)

ora la matrice risulta:

$$\begin{bmatrix} 2 & -2 & & & -1 \\ & 1 & -2 & & -1 \\ & & 1 & -3 & -2 \\ & & -3 & 10 & 4 \\ & & -2 & 4 & 8.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

riga 3 · (− 3/1): $[\ 0 \quad 0 \quad -3 \quad 9 \quad 6] = [-6]$ = riga3*

riga 4 – riga3*: $[\ 0 \quad 0 \quad 0 \quad 1 \quad -2] = [\ 6]$ (eliminata la colonna 3 dalla riga 4)

riga $3 \cdot (-2/1)$: $[\,0 \quad 0 \quad -2 \quad 6 \quad 4] = [-4]$ = riga3*
riga 5 – riga3*: $[\,0 \quad 0 \quad 0 \quad -2 \quad 4.5] = [\,5]$ (eliminata la colonna 3 dalla riga 5)
ora la matrice risulta:

$$\begin{bmatrix} 2 & -2 & & & -1 \\ & 1 & -2 & & -1 \\ & & 1 & -3 & -2 \\ & & & 1 & -2 \\ & & & -2 & 4.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 6 \\ 5 \end{bmatrix}$$

riga $4 \cdot (-2/1)$: $[\,0 \quad 0 \quad 0 \quad -2 \quad 4] = [-12]$ = riga4*
riga 5 – riga4*: $[\,0 \quad 0 \quad 0 \quad 0 \quad 0.5] = [\,17]$ (eliminata la colonna 4 dalla riga 5)
il risultato finale dell'eliminazione diretta di Gauss è il seguente, da cui si vede che la matrice $[S]$ è proprio quella della (2.22).

$$\begin{bmatrix} 2 & -2 & & & -1 \\ & 1 & -2 & & -1 \\ & & 1 & -3 & -2 \\ & & & 1 & -2 \\ & & & & 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 6 \\ 17 \end{bmatrix}$$

Si riprende l'applicazione del metodo skyline completando la soluzione del sistema, riducendo il vettore delle forze generalizzate $[R]^T = [0 \quad 1 \quad 0 \quad 0 \quad 0]$, come nell'esempio 8.5 di [1]. Si utilizzano le (2.18), (2.19), (2.20).

Si inizia calcolando il vettore $[V]$ con la (2.18), con $i = 2 \div n = 2 \div 5$, partendo dal valore fisso $V_1 = r_1 = 0$.

i=2 $m_i = m_2 = 1; \qquad i - 1 = 1; \qquad r = 1 \div 1$

$$S = \sum_{r=1}^{1} l_{ri} V_r = l_{12} V_1 = -1 \cdot 0 = 0$$

$$V_2 = R_2 - S = 1 - 0 = 1$$

i=3 $m_i = m_3 = 2; \qquad i - 1 = 2; \qquad r = 2 \div 2$

$$S = \sum_{r=2}^{2} l_{ri} V_r = l_{23} V_2 = -2 \cdot 1 = -2$$

$$V_3 = R_3 - S = 0 - (-2) = 2$$

i=4 $m_i = m_4 = 3; \qquad i - 1 = 3; \qquad r = 3 \div 3$

$$S = \sum_{r=3}^{3} l_{ri} V_r = l_{34} V_3 = -3 \cdot 2 = -6$$

$$V_4 = R_4 - S = 0 - (-6) = 6$$

i=5 $m_i = m_5 = 1; \qquad i - 1 = 4; \qquad r = 1 \div 4$

$$S = \sum_{r=1}^{4} l_{ri} V_r = l_{15} V_1 + l_{25} V_2 + l_{35} V_3 + l_{45} V_4$$

$$S = -0.5 \cdot 0 + (-1) \cdot 1 + (-2) \cdot 2 + (-2) \cdot 6 = -17$$

$$V_5 = R_5 - S = 0 - (-17) = 17$$

Si riporta per comodità il vettore trasposto $[V]^T$ che contiene i seguenti valori:

$$[V]^T = [0 \quad 1 \quad 2 \quad 6 \quad 17] \qquad (2.23)$$

Si calcola ora il vettore $[\bar{V}]$ con la (2.19), con $i = 1 \div n = 1 \div 5$.

i=1 $\bar{V}_i = \bar{V}_1 = V_1/d_{11} = 0/2 = 0$

i=2 $\bar{V}_i = \bar{V}_2 = V_2/d_{22} = 1/1 = 1$

i=3 $\bar{V}_i = \bar{V}_3 = V_3/d_{33} = 2/1 = 2$

i=4 $\bar{V}_i = \bar{V}_4 = V_4/d_{44} = 6/1 = 6$

i=5 $\bar{V}_i = \bar{V}_5 = V_5/d_{55} = 17/0.5 = 34$

Il vettore trasposto $[\bar{V}]^T$ appena calcolato contiene i seguenti valori:

$$[\bar{V}]^T = [0 \quad 1 \quad 2 \quad 6 \quad 34] \tag{2.24}$$

Ora manca solo la back-substitution da eseguire con la (2.20), seguendo i seguenti punti:

1. si devono trasferire i valori di $[\bar{V}]$ *su* $[U]$, cioè: $[U]^T = [0 \quad 1 \quad 2 \quad 6 \quad 34]$
2. l'ultima incognita $U_{n=5}$ risulta già dal passo precedente, cioè $\boldsymbol{U_5 = 34}$
3. si devono eseguire i seguenti cicli:

```
For i = n To 2 Step -1
    For r = mᵢ To i - 1
        U_r = U_r − l_ri U_i
    Next r
Next i
```

i=5 $m_i = m_5 = 1; \quad i-1 = 4; \quad r = 1 \div 4$

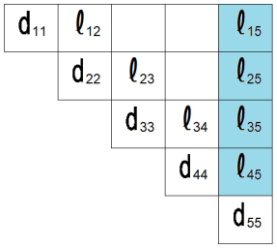
si opera sulla colonna 5 e si determina U_4

$U_1 = U_1 - l_{15}U_5 = 0 - (-0.5) \cdot 34 = 17$

$U_2 = U_2 - l_{25}U_5 = 1 - (-1) \cdot 34 = 35$

$U_3 = U_3 - l_{15}U_5 = 2 - (-2) \cdot 34 = 70$

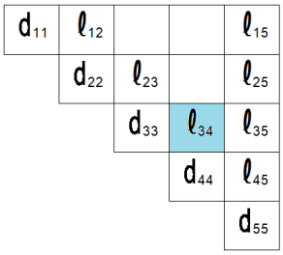
$\boldsymbol{U_4} = U_4 - l_{15}U_5 = 6 - (-2) \cdot 34 = \mathbf{74}$



i=4 $m_i = m_4 = 3; \quad i-1 = 3; \quad r = 3 \div 3$

si opera sulla colonna 4 e si determina U_3

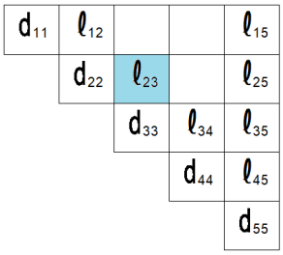
$\boldsymbol{U_3} = U_3 - l_{34}U_4 = 70 - (-3) \cdot 74 = \mathbf{292}$



i=3 $m_i = m_3 = 2; \quad i-1 = 2; \quad r = 2 \div 2$

si opera sulla colonna 3 e si determina U_2

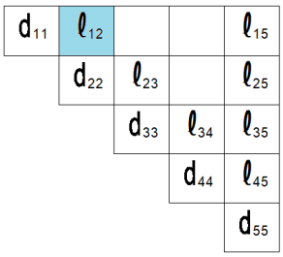
$\boldsymbol{U_2} = U_2 - l_{23}U_3 = 35 - (-2) \cdot 292 = \mathbf{619}$



i=2 $m_i = m_2 = 1; \quad i-1 = 1; \quad r = 1 \div 1$

si opera sulla colonna 2 e si determina U_1

$\boldsymbol{U_1} = U_1 - l_{12}U_2 = 17 - (-1) \cdot 619 = \mathbf{636}$



La soluzione del sistema è pertanto:

$$[U]^T = [636 \quad 619 \quad 292 \quad 74 \quad 34]$$

Vengono di seguito presentate le routine che eseguono la triangolarizzazione (COLSOL) e la riduzione del vettore dei carichi seguita dalla sostituzione all'indietro (BACKSU). A parte la traduzione e la modernizzazione del codice, si tratta delle routine presentate in [1].

```
    Sub COLSOL(File1 As StreamWriter)

        Dim Icount, k, Idofn, KN, KL, KU, KH, KLT, KI, Nevar, KK As Integer
        Dim B, C As Double
        Dim Element As Integer '22.07.08
        Dim D_O_F$ = ""
        Dim Text As String

        ' *** TRIANGULARIZATION OF STIFFNESS MATRIX

        For Idofn = 1 To NDOFT
            KN = MAXAD(Idofn)
            KL = KN + 1
            KU = MAXAD(Idofn + 1) - 1
            KH = KU - KL

            If KH > 0 Then
                k = Idofn - KH
                Icount = 0
                KLT = KU

                For j = 1 To KH
                    Icount += 1
                    KLT -= 1
                    KI = MAXAD(k)
                    Nevar = MAXAD(k + 1) - KI - 1
                    If Nevar > 0 Then
                        KK = MIN0(Icount, Nevar)
                        C = 0.0
                        For L = 1 To KK
                            C += GLOBK(KI + L) * GLOBK(KLT + L)
                        Next L
                        GLOBK(KLT) -= C
                    End If
                    k += 1
                Next j
            End If

            If KH >= 0 Then
                k = Idofn
                B = 0.0
                For KK = KL To KU
                    k -= 1
                    KI = MAXAD(k)
                    C = GLOBK(KK) / GLOBK(KI)
                    B += C * GLOBK(KK)
                    GLOBK(KK) = C
                Next KK
                GLOBK(KN) -= B
            End If

            If GLOBK(KN) <= 0 Then
                Call FindDOF(Idofn, Element, D_O_F$) '22.07.08

                Text = "*** FATAL ERROR: NON POSITIVE PIVOT FOR EQUATION "
                Text +=  Str$(Idofn) + vbCrLf
```

```
            Text += "Element: " + Str$(Element) + vbCrLf
            Text += "d.o.f. :  " + D_O_F$
            MsgBox(Text, vbExclamation, "Warning")

            PrtString = vbCrLf : PrtString += vbCrLf
            PrtString += "*** FATAL ERROR: NON POSITIVE PIVOT FOR EQUATION "
            PrtString +=  Str$(Idofn) + vbCrLf
            PrtString += "         " + "Element: " + Str$(Element) + vbCrLf
            PrtString += "         " + "d.o.f. :  " + D_O_F$
            File1.WriteLine(PrtString)
            Errore = True
            Exit Sub
        End If
    Next Idofn

End Sub
```

```
Sub BACKSU(Icase As Integer)

    Dim Idofn, Kl, Ku, k As Long
    Dim C As Double

    ' *** REDUCTION OF R.H.S. LOAD VECTOR

    For Idofn = 1 To NDOFT
        Kl = MAXAD(Idofn) + 1
        Ku = MAXAD(Idofn + 1) - 1
        If (Ku - Kl) >= 0 Then
            k = Idofn
            C = 0.0

            For KK = Kl To Ku
                k -= 1
                C += GLOBK(KK) * Rload(k)
            Next KK

            Rload(Idofn) -= C
        End If
    Next Idofn

    For Idofn = 1 To NDOFT
        k = MAXAD(Idofn)
        Rload(Idofn) /= GLOBK(k)
    Next Idofn

    If NDOFT <> 1 Then

        ' *** BACK-SUBSTITUTION

        For Idofn = NDOFT To 2 Step -1
            Kl = MAXAD(Idofn) + 1
            Ku = MAXAD(Idofn + 1) - 1

            If Ku - Kl >= 0 Then
                k = Idofn
```

```
                    For KK = Kl To Ku
                        k = k - 1
                        Rload(k) -= GLOBK(KK) * Rload(Idofn)
                    Next KK
                End If
            Next Idofn
        End If

        For Ipoin = 1 To Npoin
           If Ntype = 2 Or Ntype = 3 Then
              For Idofn = 1 To NDOFN
                 Displ(Icase, Ipoin, Idofn) = Rload(IDDOF(Idofn, Ipoin))
              Next Idofn
           End If
        Next Ipoin

    End Sub
```

2.6 Calcolo dei parametri di sollecitazione

Una volta trovati gli spostamenti incogniti $[u_G]$ nel sistema globale, si devono dapprima trasformare gli spostamenti nel sistema locale ricavando $[u_L]$, per poi calcolare le forze generalizzate $[f_L]$ nel sistema locale. Si devono quindi eseguire le seguenti trasformazioni:

spostamenti da globale a locale	$[u_L] = [T]\,[u_G]$	(2.25)
Calcolo delle forze nel sistema locale	$[f_L]^* = [K_L]\,[u_L]$	(2.26)

Per ottenere i parametri di sollecitazione

$$[p] = [R_{1,i} \quad R_{2,i} \quad R_{3,i} \quad M_{1,i} \quad M_{2,i} \quad M_{3,i} \quad R_{1,j} \quad R_{2,j} \quad R_{3,j} \quad M_{1,j} \quad M_{2,j} \quad M_{3,j}]$$

è necessario un ulteriore passaggio, che consiste nella sottrazione dei carichi equivalenti:

$$[p] = [f_L]^* - [f_L]$$

dove:

- R_1, R_2, R_3 sono gli sforzi di taglio nelle direzioni degli assi locali 1, 2, 3;
- M_1 è il momento torcente;
- M_2, M_3 sono i momenti flettenti attorno agli assi locali 2, 3.

Viene riportata di seguito la routine per il calcolo dei parametri di sollecitazione.

```
    Sub STRBE(Icase As Integer, Nele1 As Integer, Nele2 As Integer, FileWork3 As String)

        '     STRESS CALCULATION FOR BEAM OR WINKLER ELEMENTS

        Dim Idofn, Idof1, Ldofn As Integer
        Dim GlobDisp(), LocDisp(), LocLoa(), Force() As Double
        Dim Delt1(NDIME) As Double


        Using sr As StreamReader = File.OpenText(FileWork3) 'stiffness LOCAL matrixes file

            ' *** LOOP OVER ELEMENTS

            For Ielem = Nele1 To Nele2

                'read stiffness matrix
                'ReDim Stiff(78)
                For Icolu = 1 To 78
                    Stiff(Icolu) = sr.ReadLine
                Next Icolu

                'calculate displacements LocDisp() and equivalent forces LocLoa()
                'in local coordinates
                'transformation T matrices are calculated only once after reading data
                ReDim GlobDisp(NEVAB), LocDisp(NEVAB), LocLoa(NEVAB)

                For Idofn = 1 To NDOFN
                    Idof1 = Idofn + NDOFN
                    GlobDisp(Idofn) = Displ(Icase, Inc1(Ielem), Idofn)
                    GlobDisp(Idof1) = Displ(Icase, Inc2(Ielem), Idofn)
                Next Idofn

                For Idofn = 1 To NDOFN
                    Idof1 = Idofn + NDOFN
                    For Jdofn = 1 To NDOFN
                        Ldofn = Jdofn + NDOFN
                        LocDisp(Idofn) += Tmat(Ielem, Idofn, Jdofn) * GlobDisp(Jdofn)
                        LocDisp(Idof1) += Tmat(Ielem, Idofn, Jdofn) * GlobDisp(Ldofn)

                        LocLoa(Idofn) += Tmat(Ielem, Idofn, Jdofn) * ELOAD(Jdofn, Ielem)
                        LocLoa(Idof1) += Tmat(Ielem, Idofn, Jdofn) * ELOAD(Ldofn, Ielem)
                    Next Jdofn
                Next Idofn

                'calculate forces in local system: [K] u = f
```

```
            ReDim Force(NEVAB)
            Dim Index As Integer

            For Ievab = 1 To NEVAB
                For Jevab = 1 To NEVAB
                    Index = Kpos(Ievab, Jevab)
                    Force(Ievab) += Stiff(Index) * LocDisp(Jevab)
                Next Jevab
            Next Ievab

            'subtract equivalent loads
            For Ievab = 1 To NEVAB
                Force(Ievab) -= LocLoa(Ievab)
            Next Ievab

            For Ievab = 1 To NEVAB
                If Math.Abs(Force(Ievab)) < 0.0000000001 Then Force(Ievab) = 0.0
            Next Ievab

            'store values in Stre1(,,), Stre2(,,) matrices
            For Idofn = 1 To NDOFN
                Stre1(Icase, Ielem, Idofn) = -Force(Idofn)
                Stre2(Icase, Ielem, Idofn) = Force(Idofn + NDOFN)
            Next Idofn

        Next Ielem
    End Using

End Sub
```

2.7 Calcolo delle reazioni vincolari

Le reazioni vengono calcolate con un procedimento analogo a quello descritto nel paragrafo precedente, utilizzando però le matrici di rigidezza riferite al sistema globale. Per ogni elemento i:

$$[f_G]_i^* = [K_G]_i\,[u_G]$$

dove di $[u_G]$ vengono utilizzati gli spostamenti generalizzati che competono ai nodi di estremità dell'elemento. Anche in questo caso devono essere sottratti i carichi equivalenti:

$$[f_G]_i = [f_G]_i^* - [f_L]$$

Anche di $[f_L]$ vengono utilizzate le forze generalizzate che competono ai nodi di estremità dell'elemento.

Le reazioni ad ogni nodo vengono infine ottenute sommando i contributi che derivano da tutti gli elementi che vi afferiscono.

Viene riportata di seguito la routine per il calcolo delle reazioni vincolari.

```
Sub ComputeReactions(ByVal Icase As Integer, FileWork1 As String)

    Dim Idofn, Idof1, Ipoi1, Ipoi2 As Integer
    Dim Vmax1, Vmax2, Vmax3, Vmax4, Vmax5, Vmax6 As Single

    Using sr As StreamReader = File.OpenText(FileWork1) 'stiffness matrixes file

        For Ielem = 1 To Nelem

            'read stiffness matrix
            For Icolu = 1 To 78
                Stiff(Icolu) = sr.ReadLine
            Next Icolu

            ' *** GLOBAL FORCES AT NODES    f = [K] u

            For Ievab = 1 To NEVAB
                For Jevab = 1 To NEVAB
                    Idofn = LMDOF(Jevab, Ielem)
```

```
                        If Idofn > 0 Then GlobF(Icase, Ielem, Ievab) += Stiff(Kpos(Ievab,
Jevab)) * Rload(Idofn)
                    Next Jevab
                Next Ievab

                ' *** SUBTRACT EQUIVALENT LOADS

                For Ievab = 1 To NEVAB
                    GlobF(Icase, Ielem, Ievab) -= ELOAD(Ievab, Ielem)
                Next Ievab

                Ipoi1 = IJINC(Ielem, 1)
                Ipoi2 = IJINC(Ielem, 2)

                For Idofn = 1 To NDOFN
                    Idof1 = Idofn + NDOFN
                    TREAC(Icase, Ipoi1, Idofn) += GlobF(Icase, Ielem, Idofn)
                    TREAC(Icase, Ipoi2, Idofn) += GlobF(Icase, Ielem, Idof1)
                Next Idofn

            Next Ielem
        End Using

    End Sub
```

3 Bibliografia

[1] Bathe, K. J.: "*Finite Element Procedures in Engineering Analysis"*. Prentice Hall, 1982.

[2] Bertolino, F.: "*Metodi agli Elementi Finiti – (AA 2019/20)"*. Università degli Studi di Cagliari – Facoltà di Ingegneria e Architettura, 2019.

[3] D.M. 17.01.2018: "*Aggiornamento delle <<Norme tecniche per le costruzioni>>". Supplemento ordinario n° 8 alla GAZZETTA UFFICIALE – Serie generale n° 42.*

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

[5] Hinton, E. – Owen, D. R. J.: "*Finite Element Programming*". Academic Press, 1977.

[6] Vitaliani, R. – Martini, L.: "Lezioni di calcolo automatico delle strutture – *1ª parte"*. CUSL NUOVA VITA, 1987.

[7] Zienkiewicz, O. C.: "*The Finite Element Method*". Third Edition. Mc Graw Hill, 1977.

4 Appendice A

ESCLUSIONE DI RESPONSABILITÀ PER DANNI - IN NESSUN CASO L'ING. PAOLO VARAGNOLO SARÀ RESPONSABILE PER I DANNI DI QUALSIASI TIPO, DIRETTO O INDIRETTO, PER MANCATI GUADAGNI, INTERRUZIONE DELL'ATTIVITÀ, PERDITA DI INFORMAZIONI O ALTRE PERDITE ECONOMICHE DERIVANTI DALL'USO DI QUESTO PRODOTTO SOFTWARE E DELLA SUA DOCUMENTAZIONE. IL SOFTWARE VIENE PUBBLICATO A SCOPO DIDATTICO E CHIUNQUE LO UTILIZZI A QUALUNQUE TITOLO SI ASSUME OGNI RESPONSABILITA' DERIVANTE DAL SUO USO.

Vengono riportate in appendice le subroutine che non sono state inserite nel testo.

L'esecuzione ha inizio con la routine **`MDfem_Execute`**, che inizializza le variabili, legge i dati del problema, calcola gli assi locali e chiama la routine MDFEM che gestisce l'esecuzione dell'analisi statica.

La routine che legge i dati non verrà presentata, in quanto si vuole concentrare l'attenzione sulla sostanza della programmazione FEM. Per consentire di utilizzare comunque il programma, nel §4.1 l'originale subroutine **`ReadMdFemFile`** è stata riscritta in modo da inizializzare e definire i dati relativi all'esempio Goodnumb presentato nel §2.2. L'inserimento dei dati si avvale normalmente di un processore grafico che registra tutti i valori in un file, rispettando delle specifiche: l'inserimento e la descrizione delle routine necessarie avrebbe richiesto una trattazione che esula dagli intenti del presente lavoro.

La routine **`SubSpace`** (chiamata da **`MDfem_Execute`**) che permette di calcolare i modi di vibrare di una struttura verrà presentata in un successivo articolo "*Programma per l'analisi FEM di Elementi Beam 3-D – Guida Practica: Parte 2 –Analisi Modale"*.

```
Sub MDfem_Execute(dtmStart As Date)

    Dim x1, x2, y1, y2, z1, z2 As Double

    FileClose()

    Call InitVariables()

    Call ReadMdFemFile(NomeFile)

    'calculate transformation T matrix for all the elements: Tmat(,,)
    For Ielem = 1 To Nelem
        x1 = Xcoor(Inc1(Ielem)) : y1 = Ycoor(Inc1(Ielem)) : z1 = Zcoor(Inc1(Ielem))
        x2 = Xcoor(Inc2(Ielem)) : y2 = Ycoor(Inc2(Ielem)) : z2 = Zcoor(Inc2(Ielem))

        Call LocalAxes_Vitaliani_PV(Ielem, x1, y1, z1, x2, y2, z2)
    Next Ielem

    If Eigenvalues Then
        'Call SubSpace() 'calculate vibration modes
    End If
    Call MDFEM(dtmStart) 'calculate FEM static analysis

End Sub
```

4.1 I dati di input

```
Sub ReadMdFemFile(B$)

    'this is not the real reading routine:
    'it only sets the data for the tutorial Goodnumb example

    Dim Icase, Ielem, Ipoin As Integer

    RTOL = 0.000001
    NITEM = 16
    IFSS = 1
    IFPR = 0

    Tit1$ = "Column Height Example: good numbering"
    Npoin = 8
    Nelem = 9
    Ncase = 2
    Ntype = 2

    IFPOS = 1
    Eigenvalues = False
    NROOT = 1
    IfLumped = True
    NITEM = 16
    IFSS = 1
    IFPR = 0
    Gaccel = 9.807

    ' ..................................................
    Nmats = 1
    NCOMB = 1
    Ngaps = 0
    Call MdFemArrayDimensions(B$)

    'nodal coordinates
    Xcoor = {0, 0, 150, 0, 150, 0, 150, 0, 150}
    Ycoor = {0, 0, 0, 0, 0, 0, 0, 0, 0}
    Zcoor = {0, 0, 0, 150, 150, 300, 300, 450, 450}

    'fixity codes
    Iffix(1) = 1 : Iffix(2) = 1
    Iffiy(1) = 1 : Iffiy(2) = 1
    Iffiz(1) = 1 : Iffiz(2) = 1
    Ifrxx(1) = 1 : Ifrxx(2) = 1
    Ifryy(1) = 1 : Ifryy(2) = 1
    Ifrzz(1) = 1 : Ifrzz(2) = 1

    For Ipoin = 1 To Npoin
        IDDOF(1, Ipoin) = Iffix(Ipoin)
        IDDOF(2, Ipoin) = Iffiy(Ipoin)
        IDDOF(3, Ipoin) = Iffiz(Ipoin)
        IDDOF(4, Ipoin) = Ifrxx(Ipoin)
        IDDOF(5, Ipoin) = Ifryy(Ipoin)
        IDDOF(6, Ipoin) = Ifrzz(Ipoin)
    Next Ipoin

    'properties
    PROPS(Nmats, 1) = 2100000.0 'Young modulus
    PROPS(Nmats, 2) = 30.7 'Area
    PROPS(Nmats, 3) = 3892.0 'Jx=Jy
    PROPS(Nmats, 4) = 0 'Winkler coefficient
    PROPS(Nmats, 5) = 0 'Width (only for Winkler elements)
    PROPS(Nmats, 6) = 0.00785 'weight density
    PROPS(Nmats, 7) = 0 'mass density
    PROPS(Nmats, 8) = 1 'torsional inertia
    PROPS(Nmats, 9) = 0 'Poisson ratio
    PROPS(Nmats, 10) = 0.5 * PROPS(Nmats, 1) / (1 + PROPS(Nmats, 9)) 'G modulus
```

```
        TrazFlag(Nmats) = 0 'not an only tension element

        'element definition
        NelOnlyTraz = 0
        Mater = {0, 1, 1, 1, 1, 1, 1, 1, 1, 1}
        Inc1 = {0, 1, 2, 3, 3, 4, 5, 5, 6, 7}
        Inc2 = {0, 3, 4, 4, 5, 6, 6, 7, 8, 8}


        'loads data
        Icase = 1
        Titl$(Icase) = "C=1 - linear y load"
        Gravity_Case(Icase) = 0
        NpointLoads(Icase) = 0
        NspanLoads(Icase) = 2

        For Ielem = 1 To 2
            LoadType(Icase, Ielem) = 1
            Dload(Icase, Ielem, 1) = 0 'x load at node i
            Dload(Icase, Ielem, 2) = 10 'y load at node i
            Dload(Icase, Ielem, 3) = 0 'z load at node i
            Dload(Icase, Ielem, 4) = Dload(Icase, Ielem, 1) 'x load at node j
            Dload(Icase, Ielem, 5) = Dload(Icase, Ielem, 2) 'y load at node j
            Dload(Icase, Ielem, 6) = Dload(Icase, Ielem, 3) 'z load at node j
        Next Ielem

        Icase = 2
        Ipoin = 8
        Titl$(Icase) = "C=2 - concentrated top load"
        Gravity_Case(Icase) = 0
        NpointLoads(Icase) = 1
        PointLoad(Icase, Ipoin, 1) = 10
        PointLoad(Icase, Ipoin, 2) = 10
        PointLoad(Icase, Ipoin, 3) = -10
        PointLoad(Icase, Ipoin, 4) = 0
        PointLoad(Icase, Ipoin, 5) = 0
        PointLoad(Icase, Ipoin, 6) = 0

        NspanLoads(Icase) = 0


        'combinations data
        NCOMB = 1
        Tit_Comb(NCOMB) = "Combination 1"
        Comb_Factor(NCOMB, 1) = 1.3
        Comb_Factor(NCOMB, 2) = 1.5
        Call CombinationLoads() 'created 2016.06.23

    End Sub
```

4.2 Le variabili a scopo globale

Di seguito sono presentate le dichiarazioni delle variabili con scopo globale. L'eventuale modifica di tipo da single a double o viceversa comporta diverse approssimazioni nei calcoli e lievi modifiche nei risultati.

```
    Public dtmStart, dtmEnd As Date
    Public Errore As Boolean

    Public Tit1$, Titl$()
    Public DataFile, PrtFile, LoaFile, PrtString As String
    Public myPath, Tit_Comb() As String

    Public NDIME, Npoin, Nelem, NDOFN, NDOFT As Integer
    Public NNODE, NSTRE, Nmats, Nprop, Ntype, NCOMB As Integer
    Public NVFIX, NEVAB, IFSPR, IFPOS, NpointLoads(), NspanLoads() As Integer
    Public Ncase As Integer = 1
```

```
    Public Iffix(), Iffiy(), Iffiz(), Ifrxx(), Ifryy(), Ifrzz() As Integer
    Public Inc1(), Inc2(), Mater(), LoadType(,), Gravity_Case() As Integer
    Public IDDOF(,), LMDOF(,), IJINC(,), MAXAD(), MCOLH() As Integer
    Public NKGLO As Long

    Public Xcoor(), Ycoor(), Zcoor() As Single
    Public CORDS(,), XYCOO(,), SPRIN(,), Displ(,,) As Single
    Public Rx3D(), Ry3D(), Rz3D(), Rxx3D(), Ryy3D(), Rzz3D() As Single 'reaction sums in
3D Fem Modules
    Public GravityAmplif() As Single

    Public PROPS(,), GlobF(,,), TREAC(,,) As Double
    Public PointLoad(,,), Comb_Factor(,), Tmat(,,), Rload() As Double
    Public Stre1(,,), Stre2(,,), Dload(,,), DloaL(,,) As Double
    Public GLOBK(), ELOAD(,), Stiff(78), TRLOA(), FLOAD() As Double

    'from v3.2 for only compression restraints
    Public Ngaps, OldNpoin, OldNelem As Integer
    Public GAPS(,) As Integer

    'from 3.2 for only tension 2-D TRUSS or BEAM
    Public NelOnlyTraz As Integer, OnlyTraz() As Integer, TrazFlag() As Integer

    'Eigenvalues variables
    Public Eigenvalues, IfLumped As Boolean
    Public NROOT, NITEM, IFSS, IFPR As Integer
    Public RTOL, Gaccel As Single

    Public Const PiGre As Double = 3.14159265358979
```

4.3 Inizializzazioni e dimensionamenti

```
    Sub InitVariables()

        Ntype = 2

        NNODE = 2
        NDOFN = 6

        If Ntype = 2 Or Ntype = 3 Then
            NDIME = 3
            Nprop = 10
            NSTRE = 6
            NEVAB = NDOFN * NNODE
        End If

    End Sub
```

```
    Sub MdFemArrayDimensions(B$)

        'main arrays dimensioning

        ReDim Iffix(Npoin + Ngaps), Iffiy(Npoin + Ngaps), Iffiz(Npoin + Ngaps), Ifrxx(Npoin
+ Ngaps), Ifryy(Npoin + Ngaps), Ifrzz(Npoin + Ngaps)
        ReDim Xcoor(Npoin + Ngaps), Ycoor(Npoin + Ngaps), Zcoor(Npoin + Ngaps)
        ReDim SPRIN(NDOFN, Npoin + Ngaps), GAPS(3, Npoin + Ngaps)
        ReDim Inc1(Nelem + Ngaps), Inc2(Nelem + Ngaps)

        ReDim Mater(Nelem + Ngaps), OnlyTraz(Nelem + Ngaps)
        ReDim IDDOF(NEVAB, Npoin + Ngaps)
        ReDim XYCOO(NEVAB, Nelem + Ngaps), IJINC(Nelem + Ngaps, 2)
        ReDim LMDOF(NEVAB, Nelem + Ngaps)
```

```
        ReDim Comb_Factor(NCOMB, Ncase), Tit_Comb(NCOMB), GravityAmplif(NCOMB)

        ReDim TrazFlag(Nmats), PROPS(Nmats + Ngaps + 1, 10)

        ReDim GlobF(Ncase + NCOMB + NROOT, Nelem + Ngaps, 2 * NDOFN), TREAC(Ncase + NCOMB
+ NROOT, Npoin + Ngaps, NDOFN)
        ReDim Displ(Ncase + NCOMB + NROOT, Npoin, NDOFN)
        ReDim Stre1(Ncase + NCOMB + NROOT, Nelem + Ngaps, NSTRE + 2), Stre2(Ncase + NCOMB
+ NROOT, Nelem + Ngaps, NSTRE + 2)

        ReDim Titl$(Ncase + NCOMB + NROOT), Dload(Ncase + NCOMB + NROOT, Nelem + Ngaps,
6)
        ReDim Gravity_Case(Ncase + NCOMB + NROOT), PointLoad(Ncase + NCOMB + NROOT, Npoin
+ Ngaps, NDOFN)
        ReDim LoadType(Ncase + NCOMB + NROOT, Nelem + Ngaps), NpointLoads(Ncase + NCOMB +
NROOT), NspanLoads(Ncase + NCOMB + NROOT)
        ReDim Rx3D(Ncase + NCOMB), Ry3D(Ncase + NCOMB), Rz3D(Ncase + NCOMB), Rxx3D(Ncase
+ NCOMB), Ryy3D(Ncase + NCOMB), Rzz3D(Ncase + NCOMB)

        ReDim Tmat(Nelem, 6, 6) 'Tmat is the transformation matrix from local to global
coordinates

    End Sub
```

4.4 Altre Subroutines

La prima delle subroutine di questo paragrafo è il cuore del programma MdFem, che gestisce l'analisi statica con la chiamata alle subroutine principali. Seguono le altre subroutine, senza un ordine particolare. Le subroutine presentate all'interno del testo fanno parte del programma e non verranno ripresentate nel seguito.

```
    Sub MDFEM(dtmStart As Date)
        ' ====================================================================
        '
        '                    ========  MD fem   =======
        '
        '     FINITE ELEMENT (2-D,3-D) PROGRAM FOR MONO-DIMENSIONAL ELEMENTS
        '
        '            WRITTEN BY:  P. VARAGNOLO                    == pvi88,94 ==
        '
        '            LAST REVISED: 15.06.1993
        '                          03.03.1994   v 2.0
        '                          30.04.1994   v 2.1
        '                          17.06.1994   v 2.1
        '                          27.06.1994   v 2.1
        '                          18.11.1995   v 3.0  windows version
        '                          28.05.1998   v 3.2  GAP elements
        '                                ????          only tension beams
        '                            feb   2009   v 5.0  modified combinations, including
GAP elements and only tension elements
        '                          feb   2009   v 5.1  rewritten MDFEM pre-processor
        '                         feb   2009   v 6.0  improved only tension BEAM elements
managment
        '                                              + some bug fixing on load conditions
        '                          giu   2016   v 9.0  complete rewriting in vb.net, with
Cable 2015 (PVI) graphic environment
        '                                               From this release most of the out
core files have been discarded
        '                          mar 2021            2D to 3D transition
        ' ====================================================================

        Dim Time1$ = ""
        Dim ExcelFile, ElapsTime As String
        Dim Trazione, Compressione As Boolean
        Dim Iteration, DeltaTime As Integer
        Dim FileWork1 As String = myPath + "WORK1" 'open a file where will be saved the
element GLOBAL stiffness matrices
```

```
        Dim FileWork3 As String = myPath + "WORK3" 'open a file where will be saved the
element LOCAL  stiffness matrices
        Dim File1 As System.IO.StreamWriter = Nothing
        Dim File2 As System.IO.StreamWriter = Nothing
        Dim File3 As System.IO.StreamWriter = Nothing

        Errore = False

        ExcelFile = DataFile.Substring(0, Len(DataFile) - 4) + ".txt" : If Dir(ExcelFile,
0) <> "" Then Kill(ExcelFile)
        If Dir(PrtFile, 0) <> "" Then Kill(PrtFile)
        If Dir(LoaFile, 0) <> "" Then Kill(LoaFile)
        If Dir(myPath + "work1", 0) <> "" Then Kill(myPath + "work1")        'kill work1
if it exists from da previous calculations
        If Dir(myPath + "work3", 0) <> "" Then Kill(myPath + "work3")        'kill work3
if it exists from da previous calculations

        File1 = My.Computer.FileSystem.OpenTextFileWriter(PrtFile, True)
        File2 = My.Computer.FileSystem.OpenTextFileWriter(ExcelFile, True)
        File3 = My.Computer.FileSystem.OpenTextFileWriter(LoaFile, True)

        Call WriteGeneralData(File1)

        Call SetNodalData(File1)
        If Errore Then
            FileClose() : Exit Sub
        End If

        Call WriteProperties(File1, "MDFEM")

        ' *** NUMERATION OF DEGREES OF FREEDOM
        Call DOFNUM()

        ' *** CALL ELEMENT SUBROUTINE (+ generate gaps elements)
        Call INPELE(File1)

        ' *** CALCULATE ADDRESS OF DIAGONAL ELEMENTS
        Call ADDRES()

        ' *** DEAL WITH LOAD CASES AND LOAD COMBINATIONS

        ReDim DloaL(Ncase + NCOMB + NROOT, Nelem + Ngaps, 2 * NDIME) 'distributed loads
in local coordinates
        Iteration = 0

        For Icase = 1 To Ncase + NCOMB

            ' *** CALCULATE STIFFNESS MATRIX
            Call CreateStiffnessMatrix(FileWork1, FileWork3)

            Iteration = 0

            Iteration += 1

            ' *** LOOP OVER LOAD CONDITIONS
            Call LOADS(Icase, Iteration, File1)

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

            Call BACKSU(Icase)

            Call ComputeReactions(Icase, FileWork1)

            Call ComputeStresses(Icase, FileWork3)

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

```
        Next Icase

        dtmEnd = TimeValue(Now)
        DeltaTime = DateDiff("s", dtmStart, dtmEnd)
        Call ElapsedTime(DeltaTime, Time1$)
        ElapsTime = "    elaboration time (hh:mm:ss) =   " + Time1$ + vbCrLf
        File1.WriteLine()
        File1.WriteLine(ElapsTime)

        FileClose()
        File1.Close()
        If Ntype <> 4 Then File2.Close() 'Excel file

        Call KillWorkFiles()

    End Sub
```

```
    Sub CombinationLoads()
        'create new load cases corresponding to load combinations
        'this approach, different with respect to the previous releases,
        'allows combinations with non linear elements (gaps or tension elements)

        Dim Testo As String
        Dim Dire1 As Integer
        Dim q1, q2, q3, q4, q5, q6 As Double

        'control if gravity directions are admittable
        Dire1 = 0
        For Icase = 1 To Ncase
            Dire1 = Gravity_Case(Icase)
            If Dire1 <> 0 Then
                For Kcase = 1 To Ncase
                    If Gravity_Case(Kcase) <> 0 Then
                        If Gravity_Case(Kcase) <> Dire1 And Gravity_Case(Kcase) <> -Dire1
Then
                            Testo = "Gravity must be defined in only one direction to
allow load combinations." + vbCrLf
                            Testo += "Calculation stopped."
                            MsgBox(Testo, vbExclamation, "Warning")
                            Errore = True
                            Exit Sub
                        End If
                    End If
                Next Kcase
            End If
        Next Icase

        'creation of the new load cases
        For Icomb = 1 To NCOMB
            'Initialize arrays for combination loads
            For Ipoin = 1 To Npoin
                PointLoad(Ncase + Icomb, Ipoin, 1) = 0
                PointLoad(Ncase + Icomb, Ipoin, 2) = 0
                PointLoad(Ncase + Icomb, Ipoin, 3) = 0
            Next Ipoin

            For Ielem = 1 To Nelem
                If Ntype = 2 Or Ntype = 3 Then
                    For Icolu = 1 To 6
                        Dload(Ncase + Icomb, Ielem, Icolu) = 0
                    Next Icolu
                End If
            Next Ielem

            'build combination load arrays
            For Icase = 1 To Ncase
                'Gravity
                If  Gravity_Case(Icase)  <>  0  Then  Gravity_Case(Ncase  +  Icomb)  =
Gravity_Case(Icase)
```

```
                If    Gravity_Case(Icase)    <>    0    Then    GravityAmplif(Icomb)    +=
Comb_Factor(Icomb, Icase)

                For Ipoin = 1 To Npoin
                    PointLoad(Ncase + Icomb, Ipoin, 1) += PointLoad(Icase, Ipoin, 1) *
Comb_Factor(Icomb, Icase)
                    PointLoad(Ncase + Icomb, Ipoin, 2) += PointLoad(Icase, Ipoin, 2) *
Comb_Factor(Icomb, Icase)
                    PointLoad(Ncase + Icomb, Ipoin, 3) += PointLoad(Icase, Ipoin, 3) *
Comb_Factor(Icomb, Icase)
                Next Ipoin

                'span loads
                For Jelem = 1 To Nelem
                    If Ntype = 2 Or Ntype = 3 Then
                        For Icolu = 1 To 6
                            q1 = Dload(Icase, Jelem, Icolu)
                            Dload(Ncase + Icomb, Jelem, Icolu) += q1 * Comb_Factor(Icomb,
Icase)
                        Next Icolu
                    End If
                Next Jelem
            Next Icase

            'compute number of Point and Span loads, though they are used only to know if
they exist or not
            NpointLoads(Ncase + Icomb) = 0
            For Ipoin = 1 To Npoin
                If PointLoad(Ncase + Icomb, Ipoin, 1) <> 0 Or PointLoad(Ncase + Icomb,
Ipoin, 2) <> 0 Or PointLoad(Ncase + Icomb, Ipoin, 3) <> 0 Then
                    NpointLoads(Ncase + Icomb) += 1
                End If
            Next Ipoin

            NspanLoads(Ncase + Icomb) = 0
            For Ielem = 1 To Nelem
                If Ntype = 2 Or Ntype = 3 Then
                    q1 = Dload(Ncase + Icomb, Ielem, 1) : q2 = Dload(Ncase + Icomb, Ielem,
2)
                    q3 = Dload(Ncase + Icomb, Ielem, 3) : q4 = Dload(Ncase + Icomb, Ielem,
4)
                    q5 = Dload(Ncase + Icomb, Ielem, 5) : q6 = Dload(Ncase + Icomb, Ielem,
6)
                    If q1 <> 0 Or q2 <> 0 Or q3 <> 0 Or q4 <> 0 Or q5 <> 0 Or q6 <> 0 Then
                        NspanLoads(Ncase + Icomb) += 1
                        LoadType(Ncase + Icomb, Ielem) = 2
                    End If
                End If
            Next Ielem
        Next Icomb

    End Sub
```

```
    Public Sub LocalAxes_Vitaliani_PV(Ielem As Integer, x1 As Double, y1 As Double, z1 As
Double, x2 As Double, y2 As Double, z2 As Double)
        'direction cosines of local axes according to Vitaliani, Martini § 3.3.7

        Dim x3, y3, z3 As Double 'auxiliary point "o"
        Dim x4, y4, z4 As Double 'temporary array
        Dim l_xy, Lungh, CosX, CosY, Emme As Double
        Dim a1, a2, a3, b1, b2, b3 As Double

        Dim xa, ya, za, xb, yb, zb, xc, yc, zc, x1o As Double

        'translate coordinate reference system on first node
        x2 = x2 - x1 : x1 = 0
        y2 = y2 - y1 : y1 = 0
        z2 = z2 - z1 : z1 = 0
```

```
        'element slope in the vertical plane
        l_xy = Math.Sqrt((x2 - x1) ^ 2 + (y2 - y1) ^ 2)
        If l_xy = 0 Then
            'vertical element, arrange point "o" on y,z plane
            x3 = x1 '+ 10 'x,z plane
            y3 = y1 + 10
            z3 = z1
        Else
            'not vertical element, arrange point "o" on the plane containing the element
and z-axis
            CosX = (x2 - x1) / l_xy : CosY = (y2 - y1) / l_xy
            Emme = (z2 - z1) / l_xy

            z3 = 10
            x1o = -10 * Emme
            x3 = x1o * CosX
            y3 = x1o * CosY

        End If

        'definition of versor 1 ------------------------------------------------------------
------------
        '-------------------------------------------------------------------------------
------------
        'versor 1 is parallel to the element
        Lungh = Math.Sqrt(x2 ^ 2 + y2 ^ 2 + z2 ^ 2)
        xa = x2 / Lungh
        ya = y2 / Lungh
        za = z2 / Lungh

        'definition of versor 2 ------------------------------------------------------------
------------
        '-------------------------------------------------------------------------------
------------
        'vector product of versor 1 (xa,ya,za) x vector "o" (x3,y3,z3)
        a1 = xa : a2 = ya : a3 = za
        b1 = x3 - x1 : b2 = y3 - y1 : b3 = z3 - z1
        x4 = a2 * b3 - a3 * b2
        y4 = a3 * b1 - a1 * b3
        z4 = a1 * b2 - a2 * b1

        'vector product of vector (x4,y4,z4) x vector a (xa,ya,za)
        a1 = x4 : a2 = y4 : a3 = z4
        b1 = xa : b2 = ya : b3 = za
        xb = a2 * b3 - a3 * b2
        yb = a3 * b1 - a1 * b3
        zb = a1 * b2 - a2 * b1

        Lungh = Math.Sqrt(xb ^ 2 + yb ^ 2 + zb ^ 2)
        xb = xb / Lungh
        yb = yb / Lungh
        zb = zb / Lungh

        'vector product of versor 1 (xa,ya,za) x versor 2 (xb,yb,zb)
        a1 = xa : a2 = ya : a3 = za
        b1 = xb : b2 = yb : b3 = zb
        xc = a2 * b3 - a3 * b2
        yc = a3 * b1 - a1 * b3
        zc = a1 * b2 - a2 * b1

        'direction cosines array
        Lungh = Math.Sqrt(xa ^ 2 + ya ^ 2 + za ^ 2)
        Tmat(Ielem, 1, 1) = xa / Lungh  'cos(1-x)
        Tmat(Ielem, 1, 2) = ya / Lungh  'cos(1-y)
        Tmat(Ielem, 1, 3) = za / Lungh  'cos(1-z)

        Lungh = Math.Sqrt(xb ^ 2 + yb ^ 2 + zb ^ 2)
        Tmat(Ielem, 2, 1) = xb / Lungh  'cos(2-x)
        Tmat(Ielem, 2, 2) = yb / Lungh  'cos(2-y)
        Tmat(Ielem, 2, 3) = zb / Lungh  'cos(2-z)
```

```
        Lungh = Math.Sqrt(xc ^ 2 + yc ^ 2 + zc ^ 2)
        Tmat(Ielem, 3, 1) = xc / Lungh   'cos(3-x)
        Tmat(Ielem, 3, 2) = yc / Lungh   'cos(3-y)
        Tmat(Ielem, 3, 3) = zc / Lungh   'cos(3-z)

        Tmat(Ielem, 4, 4) = Tmat(Ielem, 1, 1) : Tmat(Ielem, 4, 5) = Tmat(Ielem, 1, 2) :
Tmat(Ielem, 4, 6) = Tmat(Ielem, 1, 3)
        Tmat(Ielem, 5, 4) = Tmat(Ielem, 2, 1) : Tmat(Ielem, 5, 5) = Tmat(Ielem, 2, 2) :
Tmat(Ielem, 5, 6) = Tmat(Ielem, 2, 3)
        Tmat(Ielem, 6, 4) = Tmat(Ielem, 3, 1) : Tmat(Ielem, 6, 5) = Tmat(Ielem, 3, 2) :
Tmat(Ielem, 6, 6) = Tmat(Ielem, 3, 3)

    End Sub
```

```
    Sub ADDRES()

        Dim Ndof1 As Long
        ReDim MAXAD(NDOFT + 1)

        MAXAD(1) = 1
        MAXAD(2) = 2

        If NDOFT > 1 Then
            For Idofn = 2 To NDOFT
                MAXAD(Idofn + 1) = MAXAD(Idofn) + MCOLH(Idofn) + 1
            Next Idofn
        End If

        Ndof1 = NDOFT + 1
        NKGLO = MAXAD(Ndof1) - MAXAD(1)

    End Sub
```

```
    Sub ASSEMB(FileWork1 As String, FileWork3 As String)

        'CALL ELEMENT STIFFNESS AND ASSEMBLE STRUCTURE STIFFNESS MATRIX

        Dim Ielem As Integer
        ReDim GLOBK(NKGLO)

        ' *** LOOP OVER ELEMENT GROUPS

        For Ielem = 1 To Nelem
            Select Case Ntype
                Case 2
                    'Call Beam3DstiffBathe(Ielem, FileWork1, FileWork3)
                    Call Beam3DstiffVaragnolo(Ielem, FileWork1, FileWork3)
                    'Call Beam3DstiffVitaliani(Ielem, FileWork1,FileWork3)
                Case 3
                    'Call WISTIF(Ielem, FileWork1, FileWork3)
            End Select
        Next Ielem

    End Sub
```

```
    Function MIN0(Arg1 As Integer, Arg2 As Integer) As Integer

        If Arg1 < Arg2 Then
            MIN0 = Arg1
        Else
            MIN0 = Arg2
        End If

    End Function
```

```
    Sub CreateStiffnessMatrix(FileWork1 As String, FileWork3 As String)

        Using sw As New StreamWriter(FileWork1) 'file where GLOBAL stiffness matrix is
stored
        End Using
        Using sw As New StreamWriter(FileWork3) 'file where LOCAL  stiffness matrix is
stored
        End Using

        ' *** CALL ELEMENT SUBROUTINE

        Call ASSEMB(FileWork1, FileWork3)

        '' *** ASSEMBLE SPRING RIGIDITIES

        'If IFSPR = 1 Then Call ADDSPR()

    End Sub
```

```
    Sub Beam3DstiffVaragnolo(Ielem As Integer, FileWork1 As String, FileWork3 As String)

        'the part coming from Bathe is commented: this routine products the same matrices
as Bathe

        ' *** STIFFNESS GLOBAL MATRIX FOR 3D-BEAM ELEMENTS

        Dim Imats, Jind, Kind, ii As Integer
        Dim Aleng, Alen2, Delt1(NDIME), Kmat(12, 12), Trasf(12, 12) As Double
        Dim ax, ay, az, aax, aay, aaz, Shfy, Shfz, zy, eiy, eiz, Commy, Commz, Sum As
Double
        Dim Sa(12, 24), Asa(24, 24) As Double

        ' *** ASSEMBLE STRUCTURE STIFFNESS MATRIX

        Imats = Mater(Ielem)

        Alen2 = 0.0
        For Idime = 1 To NDIME
            Delt1(Idime) = XYCOO(Idime + 3, Ielem) - XYCOO(Idime, Ielem)
            Alen2 = Alen2 + Delt1(Idime) * Delt1(Idime)
        Next Idime

        Aleng = Math.Sqrt(Alen2)

        ax = PROPS(Imats, 2) 'axial area                              COPROP(MELTYP, 1)
        ay = 0 'PROPS(Imats, 2) 'shear area in local 2 direction      COPROP(MELTYP, 2)
        az = 0 'PROPS(Imats, 2) 'shear area in local 2 direction      COPROP(MELTYP, 3)
        aax = PROPS(Imats, 8) 'torsional inertia                      COPROP(MELTYP, 4)
        aay = PROPS(Imats, 3) 'flexural inertia about local 2-axis COPROP(MELTYP, 5)
        aaz = PROPS(Imats, 3) 'flexural inertia about local 3-axis COPROP(MELTYP, 6)
        Shfy = 0.0
        Shfz = 0.0
        zy = PROPS(Imats, 1) / (Aleng * Aleng) 'E(MATTYP) / (DL * DL)
        eiy = zy * aay
        eiz = zy * aaz
        If ay <> 0 Then Shfy = 6 * eiz / (PROPS(Imats, 10) * ay)
        If az <> 0 Then Shfz = 6 * eiy / (PROPS(Imats, 10) * az)
        Commy = eiy / (1 + 2 * Shfz)
        Commz = eiz / (1 + 2 * Shfy)

        ' Form ELEMENT STIFFNESS IN LOCAL COORDINATES
        Kmat(1, 1) = PROPS(Imats, 1) * ax / Aleng
        Kmat(4, 4) = PROPS(Imats, 10) * aax / Aleng
        Kmat(2, 2) = Commz * 12 / Aleng
        Kmat(3, 3) = Commy * 12 / Aleng
        Kmat(5, 5) = Commy * 4 * Aleng * (1 + 0.5 * Shfz)
        Kmat(6, 6) = Commz * 4 * Aleng * (1 + 0.5 * Shfy)
        Kmat(2, 6) = Commz * 6
```

```
        Kmat(3, 5) = -Commy * 6

        For Iind = 1 To 6
            Jind = Iind + 6
            Kmat(Jind, Jind) = Kmat(Iind, Iind)
        Next Iind

        For Iind = 1 To 4
            Jind = Iind + 6
            Kmat(Iind, Jind) = -Kmat(Iind, Iind)
        Next Iind

        Kmat(6, 12) = Kmat(6, 6) * (1 - Shfy) / (2 + Shfy)
        Kmat(5, 11) = Kmat(5, 5) * (1 - Shfz) / (2 + Shfz)
        Kmat(2, 12) = Kmat(2, 6)
        Kmat(6, 8) = -Kmat(2, 6)
        Kmat(8, 12) = -Kmat(2, 6)
        Kmat(3, 11) = Kmat(3, 5)
        Kmat(5, 9) = -Kmat(3, 5)
        Kmat(9, 11) = -Kmat(3, 5)

        For Iind = 2 To 12
            Kind = Iind - 1
            For Jind = 1 To Kind
                Kmat(Iind, Jind) = Kmat(Jind, Iind)
            Next Jind
        Next Iind

        'Store local stiffness matrix in a vector according to Bathe technique in STAP
program
        ii = 0
        For Irow = 1 To 12
            For Icol = Irow To 12
                ii += 1
                Stiff(ii) = Kmat(Irow, Icol)
            Next Icol
        Next Irow

        Using sw As StreamWriter = File.AppendText(FileWork3)
            For Icol = 1 To 78
                sw.WriteLine(Stiff(Icol))
            Next Icol
        End Using

        Call FillTrasf(Ielem, Trasf) 'fill Trasf(12, 12) matrix with 4 Transformation
Tmat(6, 6)

        'first product: Sa(12, 12) = [T]transp * [Kmat]
        'change rows and columns of Tmat to obtain the transpose matrix
        For Ievab = 1 To NEVAB
            For Jevab = 1 To NEVAB
                Sum = 0
                For Kevab = 1 To NEVAB
                    Sum += Trasf(Kevab, Ievab) * Kmat(Kevab, Jevab)
                Next Kevab
                Sa(Ievab, Jevab) = Sum
            Next Jevab
        Next Ievab

        'second product: Asa(12, 12) = [Sa] * [T]
        For Ievab = 1 To NEVAB
            For Jevab = 1 To NEVAB
                Sum = 0
                For Kevab = 1 To NEVAB
                    Sum += Sa(Ievab, Kevab) * Trasf(Kevab, Jevab)
                Next Kevab
                Asa(Ievab, Jevab) = Sum
            Next Jevab
        Next Ievab
```

```
        'Store global stiffness matrix in a vector according to Bathe technique in STAP
program
        ii = 0
        For Irow = 1 To NEVAB
            For Icol = Irow To NEVAB
                ii += 1
                Stiff(ii) = Asa(Irow, Icol)
            Next Icol
        Next Irow

        Using sw As StreamWriter = File.AppendText(FileWork1)
            For Icol = 1 To 78
                sw.WriteLine(Stiff(Icol))
            Next Icol
        End Using

        Call ADDBAN(Ielem)

    End Sub
```

```
    Sub COLHT(Ielem)

        Dim Ldofn As Integer = 100000
        Dim Idofn, Icolh As Integer

        For Ievab = 1 To NEVAB
            Idofn = LMDOF(Ievab, Ielem)

            If Idofn > 0 Then
                Icolh = Idofn - Ldofn
            End If

            If Icolh <= 0 And Idofn > 0 Then
                Ldofn = Idofn
            End If

        Next Ievab

        For Ievab = 1 To NEVAB
            Idofn = LMDOF(Ievab, Ielem)
            If Idofn <> 0 Then
                Icolh = Idofn - Ldofn
                If Icolh > MCOLH(Idofn) Then MCOLH(Idofn) = Icolh
            End If
        Next Ievab

    End Sub
```

```
    Sub DOFNUM()

        ' *** NUMERATION OF DEGREES OF FREEDOM
        ' d.o.f. will be renumbered in MOME_RELEASE if some release is requested

        NDOFT = 0

        For Ipoin = 1 To Npoin
            For Idofn = 1 To NDOFN
                If IDDOF(Idofn, Ipoin) = 0 Then
                    NDOFT = NDOFT + 1
                    IDDOF(Idofn, Ipoin) = NDOFT
                Else
                    IDDOF(Idofn, Ipoin) = 0
                End If
            Next Idofn
        Next Ipoin
```

```
    End Sub
```

```
    Sub INPELE(File1 As StreamWriter)

        Dim TYPEL As String = ""

        ReDim MCOLH(NDOFT)
        Dim OldInc(Nelem + Ngaps) As Integer
        Dim Idim1, Idof1 As Integer


        PrtString = vbCrLf : PrtString += vbCrLf
        PrtString += "     SET  TYPE  ELEMENT    NODE I   NODE J" + vbCrLf
        PrtString += "----------------------------------------"
        File1.WriteLine(PrtString)

        For Ielem = 1 To Nelem
            'SET UP INCIDENCES
            For Idime = 1 To NDIME
                Idim1 = Idime + 3
                XYCOO(Idime, Ielem) = CORDS(Inc1(Ielem), Idime)
                XYCOO(Idim1, Ielem) = CORDS(Inc2(Ielem), Idime)
            Next Idime

            IJINC(Ielem, 1) = Inc1(Ielem)
            IJINC(Ielem, 2) = Inc2(Ielem)

            For Idofn = 1 To NDOFN
                Idof1 = Idofn + NDOFN
                'set degrees of freedom at the element ends only if no releases have been
declared
                If LMDOF(Idofn, Ielem) = 0 Then
                    LMDOF(Idofn, Ielem) = IDDOF(Idofn, Inc1(Ielem))
                End If
                If LMDOF(Idof1, Ielem) = 0 Then
                    LMDOF(Idof1, Ielem) = IDDOF(Idofn, Inc2(Ielem))
                End If
            Next Idofn

            'CALCULATE COLUMN HEIGHTS
            Call COLHT(Ielem)

            If Ntype = 2 Then TYPEL = "BEAM"
            If Ntype = 3 Then TYPEL = "WINK"
            PrtString = "    " + String.Format("{0,5}", Mater(Ielem).ToString("####0"))
            PrtString += "  " + TYPEL
            PrtString += "  " + String.Format("{0,5}", Ielem.ToString("####0"))
            PrtString += "      " + String.Format("{0,5}", Inc1(Ielem).ToString("####0"))
            PrtString += "    " + String.Format("{0,5}", Inc2(Ielem).ToString("####0"))
            File1.WriteLine(PrtString)
        Next Ielem

        'add gap elements gap v3.2
        'If Ngaps > 0 Then Call GenerateGaps(File1)

    End Sub
```

```
    Sub WriteGeneralData(File1 As StreamWriter)

        Dim Typel As String = ""

        PrtString = "      " + "================================================" +
vbCrLf
        PrtString += "     " + "                    M D F E M                   " +
vbCrLf
        PrtString += "      " + "================================================" +
vbCrLf
```

```
        PrtString += vbCrLf : PrtString += vbCrLf
        PrtString += "F I N I T E   E L E M E N T " + Str(NDIME) + "-D  A N A L Y S I S"
+ vbCrLf
        PrtString += vbCrLf : PrtString += vbCrLf

        PrtString += vbCrLf : PrtString += vbCrLf
        PrtString += Tit1$ + vbCrLf
        PrtString += vbCrLf : PrtString += vbCrLf

        PrtString += vbCrLf : PrtString += vbCrLf
        PrtString += "***  G E N E R A L   D A T A  ***" + vbCrLf
        PrtString += vbCrLf

        PrtString  +=  "N.  OF  POINTS  .................  ="  +  String.Format("{0,5}",
Npoin.ToString("####0")) + vbCrLf
        PrtString  +=  "N.  OF  ELEMENTS  ..............  ="  +  String.Format("{0,5}",
Nelem.ToString("####0")) + vbCrLf
        PrtString  +=  "N.  OF  LOAD  CONDITIONS  .....  ="  +  String.Format("{0,5}",
Ncase.ToString("####0")) + vbCrLf
        PrtString  +=  "N.  OF  PROPERTY  SETS  ........  ="  +  String.Format("{0,5}",
Nmats.ToString("####0")) + vbCrLf
        PrtString += vbCrLf : PrtString += vbCrLf : PrtString += vbCrLf

        PrtString += vbCrLf : PrtString += vbCrLf : PrtString += vbCrLf
        PrtString += "***  E L E M E N T   G R O U P   D A T A  ***" + vbCrLf
        PrtString += vbCrLf
        PrtString += "ELEMENT TYPE     N. OF ELEMENTS" + vbCrLf

        If Ntype = 1 Then Typel = "TRUSS"
        If Ntype = 2 Then Typel = "BEAM"
        If Ntype = 3 Then Typel = "WINK"
        If Ntype = 4 Then Typel = "GRID"

        PrtString += "          " + Typel + "                " + String.Format("{0,5}",
Nelem.ToString("####0"))
        File1.WriteLine(PrtString)

    End Sub
```

```
    Sub SetNodalData(File1 As StreamWriter)

        'Ngaps is set in the pre-processor
        ReDim CORDS(Npoin + Ngaps, 3)

        Dim Testo As String
        Dim Ldofn, GapsC As Integer
        Dim Kstep As Integer = 1

        ' *** READ NODES DATA

        PrtString = vbCrLf : PrtString += vbCrLf : PrtString += vbCrLf
        PrtString += "***  N O D A L   D A T A  ***" + vbCrLf

        'initialize array CORDS(,) used in MdFem
        For Ipoin = 1 To Npoin
            CORDS(Ipoin, 1) = Xcoor(Ipoin)
            CORDS(Ipoin, 2) = Ycoor(Ipoin)
            CORDS(Ipoin, 3) = Zcoor(Ipoin)
            If NDIME = 2 Then CORDS(Ipoin, 3) = 0#
        Next Ipoin

        ' *** WRITE NODAL DATA
        PrtString += vbCrLf : PrtString += " NODE         X-COORD.       Y-COORD.       Z-
COORD."
        File1.WriteLine(PrtString)

        For Ipoin = 1 To Npoin
            If NDIME = 3 Then
```

```
                PrtString = String.Format("{0,5}", Ipoin.ToString("####0")) + "    "
                PrtString         +=         String.Format("{0,15}",         CORDS(Ipoin,
1).ToString("0.0000E+00"))
                PrtString         +=         String.Format("{0,15}",         CORDS(Ipoin,
2).ToString("0.0000E+00"))
                PrtString         +=         String.Format("{0,15}",         CORDS(Ipoin,
3).ToString("0.0000E+00"))
                File1.WriteLine(PrtString)
            Else
                PrtString = String.Format("{0,5}", Ipoin.ToString("####0")) + "    "
                PrtString         +=         String.Format("{0,15}",         CORDS(Ipoin,
1).ToString("0.0000E+00"))
                PrtString         +=         String.Format("{0,15}",         CORDS(Ipoin,
2).ToString("0.0000E+00"))
                File1.WriteLine(PrtString)
            End If
        Next Ipoin

        ' *** WRITE RESTRAINTS DATA

        PrtString = vbCrLf : PrtString += vbCrLf
        PrtString += "***  RESTRAINTS  INFORMATIONS  ***" + vbCrLf
        PrtString += vbCrLf
        PrtString += "  N.1  N.2    Step       -- degrees  of  freedom --" + vbCrLf
        PrtString += "                          x    y    z   xx   yy   zz"
        File1.WriteLine(PrtString)

        For Ipoin = 1 To Npoin
            If IDDOF(1, Ipoin) + IDDOF(2, Ipoin) + IDDOF(3, Ipoin) + IDDOF(4, Ipoin) +
IDDOF(5, Ipoin) + IDDOF(6, Ipoin) > 0 Then
                PrtString = String.Format("{0,5}", Ipoin.ToString("####0"))
                PrtString += String.Format("{0,5}", Ipoin.ToString("####0"))
                PrtString += "      1    "
                PrtString += String.Format("{0,5}", IDDOF(1, Ipoin).ToString("0"))
                PrtString += String.Format("{0,5}", IDDOF(2, Ipoin).ToString("0"))
                PrtString += String.Format("{0,5}", IDDOF(3, Ipoin).ToString("0"))
                PrtString += String.Format("{0,5}", IDDOF(4, Ipoin).ToString("0"))
                PrtString += String.Format("{0,5}", IDDOF(5, Ipoin).ToString("0"))
                PrtString += String.Format("{0,5}", IDDOF(6, Ipoin).ToString("0"))
                File1.WriteLine(PrtString)
            End If
        Next Ipoin

        ' *** READ SPRINGS VALUES

        IFSPR = 0
        For Ipoin = 1 To Npoin
            If SPRIN(1, Ipoin) + SPRIN(2, Ipoin) + SPRIN(3, Ipoin) > 0 Then
                IFSPR = 1
                Exit For
            End If
        Next Ipoin

        Errore = False

        If IFSPR = 1 Then
            Dim Spri1, Spri2, Spri3, Spri4, Spri5, Spri6 As Single

            PrtString = vbCrLf : PrtString += vbCrLf
            PrtString += "***  S P R I N G S  ***" + vbCrLf
            PrtString += vbCrLf : PrtString += vbCrLf
            If Ntype = 2 Or Ntype = 3 Then PrtString += "  N.1  N.2    STEP          Kx
Ky            Kz            Kxx            Kyy            Kzz "
            File1.WriteLine(PrtString)

            For Ipoin = 1 To Npoin
                If Ntype = 2 Or Ntype = 3 Then
                    Spri1 = SPRIN(1, Ipoin) : Spri2 = SPRIN(2, Ipoin) : Spri3 = SPRIN(3,
Ipoin)
                    Spri4 = SPRIN(1, Ipoin) : Spri5 = SPRIN(2, Ipoin) : Spri6 = SPRIN(3,
Ipoin)
```

```
                End If

                If Spri1 + Spri2 + Spri3 + Spri4 + Spri5 + Spri6 > 0 Then
                    PrtString   =   String.Format("{0,5}",   Ipoin.ToString("####0"))   +
String.Format("{0,5}", Ipoin.ToString("####0"))
                    PrtString   +=   String.Format("{0,8}",   Kstep.ToString("####0"))   +
String.Format("{0,15}", Spri1.ToString("0.0000E+00"))
                    PrtString += String.Format("{0,15}", Spri2.ToString("0.0000E+00")) +
String.Format("{0,15}", Spri3.ToString("0.0000E+00"))
                    PrtString += String.Format("{0,15}", Spri4.ToString("0.0000E+00"))
                    PrtString += String.Format("{0,15}", Spri5.ToString("0.0000E+00")) +
String.Format("{0,15}", Spri6.ToString("0.0000E+00"))
                    File1.WriteLine(PrtString)
                End If

                'CHECK FOR RESTRAINED D.O.F.
                For Idofn = 1 To NDOFN
                    Ldofn = IDDOF(Idofn, Ipoin)
                    If Ldofn = 1 And SPRIN(Idofn, Ipoin) <> 0# Then
                        PrtString = vbCrLf
                        PrtString += "***  FATAL Error SPRING VALUE On RESTRAINED  D.O.F."
+ vbCrLf
                        PrtString += "     NODE   N. " + Str(Ipoin) + vbCrLf
                        PrtString += "     D.O.F. N. " + Str(Idofn)
                        File1.WriteLine(PrtString)

                        Testo = "***  FATAL Error SPRING VALUE On RESTRAINED  D.O.F." +
vbCrLf
                        Testo += "NODE   N. " + Str$(Ipoin)
                        Testo += " D.O.F. N. " + Str$(Idofn)
                        MsgBox(Testo, vbExclamation, "Warning")
                        Errore = True
                        Exit Sub
                    End If
                Next Idofn
            Next Ipoin
        End If

        'READ GAP definition v3.2
        If Ngaps > 0 Then
            Dim GapX, GapY, GapZ As Integer

            PrtString = vbCrLf
            PrtString += "***  G A P S  ***" + vbCrLf
            PrtString += vbCrLf
            If Ntype = 2 Or Ntype = 3 Then PrtString += "   N.1   N.2     STEP      ux
uy         uz"
            File1.WriteLine(PrtString)

            For Ipoin = 1 To Npoin
                If Ntype = 2 Or Ntype = 3 Then GapX = GAPS(1, Ipoin) : GapY = GAPS(2,
Ipoin) : GapZ = GAPS(3, Ipoin)

                GapsC = 0
                If GapX <> 0 Then GapsC += 1
                If GapY <> 0 Then GapsC += 1
                If GapZ <> 0 Then GapsC += 1
                If GapsC > 1 Then
                    PrtString = vbCrLf
                    PrtString += "***  FATAL Error: GAP ELEMENT MUST BE DEFINED ONLY IN
ONE DIRECTION."
                    File1.WriteLine(PrtString)
                    Testo = "***  FATAL Error: GAP ELEMENT MUST BE DEFINED ONLY IN ONE
DIRECTION." + vbCrLf
                    MsgBox(Testo, vbExclamation, "Warning")
                    Errore = True
                    Exit Sub
                End If

                If GapsC <> 0 Then
```

```
                    PrtString   =   String.Format("{0,5}",   Ipoin.ToString("####0"))   +
String.Format("{0,5}", Ipoin.ToString("####0"))
                    PrtString += String.Format("{0,8}", Kstep.ToString("####0"))
                    If Ntype = 2 Or Ntype = 3 Then
                        PrtString += String.Format("{0,11}", GapX.ToString("####0"))
                        PrtString += String.Format("{0,11}", GapY.ToString("####0"))
                        PrtString += String.Format("{0,11}", GapZ.ToString("####0"))
                    End If
                    File1.WriteLine(PrtString)
                End If

                'CHECK FOR RESTRAINED D.O.F.
                For Idofn = 1 To NDOFN
                    Ldofn = IDDOF(Idofn, Ipoin)
                    If Ldofn = 1 And GAPS(Idofn, Ipoin) <> 0# Then
                        PrtString = vbCrLf
                        PrtString += "***  FATAL Error GAP ELEMENT On RESTRAINED  D.O.F."
+ vbCrLf
                        PrtString += "     NODE   N. " + Str(Ipoin) + vbCrLf
                        PrtString += "     D.O.F. N. " + Str(Idofn) + vbCrLf
                        PrtString += "     GAP WILL BE DELETED"
                        File1.WriteLine(PrtString)

                        Testo = "***  FATAL Error GAP ELEMENT On RESTRAINED  D.O.F." +
vbCrLf
                        Testo += "NODE   N. " + Str$(Ipoin)
                        Testo += "D.O.F. N. " + Str$(Idofn)
                        Testo += "GAP WILL BE DELETED"
                        MsgBox(Testo, vbExclamation, "Warning")
                        Errore = True
                        Exit Sub
                    End If
                Next Idofn
            Next Ipoin
        End If

    End Sub
```

```
    Sub WriteProperties(File1 As StreamWriter, DaDove As String)

        PrtString = vbCrLf : PrtString += vbCrLf
        PrtString += "* * *  M A T E R I A L   P R O P E R T I E S  * * *"
        File1.WriteLine(PrtString)

        ' *** LOOP OVER MATERIAL SETS

        For Isets = 1 To Nmats
            If Ntype = 2 Or Ntype = 3 Then
                PrtString = vbCrLf
                PrtString += "  Set        Young       Poisson     weight        mass " +
vbCrLf
                PrtString += "           modulus      ratio     density     density"
            End If
            File1.WriteLine(PrtString)

            PrtString = String.Format("{0,5}", Isets.ToString("####0")) + "  "
            PrtString += String.Format("{0,12}", PROPS(Isets, 1).ToString("0.0000E+00"))
            PrtString += String.Format("{0,12}", PROPS(Isets, 9).ToString("0.0000E+00"))
            PrtString += String.Format("{0,12}", PROPS(Isets, 6).ToString("0.0000E+00"))
            PrtString += String.Format("{0,12}", PROPS(Isets, 7).ToString("0.0000E+00"))
            File1.WriteLine(PrtString)

            If Ntype = 2 Or Ntype = 3 Then
                PrtString = vbCrLf
                PrtString += "             AREA          J            Jt       K Winkler
width    " + vbCrLf
            End If

            PrtString += "       "
```

```
            PrtString += String.Format("{0,12}", PROPS(Isets, 2).ToString("0.0000E+00"))
            PrtString += String.Format("{0,12}", PROPS(Isets, 3).ToString("0.0000E+00"))
            PrtString += String.Format("{0,12}", PROPS(Isets, 8).ToString("0.0000E+00"))
            PrtString += String.Format("{0,12}", PROPS(Isets, 4).ToString("0.0000E+00"))
            PrtString += String.Format("{0,12}", PROPS(Isets, 5).ToString("0.0000E+00"))
            File1.WriteLine(PrtString)

            If TrazFlag(Isets) <> 0 Then 'v3.2
                PrtString = "           For the previous Set only tension stresses are
allowed"
                File1.WriteLine(PrtString)
            End If
        Next Isets

    End Sub
```

```
    Sub LOADS(Icase As Integer, Iteration As Integer, File1 As StreamWriter)

        ReDim Rload(NDOFT), ELOAD(NEVAB, Nelem)

        If Iteration = 1 Then
            If Icase <= Ncase Then ' it's a load condition
                PrtString = vbCrLf : PrtString += vbCrLf
                PrtString  +=  "================================================="  +
vbCrLf
                PrtString += "           LOAD CONDITION : " + Str(Icase) + vbCrLf
                PrtString += vbCrLf
                PrtString += Titl$(Icase) + vbCrLf
                PrtString  +=  "================================================="  +
vbCrLf
                File1.WriteLine(PrtString)
            Else
                PrtString = vbCrLf : PrtString += vbCrLf
                PrtString  +=  "================================================="  +
vbCrLf
                PrtString += "              LOAD COMBINATION : " + Str(Icase - Ncase) +
vbCrLf
                PrtString += vbCrLf
                PrtString += Tit_Comb(Icase - Ncase) + vbCrLf
                PrtString  +=  "================================================="  +
vbCrLf
                PrtString += vbCrLf : PrtString += vbCrLf
                PrtString += "LOAD CASE      FACTOR" + vbCrLf
                File1.WriteLine(PrtString)
                For Jcase = 1 To Ncase
                    PrtString = "    " + String.Format("{0,5}", Jcase.ToString("####0"))
                    PrtString  +=  String.Format("{0,12}",  Comb_Factor(Icase  -  Ncase,
Jcase).ToString("####0.00000"))
                    File1.WriteLine(PrtString)
                Next Jcase
            End If
        End If

        Call LOADBE(Icase, Iteration, File1)

    End Sub
```

```
    Sub LOADBE(Icase As Integer, Iteration As Integer, File1 As StreamWriter)

        '     LOADS FOR BEAM OR WINKLER ELEMENTS

        Dim Testo As String
        Dim DIREC As String = ""
        Dim Imats, Ldofn, Kstep As Integer
        Dim UNWEI, p1, p2, p3, p4, p5, p6 As Single

        ' *** SELF WEIGHT
```

```
        If Gravity_Case(Icase) <> 0 Then

            If Gravity_Case(Icase) = -1 Then DIREC = "-X"
            If Gravity_Case(Icase) = 1 Then DIREC = " X"
            If Gravity_Case(Icase) = -2 Then DIREC = "-Y"
            If Gravity_Case(Icase) = 2 Then DIREC = " Y"
            If Gravity_Case(Icase) = -3 Then DIREC = "-Z"
            If Gravity_Case(Icase) = 3 Then DIREC = " Z"

            If DIREC = "" Then DIREC = "-Z" : Gravity_Case(Icase) = -3 'default value

            If Iteration = 1 Then
                PrtString = vbCrLf : PrtString += vbCrLf
                PrtString += " GRAVITY LOAD ACTIVE In Global " + DIREC + " DIRECTION"
                File1.WriteLine(PrtString)
            End If

            ' *** DEAL WITH SELF WEIGHT

            If Iteration = 1 Then
                PrtString = vbCrLf
                PrtString += "***  SPAN LOADS DUE TO SELF WEIGHT  ***" + vbCrLf
                PrtString += vbCrLf
                PrtString += " Type  elem. elem. Step       Qx i        Qy i        Qz i
Qx j        Qy j        Qz j"
                File1.WriteLine(PrtString)
            End If

            ' *** LOOP OVER ELEMENT GROUPS

            For Ielem = 1 To Nelem

                Imats = Mater(Ielem)
                UNWEI  =  PROPS(Imats,  2)  *  PROPS(Imats,  6)  *  Gravity_Case(Icase)  /
Math.Abs(Gravity_Case(Icase))
                If Icase > Ncase Then 'it's a combination
                    UNWEI *= GravityAmplif(Icase - Ncase)
                End If

                ReDim TRLOA(6)

                If Gravity_Case(Icase) = 1 Or Gravity_Case(Icase) = -1 Then TRLOA(1) =
UNWEI
                If Gravity_Case(Icase) = 2 Or Gravity_Case(Icase) = -2 Then TRLOA(2) =
UNWEI
                If Gravity_Case(Icase) = 3 Or Gravity_Case(Icase) = -3 Then TRLOA(3) =
UNWEI

                TRLOA(4) = TRLOA(1)
                TRLOA(5) = TRLOA(2)
                TRLOA(6) = TRLOA(3)

                ReDim FLOAD(NEVAB)
                Call UDLOBE(Icase, Ielem)

                For Ievab = 1 To NEVAB
                    Ldofn = LMDOF(Ievab, Ielem)
                    If Ldofn <> 0 Then Rload(Ldofn) += FLOAD(Ievab)
                Next Ievab

                Kstep = 1
                If Iteration = 1 Then
                    LoadType(Icase, Ielem) = 2
                    PrtString       =       String.Format("{0,5}",       LoadType(Icase,
Ielem).ToString("####0"))
                    PrtString   +=   String.Format("{0,6}",   Ielem.ToString("####0"))   +
String.Format("{0,6}",       Ielem.ToString("####0"))       +       String.Format("{0,6}",
Kstep.ToString("####0"))
                    PrtString += "  "
                    PrtString += String.Format("{0,12}", TRLOA(1).ToString("0.0000E+00"))
```

```
                    PrtString += String.Format("{0,12}", TRLOA(2).ToString("0.0000E+00"))
                    PrtString += String.Format("{0,12}", TRLOA(3).ToString("0.0000E+00"))

                    PrtString += String.Format("{0,12}", TRLOA(1).ToString("0.0000E+00"))
                    PrtString += String.Format("{0,12}", TRLOA(2).ToString("0.0000E+00"))
                    PrtString += String.Format("{0,12}", TRLOA(3).ToString("0.0000E+00"))
                    File1.WriteLine(PrtString)
                End If
            Next Ielem
        End If 'gravity

        ' *** POINT LOADS

        If NpointLoads(Icase) > 0 Then

            For Ipoin = 1 To Npoin
                If Ipoin = 1 And Iteration = 1 Then
                    PrtString = vbCrLf
                    PrtString += "***  P O I N T    L O A D S   ***" + vbCrLf
                    PrtString += vbCrLf
                    PrtString += " NODE       Fx            Fy            Fz           Mxx
Myy            Mzz"
                    File1.WriteLine(PrtString)
                End If

                ReDim FLOAD(NDOFN)

                FLOAD(1) = PointLoad(Icase, Ipoin, 1)
                FLOAD(2) = PointLoad(Icase, Ipoin, 2)
                FLOAD(3) = PointLoad(Icase, Ipoin, 3)
                FLOAD(4) = PointLoad(Icase, Ipoin, 4)
                FLOAD(5) = PointLoad(Icase, Ipoin, 5)
                FLOAD(6) = PointLoad(Icase, Ipoin, 6)

                For Idofn = 1 To NDOFN
                    Ldofn = IDDOF(Idofn, Ipoin)
                    If Ldofn = 0 And FLOAD(Idofn) <> 0# Then
                        If Iteration = 1 Then
                            PrtString = vbCrLf
                            PrtString += "***  WARNING Point LOAD On RESTRAINED  D.O.F."
+ vbCrLf
                            PrtString += "              NODE   N. " + Str(Ipoin) + vbCrLf
                            PrtString += "              D.O.F. N. " + Str(Idofn)
                            File1.WriteLine(PrtString)

                            Testo = "***   WARNING Point LOAD On RESTRAINED   D.O.F." +
vbCrLf
                            Testo += "NODE   N. " + Str$(Ipoin)
                            Testo += " D.O.F. N. " + Str$(Idofn)
                            MsgBox(Testo, vbExclamation, "Warning")
                        End If
                    Else
                        Rload(Ldofn) += FLOAD(Idofn)
                    End If
                Next Idofn
                If Iteration = 1 Then
                    p1 = PointLoad(Icase, Ipoin, 1) : p2 = PointLoad(Icase, Ipoin, 2) :
p3 = PointLoad(Icase, Ipoin, 3)
                    p4 = PointLoad(Icase, Ipoin, 4) : p5 = PointLoad(Icase, Ipoin, 5) :
p6 = PointLoad(Icase, Ipoin, 6)
                    If p1 <> 0 Or p2 <> 0 Or p3 <> 0 Then
                        PrtString = String.Format("{0,5}", Ipoin.ToString("####0")) + " "
                        PrtString += String.Format("{0,15}", p1.ToString("0.0000E+00"))
                        PrtString += String.Format("{0,15}", p2.ToString("0.0000E+00"))
                        PrtString += String.Format("{0,15}", p3.ToString("0.0000E+00"))
                        PrtString += String.Format("{0,15}", p4.ToString("0.0000E+00"))
                        PrtString += String.Format("{0,15}", p5.ToString("0.0000E+00"))
                        PrtString += String.Format("{0,15}", p6.ToString("0.0000E+00"))
                        File1.WriteLine(PrtString)
                    End If
                End If
```

```
            Next Ipoin 'point loads
        End If

        ' *** SPAN LOADS

        If NspanLoads(Icase) > 0 Then
            For Ielem = 1 To Nelem

                ReDim TRLOA(6), FLOAD(NEVAB)

                TRLOA(1) = Dload(Icase, Ielem, 1)
                TRLOA(2) = Dload(Icase, Ielem, 2)
                TRLOA(3) = Dload(Icase, Ielem, 3)
                TRLOA(4) = Dload(Icase, Ielem, 4)
                TRLOA(5) = Dload(Icase, Ielem, 5)
                TRLOA(6) = Dload(Icase, Ielem, 6)

                If Iteration = 1 And Ielem = 1 Then
                    PrtString = vbCrLf
                    PrtString += "***   S P A N    L O A D S    ***" + vbCrLf
                    PrtString += vbCrLf
                    PrtString += "       First Last  Gen." + vbCrLf
                    PrtString += " Type  elem. elem. step  context     Qx i         Qy i
Qz i        Qx j        Qy j        Qz j"
                    File1.WriteLine(PrtString)
                End If
                If Iteration = 1 Then
                    Kstep = 1
                    If LoadType(Icase, Ielem) > 0 Then
                        PrtString      =      String.Format("{0,5}",      LoadType(Icase,
Ielem).ToString("####0"))
                        PrtString += String.Format("{0,6}", Ielem.ToString("####0"))
                        PrtString += String.Format("{0,6}", Ielem.ToString("####0"))
                        PrtString += String.Format("{0,6}", Kstep.ToString("####0"))
                        PrtString += "  global "

                        PrtString               +=               String.Format("{0,12}",
TRLOA(1).ToString("0.0000E+00"))
                        PrtString               +=               String.Format("{0,12}",
TRLOA(2).ToString("0.0000E+00"))
                        PrtString               +=               String.Format("{0,12}",
TRLOA(3).ToString("0.0000E+00"))
                        PrtString               +=               String.Format("{0,12}",
TRLOA(4).ToString("0.0000E+00"))
                        PrtString               +=               String.Format("{0,12}",
TRLOA(5).ToString("0.0000E+00"))
                        PrtString               +=               String.Format("{0,12}",
TRLOA(6).ToString("0.0000E+00"))
                        File1.WriteLine(PrtString)
                    End If
                End If

                If LoadType(Icase, Ielem) = 1 Then Call UDLOBE(Icase, Ielem)
                If LoadType(Icase, Ielem) = 2 Then Call TRLOAD(Icase, Ielem)
                If LoadType(Icase, Ielem) > 0 Then
                    For Ievab = 1 To NEVAB
                        Ldofn = LMDOF(Ievab, Ielem)
                        If Ldofn <> 0 Then Rload(Ldofn) += FLOAD(Ievab)
                    Next Ievab
                End If
            Next Ielem 'Span Loads
        Else
            If Iteration = 1 Then
                PrtString = vbCrLf
                PrtString += "***   S P A N    L O A D S    ***" + vbCrLf
                PrtString += "      none"
                PrtString += vbCrLf
                File1.WriteLine(PrtString)
            End If
        End If
```

```
    End Sub
```

```
    Sub ComputeStresses(ByVal Icase As Integer, FileWork3 As String)

        Dim Nele1, Nele2 As Integer

        ' *** LOOP OVER ELEMENT GROUPS

        Nele1 = 1
        Nele2 = Nele1 + Nelem - 1

        'CALL STRESS SUBROUTINES

        If Ntype = 2 Or Ntype = 3 Then
            Call STRBE(Icase, Nele1, Nele2, FileWork3)
        End If

        Call CleanValues(Icase)

    End Sub
```

```
    Sub TRLOAD(Icase As Integer, Ielem As Integer)

        '     TRAPEZOIDAL LOAD FOR BEAM, WINKLER ELEMENT (IELEM)
        '     CALCULATION OF GLOBAL LOAD VECTOR

        Dim Idof1, Ldofn As Integer
        Dim LocLoa(12), Delt1(3) As Double
        Dim Leng2, Lengt, p1, p2, p3, p4, p5, p6 As Double

        'transform linear loads from global to local coordinates: transformation T matrices
are calculated only once after reading data
        For Idofn = 1 To NDOFN
            For Jdofn = 1 To NDOFN
                LocLoa(Idofn) += Tmat(Ielem, Idofn, Jdofn) * TRLOA(Jdofn)
            Next Jdofn
        Next Idofn

        'calculate element length
        Leng2 = 0.0
        For Idime = 1 To NDIME
            Delt1(Idime) = XYCOO(Idime + 3, Ielem) - XYCOO(Idime, Ielem)
            Leng2 += Delt1(Idime) * Delt1(Idime)
        Next Idime
        Lengt = Math.Sqrt(Leng2)

        'uniform loads: value is constant
        p1 = LocLoa(1) 'load in the direction of local axis 1, at node i
        p2 = LocLoa(2) 'load in the direction of local axis 2, at node i
        p3 = LocLoa(3) 'load in the direction of local axis 3, at node i
        p4 = LocLoa(4) 'load in the direction of local axis 1, at node j
        p5 = LocLoa(5) 'load in the direction of local axis 2, at node j
        p6 = LocLoa(6) 'load in the direction of local axis 3, at node j

        DloaL(Icase, Ielem, 1) += p1 '1-3 = indexes for i-node
        DloaL(Icase, Ielem, 2) += p2
        DloaL(Icase, Ielem, 3) += p3
        DloaL(Icase, Ielem, 4) += p4 '4-6 = indexes for j-node
        DloaL(Icase, Ielem, 5) += p5
        DloaL(Icase, Ielem, 6) += p6

        'node i
        LocLoa(1) = p1 * Lengt / 2 + (p4 - p1) * Lengt / 6
        LocLoa(2) = p2 * Lengt / 2 + 3 * (p5 - p2) * Lengt / 20
        LocLoa(3) = p3 * Lengt / 2 + 3 * (p6 - p3) * Lengt / 20
        LocLoa(4) = 0
        LocLoa(5) = -p3 * Leng2 / 12.0 - (p6 - p3) * Leng2 / 30
```

```
        LocLoa(6) = p2 * Leng2 / 12.0 + (p5 - p2) * Leng2 / 30

        'node j
        LocLoa(7) = p1 * Lengt / 2 + (p4 - p1) * Lengt / 3
        LocLoa(8) = p2 * Lengt / 2 + 7 * (p5 - p2) * Lengt / 20
        LocLoa(9) = p3 * Lengt / 2 + 7 * (p6 - p3) * Lengt / 20
        LocLoa(10) = -LocLoa(4)
        LocLoa(11) = -(-p3 * Leng2 / 12.0 - (p6 - p3) * Leng2 / 20)
        LocLoa(12) = -(p2 * Leng2 / 12.0 + (p5 - p2) * Leng2 / 20)

        'transform force vector from local to global coordinates
        'change rows and columns of Tmat to obtain the transpose of the matrix
        For Idofn = 1 To NDOFN
            Idof1 = Idofn + NDOFN
            For Jdofn = 1 To NDOFN
                Ldofn = Jdofn + NDOFN
                FLOAD(Idofn) += Tmat(Ielem, Jdofn, Idofn) * LocLoa(Jdofn)
                FLOAD(Idof1) += Tmat(Ielem, Jdofn, Idofn) * LocLoa(Ldofn)
            Next Jdofn
        Next Idofn

        For Ievab = 1 To NEVAB
            ELOAD(Ievab, Ielem) += FLOAD(Ievab)
        Next Ievab

    End Sub
```

```
    Sub UDLOBE(Icase As Integer, Ielem As Integer)

        '     UNIFORM LOAD FOR BEAM, WINKLER ELEMENT (IELEM)
        '     CALCULATION OF GLOBAL LOAD VECTOR from local linear loads

        Dim Idof1, Ldofn As Integer
        Dim LocLoa(12), Delt1(3) As Double
        Dim Leng2, Lengt, p1, p2, p3 As Double

        'transform linear loads from global to local coordinates: transformation T matrices
are calculated only once after reading data
        For Idofn = 1 To NDOFN
            For Jdofn = 1 To NDOFN
                LocLoa(Idofn) += Tmat(Ielem, Idofn, Jdofn) * TRLOA(Jdofn)
            Next Jdofn
        Next Idofn

        'calculate element length
        Leng2 = 0.0
        For Idime = 1 To NDIME
            Delt1(Idime) = XYCOO(Idime + 3, Ielem) - XYCOO(Idime, Ielem)
            Leng2 += Delt1(Idime) * Delt1(Idime)
        Next Idime
        Lengt = Math.Sqrt(Leng2)

        'uniform loads: value is constant
        p1 = LocLoa(1) 'carico lungo l'asse locale 1
        p2 = LocLoa(2) 'carico lungo l'asse locale 2
        p3 = LocLoa(3) 'carico lungo l'asse locale 3

        DloaL(Icase, Ielem, 1) += p1 '1-3 = indexes for i-node
        DloaL(Icase, Ielem, 2) += p2
        DloaL(Icase, Ielem, 3) += p3
        DloaL(Icase, Ielem, 4) += p1 '4-6 = indexes for j-node
        DloaL(Icase, Ielem, 5) += p2
        DloaL(Icase, Ielem, 6) += p3

        'node i
        LocLoa(1) = p1 * Lengt / 2.0
        LocLoa(2) = p2 * Lengt / 2.0
        LocLoa(3) = p3 * Lengt / 2.0
        LocLoa(4) = 0
```

```
        LocLoa(5) = -p3 * Leng2 / 12.0
        LocLoa(6) = p2 * Leng2 / 12.0

        'node j
        LocLoa(7) = LocLoa(1)
        LocLoa(8) = LocLoa(2)
        LocLoa(9) = LocLoa(3)
        LocLoa(10) = -LocLoa(4)
        LocLoa(11) = -LocLoa(5)
        LocLoa(12) = -LocLoa(6)

        'transform force vector from local to global coordinates
        'change rows and columns of Tmat to obtain the transpose of the matrix
        For Idofn = 1 To NDOFN
            Idof1 = Idofn + NDOFN
            For Jdofn = 1 To NDOFN
                Ldofn = Jdofn + NDOFN
                FLOAD(Idofn) += Tmat(Ielem, Jdofn, Idofn) * LocLoa(Jdofn)
                FLOAD(Idof1) += Tmat(Ielem, Jdofn, Idofn) * LocLoa(Ldofn)
            Next Jdofn
        Next Idofn

        For Ievab = 1 To NEVAB
            ELOAD(Ievab, Ielem) += FLOAD(Ievab)
        Next Ievab

    End Sub
```

```
    Function MIN0(Arg1 As Integer, Arg2 As Integer) As Integer

        If Arg1 < Arg2 Then
            MIN0 = Arg1
        Else
            MIN0 = Arg2
        End If

    End Function
```

```
    Sub FindDOF(ByVal Kdofn As Integer, ByRef Element As Integer, ByRef D_O_F$)

        'new routine written to find the d.o.f. with a lack in constraints - 22 jul 2008

        'Kdofn is the d.o.f. with zero or negative pivot
        Dim Dofnn(NDOFN) As String
        Dim Idof1 As Integer

        If Ntype = 2 Or Ntype = 3 Then
            Dofnn(1) = "x-displacement"
            Dofnn(2) = "y-displacement"
            Dofnn(3) = "z-displacement"
            Dofnn(4) = "xx-rotation"
            Dofnn(5) = "yy-rotation"
            Dofnn(6) = "zz-rotation"
        End If

        For Ielem = 1 To Nelem
            For Idofn = 1 To NDOFN
                Idof1 = Idofn + NDOFN
                If LMDOF(Idofn, Ielem) = Kdofn Then
                    Element = Ielem
                    D_O_F$ = Dofnn(Idofn)
                    Exit Sub
                End If
                If LMDOF(Idof1, Ielem) = Kdofn Then
                    Element = Ielem
                    D_O_F$ = Dofnn(Idofn)
                    Exit Sub
```

```
                End If
            Next Idofn
        Next Ielem

    End Sub
```

```
    Sub OutSTRBE(ByVal Icase As Integer, ByVal Nele1 As Integer, ByVal Nele2 As Integer,
File1 As StreamWriter, File2 As StreamWriter)

        Dim Delt1(3) As Single
        Dim Leng2, Lengt, Dist1, AlfaX, AlfaY As Single
        Dim ExcelString As String

        ExcelString = "***     M E M B E R   F O R C E S      ***" + vbCrLf
        ExcelString += vbCrLf
        ExcelString += "* positive sign For tension axial forces *" + vbCrLf
        ExcelString += vbCrLf
        ExcelString += "                                      AXIAL         SHEAR         SHEAR
TORSION      BENDING      BENDING " + vbCrLf
        ExcelString += "ELEMENT  NODE        DIST          R1            R2            R3
M1           M2           M3       Icase"
        File2.WriteLine(ExcelString)

        PrtString = vbCrLf : PrtString += vbCrLf
        PrtString += "***     M E M B E R   F O R C E S      ***" + vbCrLf
        PrtString += vbCrLf
        PrtString += "* positive sign For tension axial forces *" + vbCrLf
        PrtString += vbCrLf
        PrtString += "                                      AXIAL         SHEAR         SHEAR
TORSION      BENDING      BENDING " + vbCrLf
        PrtString += "ELEMENT  NODE        DIST          R1            R2            R3
M1           M2           M3"
        File1.WriteLine(PrtString)

        ' *** LOOP OVER ELEMENTS
        For Ielem = Nele1 To Nele2
            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)
            Dist1 = 0#
            AlfaX = Delt1(1) / Lengt
            AlfaY = Delt1(2) / Lengt

            ' *** CALCULATE MAXIMUM BENDING MOMENT
            'not yet done

            ' *** OUTPUT FOR EXCEL (TXT) FILE
            ExcelString = String.Format("{0,5}", Ielem.ToString("####0")) + " "
            ExcelString += String.Format("{0,7}", IJINC(Ielem, 1).ToString("####0"))
            ExcelString += String.Format("{0,13}", Dist1.ToString("0.0000E+00"))
            For Idofn = 1 To NDOFN
                ExcelString    +=    String.Format("{0,13}",    Stre1(Icase,    Ielem,
Idofn).ToString("0.0000E+00"))
            Next Idofn
            ExcelString += String.Format("{0,7}", Icase.ToString("####0"))
            File2.WriteLine(ExcelString)

            PrtString = String.Format("{0,5}", Ielem.ToString("####0")) + " "
            PrtString += String.Format("{0,7}", IJINC(Ielem, 1).ToString("####0"))
            PrtString += String.Format("{0,13}", Dist1.ToString("0.0000E+00"))
            For Idofn = 1 To NDOFN
                PrtString    +=    String.Format("{0,13}",    Stre1(Icase,    Ielem,
Idofn).ToString("0.0000E+00"))
            Next Idofn
            File1.WriteLine(PrtString)
```

```
            If Ntype <> 3 Then
                'if Dload(Icase, Ielem, 2)<>0 or Dload(Icase, Ielem, 3)<>0 or Dload(Icase,
Ielem, 5)<>0 or Dload(Icase, Ielem, 6)<>0 then
                '    If Lflag = 1 Then
                '        PrtString = "                              "
                '      PrtString += String.Format("{0,13}", XTeq0.ToString("0.0000E+00"))
+ "  "
                '                                  PrtString    +=    String.Format("{0,13}",
Moment.ToString("0.0000E+00"))
                '      PrtString += String.Format("{0,13}", Dist1.ToString("0.0000E+00"))
                '        File1.WriteLine(PrtString)
                '    End If
                'End If
            End If

            ' *** OUTPUT FOR EXCEL (TXT) FILE
            ExcelString = String.Format("{0,5}", Ielem.ToString("####0")) + " "
            ExcelString += String.Format("{0,7}", IJINC(Ielem, 2).ToString("####0"))
            ExcelString += String.Format("{0,13}", Lengt.ToString("0.0000E+00"))
            For Idofn = 1 To NDOFN
                ExcelString    +=    String.Format("{0,13}",    Stre2(Icase,    Ielem,
Idofn).ToString("0.0000E+00"))
            Next Idofn
            ExcelString += String.Format("{0,7}", Icase.ToString("####0"))
            File2.WriteLine(ExcelString)

            PrtString = "      "
            PrtString += String.Format("{0,7}", IJINC(Ielem, 2).ToString("####0"))
            PrtString += String.Format("{0,13}", Lengt.ToString("0.0000E+00"))
            For Idofn = 1 To NDOFN
                PrtString    +=    String.Format("{0,13}",    Stre2(Icase,    Ielem,
Idofn).ToString("0.0000E+00"))
            Next Idofn
            File1.WriteLine(PrtString)
            PrtString = "-------------------------------------------------------------------
----------------------------------------"
            File1.WriteLine(PrtString)
        Next Ielem

        ExcelString = ""
        File2.WriteLine(ExcelString)

        PrtString = vbCrLf
        File1.WriteLine(PrtString)

    End Sub
```

```
    Sub WriteResults(ByVal Icase As Integer, File1 As StreamWriter, File2 As StreamWriter)

        Dim NOUTP As Integer = NDOFN
        Dim Sumx, Sumy, Sumz, Suxx, Suyy, Suzz As Double

        'following rows come from BACKSU
        If Ntype = 2 Or Ntype = 3 Then
            PrtString = vbCrLf : PrtString += vbCrLf
            PrtString += "***  D I S P L A C E M E N T S  ***" + vbCrLf
            PrtString += vbCrLf
            PrtString += " NODE        X-displ.        Y-displ.        Z-displ.       XX-
rotat.      YY-rotat.      ZZ-rotat."
        End If
        File1.WriteLine(PrtString)

        For Ipoin = 1 To Npoin
            PrtString = String.Format("{0,5}", Ipoin.ToString("####0")) + " "
            For Idofn = 1 To NDOFN
                PrtString    +=    String.Format("{0,15}",    Displ(Icase,    Ipoin,
Idofn).ToString("0.0000E+00"))
            Next Idofn
```

```
            File1.WriteLine(PrtString)
        Next Ipoin

        'following rows come from REACT
        PrtString = vbCrLf : PrtString += vbCrLf
        PrtString += "***  P O I N T   F O R C E S  &  R E A C T I O N S  ***" + vbCrLf
        PrtString += vbCrLf

        If Ntype = 2 Then PrtString += " NODE         Fx              Fy              Fz
Mxx            Myy            Mzz"
        If Ntype = 3 Then PrtString += " NODE         Fx              Fy              Fz
Mxx            Myy            Mzz        Sigma t"
        File1.WriteLine(PrtString)

        For Ipoin = 1 To Npoin
            PrtString = String.Format("{0,5}", Ipoin.ToString("####0"))
            PrtString      +=      String.Format("{0,15}",      TREAC(Icase,      Ipoin,
1).ToString("0.0000E+00"))
            PrtString      +=      String.Format("{0,15}",      TREAC(Icase,      Ipoin,
2).ToString("0.0000E+00"))
            PrtString      +=      String.Format("{0,15}",      TREAC(Icase,      Ipoin,
3).ToString("0.0000E+00"))
            PrtString      +=      String.Format("{0,15}",      TREAC(Icase,      Ipoin,
4).ToString("0.0000E+00"))
            PrtString      +=      String.Format("{0,15}",      TREAC(Icase,      Ipoin,
5).ToString("0.0000E+00"))
            PrtString      +=      String.Format("{0,15}",      TREAC(Icase,      Ipoin,
6).ToString("0.0000E+00"))
            If Ntype = 3 Then
                'PrtString       +=       String.Format("{0,15}",       SoilStress(Icase,
Ipoin).ToString("0.0000E+00"))
            End If
            File1.WriteLine(PrtString)

        Next Ipoin

        If Ntype = 2 Or Ntype = 3 Then
            For Ipoin = 1 To Npoin
                Sumx += TREAC(Icase, Ipoin, 1)
                Sumy += TREAC(Icase, Ipoin, 2)
                Sumz += TREAC(Icase, Ipoin, 3)
                Suxx += TREAC(Icase, Ipoin, 4)
                Suyy += TREAC(Icase, Ipoin, 5)
                Suzz += TREAC(Icase, Ipoin, 6)
            Next Ipoin
        End If

        'round values
        For Ipoin = 1 To Npoin
            If Math.Abs(Sumx) < 0.000000001 Then Sumx = 0#
            If Math.Abs(Sumy) < 0.000000001 Then Sumy = 0#
            If Math.Abs(Sumz) < 0.000000001 Then Sumz = 0#
            If Math.Abs(Suxx) < 0.000000001 Then Suxx = 0#
            If Math.Abs(Suyy) < 0.000000001 Then Suyy = 0#
            If Math.Abs(Suzz) < 0.000000001 Then Suzz = 0#
        Next Ipoin

        PrtString = vbCrLf
        PrtString += "TOTAL"
        PrtString += String.Format("{0,15}", Sumx.ToString("0.0000E+00"))
        PrtString += String.Format("{0,15}", Sumy.ToString("0.0000E+00"))
        PrtString += String.Format("{0,15}", Sumz.ToString("0.0000E+00"))
        PrtString += String.Format("{0,15}", Suxx.ToString("0.0000E+00"))
        PrtString += String.Format("{0,15}", Suyy.ToString("0.0000E+00"))
        PrtString += String.Format("{0,15}", Suzz.ToString("0.0000E+00"))
        File1.WriteLine(PrtString)

        'CALL STRESS output SUBROUTINES
        If Ntype = 2 Or Ntype = 3 Then
            Call OutSTRBE(Icase, 1, Nelem, File1, File2)
        End If
```

```
    End Sub
```

```
    Function Kpos(Nrow As Integer, Ncol As Integer)

        If Ncol < Nrow Then 'swap numbers
            Dim Dummy As Integer
            Dummy = Nrow
            Nrow = Ncol
            Ncol = Dummy
        End If

        Kpos = 0
        For Irow = 1 To Nrow - 1
            For Icol = Irow To 12
                Kpos += 1
            Next Icol
        Next Irow

        For Icol = Nrow To Ncol
            Kpos += 1
        Next Icol

    End Function
```

```
    Sub FillTrasf(Ielem As Integer, ByRef Trasf(,) As Double)
        'fill local to global transformation matrix Trasf
        'with Tmat(6,6)

        Dim Irow1, Icol1 As Integer

        ReDim Trasf(12, 12)

        'copy Tmat array on 1-6 rows and 1-6 columns
        For Irows = 1 To 6
            For Icols = 1 To 6
                Trasf(Irows, Icols) = Tmat(Ielem, Irows, Icols)
            Next Icols
        Next Irows

        'copy Tmat array on 7-12 rows and 7-12 columns
        For Irows = 1 To 6
            Irow1 = Irows + 6
            For Icols = 1 To 6
                Icol1 = Icols + 6
                Trasf(Irow1, Icol1) = Tmat(Ielem, Irows, Icols)
            Next Icols
        Next Irows

    End Sub
```

```
    Sub CleanValues(ByVal Icase As Integer)
        'if member forces values are very small, they are rounded to zero

        Dim Nele1, Nele2 As Integer
        Dim MaxValue As Single = 0.00000001

        'find Max value
        Nele1 = 1
        Nele2 = Nele1 + Nelem - 1

        For Ielem = Nele1 To Nele2
            If Ntype = 1 Then
                If MaxValue < Math.Abs(Stre1(Icase, Ielem, 1)) Then MaxValue =
Math.Abs(Stre1(Icase, Ielem, 1))
```

```
                If MaxValue < Math.Abs(Stre2(Icase, Ielem, 1)) Then MaxValue =
Math.Abs(Stre2(Icase, Ielem, 1))
            Else
                If MaxValue < Math.Abs(Stre1(Icase, Ielem, 1)) Then MaxValue =
Math.Abs(Stre1(Icase, Ielem, 1))
                If MaxValue < Math.Abs(Stre1(Icase, Ielem, 2)) Then MaxValue =
Math.Abs(Stre1(Icase, Ielem, 2))
                If MaxValue < Math.Abs(Stre1(Icase, Ielem, 3)) Then MaxValue =
Math.Abs(Stre1(Icase, Ielem, 3))
                If MaxValue < Math.Abs(Stre2(Icase, Ielem, 1)) Then MaxValue =
Math.Abs(Stre2(Icase, Ielem, 1))
                If MaxValue < Math.Abs(Stre2(Icase, Ielem, 2)) Then MaxValue =
Math.Abs(Stre2(Icase, Ielem, 2))
                If MaxValue < Math.Abs(Stre2(Icase, Ielem, 3)) Then MaxValue =
Math.Abs(Stre2(Icase, Ielem, 3))
            End If
        Next Ielem

        'round them if it's the case
        Nele1 = 1
        Nele2 = Nele1 + Nelem - 1

        For Ielem = Nele1 To Nele2
            If Ntype = 1 Then
                If Math.Abs(Stre1(Icase, Ielem, 1)) / MaxValue < 0.000001 Then
Stre1(Icase, Ielem, 1) = 0
                If Math.Abs(Stre2(Icase, Ielem, 1)) / MaxValue < 0.000001 Then
Stre2(Icase, Ielem, 1) = 0
            Else
                If Math.Abs(Stre1(Icase, Ielem, 1)) / MaxValue < 0.000001 Then
Stre1(Icase, Ielem, 1) = 0
                If Math.Abs(Stre1(Icase, Ielem, 2)) / MaxValue < 0.000001 Then
Stre1(Icase, Ielem, 2) = 0
                If Math.Abs(Stre1(Icase, Ielem, 3)) / MaxValue < 0.000001 Then
Stre1(Icase, Ielem, 3) = 0
                If Math.Abs(Stre2(Icase, Ielem, 1)) / MaxValue < 0.000001 Then
Stre2(Icase, Ielem, 1) = 0
                If Math.Abs(Stre2(Icase, Ielem, 2)) / MaxValue < 0.000001 Then
Stre2(Icase, Ielem, 2) = 0
                If Math.Abs(Stre2(Icase, Ielem, 3)) / MaxValue < 0.000001 Then
Stre2(Icase, Ielem, 3) = 0
            End If
        Next Ielem

    End Sub
```

```
    Sub ElapsedTime(DeltaTime As Integer, ByRef Time1$)

        Dim Hour1$ = "0", Minute1$ = "0", Second1$ = "0"

        If DeltaTime > 3600 Then
            Dim Hours As Integer = Int(DeltaTime / 3600)
            DeltaTime = DeltaTime - Hours * 3600
            Hour1$ = Str$(Hours)
        End If

        If DeltaTime > 60 Then
            Dim Minutes As Integer = Int(DeltaTime / 60)
            DeltaTime = DeltaTime - Minutes * 60
            Minute1$ = Str$(Minutes)
        End If

        Second1$ = Str$(Int(DeltaTime))

        Time1$ = LTrim$(Hour1$) & ":" & LTrim$(Minute1$) & ":" & LTrim$(Second1$)

    End Sub
```

```
Sub KillWorkFiles()

    Dim FileSelfWeight = Left$(DataFile, Len(DataFile) - 4) & ".swe"
    If Dir(FileSelfWeight, 0) <> "" Then Kill(FileSelfWeight)
    If Dir(myPath & "release.tmp") <> "" Then Kill(myPath & "release.tmp")
    If Dir(myPath + "WORK1", 0) <> "" Then Kill(myPath + "WORK1")
    If Dir(myPath + "WORK2", 0) <> "" Then Kill(myPath + "WORK2")
    If Dir(myPath + "WORK3", 0) <> "" Then Kill(myPath + "WORK3")

End Sub
```