UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA CIVIL
DEPARTAMENTO DE CONSTRUÇÃO CIVIL

SISTEMAS LINEARES NO MÉTODO DOS ELEMENTOS FINITOS

UMA ABORDAGEM DO DESEMPENHO DE MÉTODOS DE ARMAZENAGEM DE VARIÁVEIS UTILIZANDO A LINGUAGEM PASCAL EM MICROCOMPUTADORES DO TIPO PC

Autor: Jorge Abeid Neto

Orientador: Prof. Dr. Francisco Antonio Menezes

Dissertação de mestrado apresentada à Faculdade de Engenharia Civil da Universidade Estadual de Campinas, para obtenção do título de Mestre em Engenharia Civil, Área de Concentração: Estruturas

CAMPINAS
1994

SISTEMAS LINEARES NO MÉTODO DOS ELEMENTOS FINITOS

UMA ABORDAGEM DO DESEMPENHO DE MÉTODOS DE ARMAZENAGEM DE VARIÁVEIS UTILIZANDO A LINGUAGEM PASCAL EM MICROCOMPUTADORES DO TIPO PC

Dissertação defendida e aprovada, em 09 de dezembro de 1994, pela banca examinadora constituida pelos professores:

Prof. Dr. Francisco Antonio Menezes -Orientador

Prof. Dr. Ronaldo Garcia de Figueiredo -Professor Convidado

Prof. Dr. Aloísio Ernesto Assan

Ao Criador, que além de me dar a vida, me concedeu a oportunidade e a faculdade da compreensão deste infinitésimo, que Sua maravilhosa obra criadora, pode proporcionar.

Agradeço aos meus Pais, que tornaram possível esse periodo de minha vida.

Agradeço à minha Monica, minha doce esposa, que pela sua dedicação e pelo seu carinho, impulsionou mais esta etapa da minha existencia.

Agradeço aos meus Cesar Augusto, Ana Cristina e Mariana, filhos tão amados, que me impuseram a necessidade do exemplo.

Agradeço aos meus mestres e professores que me acolheram com orientação e dedicação.

Agardeço ao Professor Persio Leister de Almeida Barros, pela sua valorosa ajuda nos meandros tortuosos da Engenharia Computacional.

Agradeço aos estagiários do nosso Setor de Computação, Fábio, Renato, Mário e Adriana, que me aturaram com tanta paciencia.

Agradeço àqueles, que no silencio de suas orações, pediram inspiração e ajuda para essa àrdua caminhada.

SUMÁRIO

RESUMO

A análise de fenômenos físicos, via Método dos Elementos Finitos, termina por produzir sistemas de equações de grandes proporções, tornando problemático o uso de microcomputadores do tipo PC, gerenciados por sistemas DOS, tendo em vista a grande quantidade de variáveis que precisam ser armazenadas para a sua solução.

O trabalho aqui apresentado investiga métodos e técnicas que possibilitem o uso daqueles equipamentos, que são acessíveis ou disponíveis por profissionais de Engenharia em seus escritórios, para tal fim, proporcionando a popularização, se assim se pode dizer, do referido método.

Como veículo gerador do problema foi utilizado a Análise Matricial de Estruturas, tendo em vista sua simplicidade, desenvolvendo-se um programa de cálculo de estruturas reticuladas planas, em ambiente Borland Pascal, na sua versão 7.0. O programa elaborado aplica o método de eliminação de Gauss na construção de um algoritmo que promove a fatorização LU, para solução do sistema de equações obtido.

O programa assim elaborado serve de base para aplicação das diversas técnicas de armazenamento de dados por ele produzidos, proporcionando a montagem de tres algoritmos, cada um referente a um método, de tal forma a apresentar os resultados comparativos entre eles no tocante a tempo de processamento e capacidade de armazenamento, quando aplicados à analise de tres estruturas propostas. Os resultados foram obtidos em microcomputadores do tipo PC, modelo 486 DX de 33MHz.

ABSTRACT

The analysis of physical phenomenon by the Finite Element Method, leads away to large linear systems and creates some difficulty to solve it on DOS personal computers (PC), because there is an excessive number of variables to store in machine RAM .

This paper presents an investigation into the techniques and methods to do possible the use the PC equipments. Such machines are present in the majorities of Engineering offices and this is the reason of its importance.

The Matricial Structure Analysis was used as the generator of the linear equations systems , because it is an unsophisticated way to do that, with the development of a program to analyse bidimensional frameworks in Borland Pascal 7.0. This program employs a Gaussian algorithm by LU fatorization to solve the linear system.

So, the program elaborated is the application basis to several techniques of data storage offers itself, providing the construction of three algorithms, each one relative a particular method, thus the comparison of performance and data storage capacity are showed when the three methods are used to analyse three particular frameworks.

Microcomputers PC model 486 DX , 33 Mhz of clock, are used to obtain results.

LISTA DE TABELAS

LISTA DE FIGURAS

1-INTRODUÇÃO

1.1-Objetivos

O presente trabalho tem por enfoque analisar o desempenho de alguns métodos de armazenagem de grandes quantidades de variáveis, armazenagem essa, obrigatória para a solução de sistemas de equações lineares, gerados pela aplicação do método dos elementos finitos, aplicado à analise de fenomenos fisicos em geral, tendo em vista o uso de microcomputadores do tipo PC (Personal Computers), gerenciados por sistema DOS (Disk Operating System), em ambiente Pascal, mais precisamente o compilador Borland Pascal na sua versão 7.0

Há que se considerar, em resposta a um eventual questionamento da oportunidade deste estudo, que mesmo já estando disponíveis sistemas operacionais que resolvem esse problema, o do armazenamento de grandes quantidades de variáveis, os compiladores não operam ainda nesses ambientes e mesmo após esse evento, o surgimento de compiladores que funcionem nesses sistemas operacionais, haverá um tempo para que os profissionais de Engenharia se adequem a essa futura realidade, o que significa sucateamento de softwares e equipamentos com os respectivos custos de investimento e treinamento em novas técnicas. Assim, acredita-se que o presente trabalho além de extremamente oportuno e importante como pode ser comprovado no capítulo 7, tenha vida útil garantida por vários anos.

Os algoritmos aqui apresentados foram elaborados para o caso estrutural de sistemas reticulados planos, sendo portanto, dificeis de serem objeto de uma simples cópia para a análise de outros fenômenos, mas o seu entendimento, ou a compreensão

das técnicas utilizadas permitirão com extrema facilidade a construção de algoritmos apropriados à análise desses fenômenos a partir das conclusões aqui apresentadas.

1.2- O Método dos Elementos Finitos e a Análise Matricial de Estruturas

O método dos elementos finitos estabelece uma técnica, extremamente eficiente, para a análise de fenômenos físicos, na sua transmissão através de um meio material, tendo portanto aplicação garantida em fenômenos termodinâmicos, fluidodinâmicos e estruturais, entre outros.

A técnica consiste na discretização do corpo em análise, em elementos finitos, pequenos elementos de geometria simples, tais como retângulos ou triângulos. Feita a discretização, podem ser adotadas funções ditas suaves, que representem por aproximação da função fenômeno físico em análise, a transmissão do dito fenômeno físico de um nó (vértices, centro ou meio dos lados do elemento finito adotado) para outro.

O processo de discretização promove a divisão do corpo material em uma malha formada por uma coleção de elementos finitos, que por adjacência e continuidade compartilham seus nós. Se for feita a soma dos valores que as variáveis de projeto assumem em cada nó, que no caso estrutural são esforços generalizados, tendo em vista a contribuição de cada elemento àquele nó e estendido o procedimento a todos os elementos da corpo em análise, ao final do processo, estará definida em cada nó a quantificação de cada variável do fenômeno físico em questão.

MOREIRA[1], define, com muita propriedade, a Análise Matricial de Estruturas, aplicável às estruturas de barras, como sendo o primeiro capítulo do Método dos Elementos Finitos, uma vez que, trabalhando com funções exatas, pois o fenomeno estrutural opera com funções suaves, constroi a denominada matriz de rigidez local e a partir dela usa o método para a complementação do cálculo. Aqui, a estrutura já está discretizada por natureza, tendo suas barras constituintes como elementos finitos, ou caso seja necessário, barras discretizadas em barras menores.

Como o objetivo deste trabalho tem por base o trato da solução do sistema de equações lineares gerado pela aplicação do método, escolheu-se, tendo em vista a sua simplicidade, a análise de estruturas planas, no regime elástico linear, sob carregamento estático, pelo processo dos deslocamentos, como veículo gerador do problema a ser aqui abordado.

Assim, no caso da análise de estruturas reticuladas planas o método tem por objetivo construir uma matriz-fator com os esforços necessários para que os nós, ou pontos de inicio e/ou de fim das barras, sofram deslocamentos unitários em cada uma das coordenadas globais definidas previamente, de tal forma que os deslocamentos observados devidos a um carregamento atuante, quando multiplicados por essa matriz-fator, reproduzam em cada coordenada as ações que compõem o tal carregamento.

O processo conduzirá, desta forma, à solução de um sistema de equações lineares que pode ser escrito matricialmente como:

$$[K].\{x\} = \{b\} \qquad (1.1.1)$$

onde [K] é a matriz dos coeficientes, à qual se dá o nome de matriz de rigidez global, que é a matriz fator como acima denominamos, $\{x\}$ é o vetor dos deslocamentos a ser calculado e $\{b\}$ é o vetor das ações externas atuantes em cada nó.

1.3-A Matriz de Rigidez e Sua Armazenagem

O processo tem início com a análise de cada barra tomada como um elemento isolado, tendo um sistema local de coordenadas associado ao número de graus de liberdade que se deseja para os nós. Como estão sendo contemplados, neste trabalho, sistemas planos, define-se em cada nó uma coordenada para translação axial, uma coordenada para translação perpendicular ao eixo axial e outra coordenada para medir a rotação. A ilustração das coordenadas locais, bem como sua numeração, com referencia a uma barra, pode ser observada na figura 1.3.1.

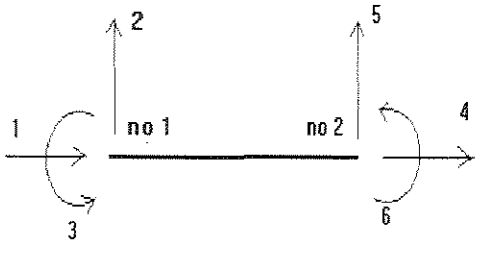

fig. 1.3.1 - SISTEMA LOCAL DE COORDENADAS

Cada barra terá, portanto, para o sistema local definido acima, uma matriz de rigidez, dita local, conforme apresentado na equação 1.3.1:

$$[k] = \begin{bmatrix} EA/L & 0 & 0 & -EA/L & 0 & 0 \\ 0 & 12EI/L^3 & 6EI/L^2 & 0 & -12EI/L^3 & 6EI/L^2 \\ 0 & 6EI/L^2 & 4EI/L & 0 & -6EI/L^2 & 2EI/L \\ -EA/L & 0 & 0 & EA/L & 0 & 0 \\ 0 & -12EI/L^3 & -6EI/L^2 & 0 & 12EI/L^3 & -6EI/L^2 \\ 0 & 6EI/L^2 & 2EI/L & 0 & -6EI/L^2 & 4EI/L \end{bmatrix}$$

(1.3.1)

onde:

E=módulo de elasticidade do material da barra;

A=área de secção transversal da barra;

L=comprimento da barra;

I=momento de inércia da secção transversal da barra em relação ao eixo ortogonal ao plano contido pelos eixos definidos e passante pelo baricentro da dita secção;

A matriz $[k]$, acima indicada, foi obtida impondo a cada nó um deslocamento unitário em cada eixo coordenado, de tal forma que não sejam observados deslocamentos nas demais coordenadas e calculando as ações a serem impostas de forma a garantir o equilíbrio do elemento.

Pode-se facilmente observar, como ilustrado na figura 1.3.2, que a matriz acima se divide em quatro submatrizes, sendo a submatriz superior esquerda dedicada às ações nas coordenadas locais do nó início da barra, devidas a deslocamentos unitários impostos

ao mesmo nó início, em cada uma de suas 3 coordenadas locais. O quadrante inferior direito contém as ações nas coordenadas locais do nó final da barra, devidas a deslocamentos unitários impostos a esse nó em suas 3 coordenadas locais. No quadrante superior direito estão expressas as ações nas 3 coordenadas locais do nó inicial, devidos aos deslocamentos impostos nas coordenadas locais do nó final da barra e a submatriz inferior esquerda contempla as ações nas coordenadas locais do nó final da barra devidas aos deslocamentos unitários impostos ao nó inicial da mesma.

nó início nó início	**nó início nó fim**
nó fim nó início	**nó fim nó fim**

fig 1.3.2

Analisando agora uma simples estrutura constituida por uma viga contínua como a da figura 1.3.3, verifica-se que ela é formada por um conjunto de barras ligadas umas às outras através de seus nós. Adotando uma numeração sequencial para os nós e uma numeração sequencial para as barras, tem-se que o nó final de uma barra é o nó início da sua subsequente, com excessão da última.

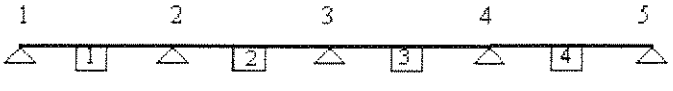

fig. 1.3.3

Dessa forma os coeficientes contidos na submatriz inferior direita da matriz de rigidez da barra 1, são forças aplicadas nas mesmas coordenadas que correspondem aos coeficientes do quadrante superior esquerdo da matriz de rigidez da barra 2, e devem portanto ser somados. Assim a matriz de rigidez global pode ser montada por superposição de forças aplicadas nas mesmas coordenadas. Para a viga do exemplo, admitindo os nós com a numeração seqüencial adotada, a matriz de rigidez global terá o aspecto mostrado na figura 1.3.4.

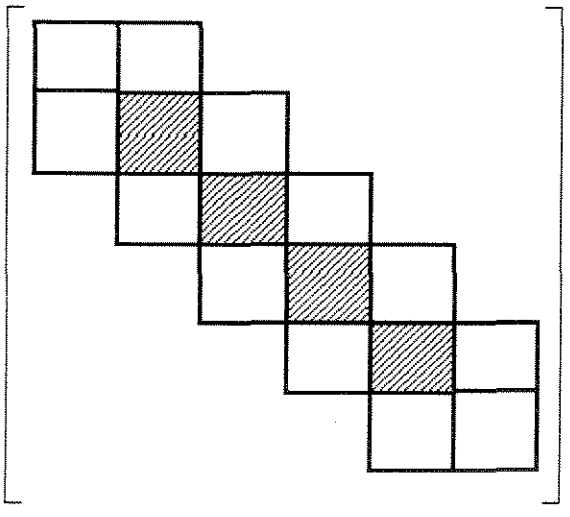

fig. 1.3.4 - ASPECTO ESQUEMÁTICO DA MATRIZ DE RIGIDEZ GLOBAL DA VIGA DA FIG. 1.3.3

Continuando a análise do caso estrutural plano e reticulado, podemos verificar que a matriz de rigidez global terá uma dimensão de três vezes o número de nós por três vezes o número de nós,ou seja, se está sendo tratada uma estrutura com 50 nós, terá que ser armazenada uma matriz de rigidez de 150 linhas por 150 colunas, o que torna

obrigatória a reserva de 22500 posições de variáveis do tipo real na memória do equipamento. O problema surge quando se deseja utilizar microcomputadores do tipo PC, gerenciados por sistema DOS, utilizando-se compiladores como o Pascal, C ou C++, para a análise deste tipo de problema. Conforme apresentado por Roth[2], a arquitetura das máquinas das séries 80*86 acessam a memória em segmentos de 64 Kbytes e os compiladores, C, C++, Pascal, incluindo o Borland Pascal 7.0, reservam apenas um segmento de 64 Kbytes, para ser utilizado como área de dados acessada diretamente pela linguagem. Acrescido o fato de que cada variável do tipo real necessita 6 bytes para seu armazenamento, no caso da utilização do Borland Pascal, verifica-se que se estivesse disponível esse segmento apenas para a matriz de rigidez global, o equipamento estaria limitado a estruturas de 34 nós e se for aventado ainda o fato de que são necessários espaços de memória para os outros dados e resultados da análise, e que, a bem da precisão dos resultados, o ideal seria se trabalhar com variáveis com dupla precisão, que ocupam 8 bytes, chegar-se-á à conclusão que a análise ficaria restrita a estruturas ainda menores, o que tornaria os microcomputadores do tipo PC impróprios para esse tipo de utilização.

Nos capítulos posteriores estarão sendo analisados os artifícios existentes para contornar esse problema.

2-METODOLOGIA

Tendo em vista o objetivo do trabalho descrito no capítulo anterior, optou-se pela construção de um programa básico capaz de calcular estruturas reticuladas planas, mesmo que restrito a pequenas estruturas, mas que fosse capaz de permitir a construção de outras versões e assim se poder avaliar o desempenho de cada uma em relação ao tempo de processamento e ao tamanho de estrutura que cada versão seja capaz de analisar, utilizando para validação dos mesmos o programa SAP90-Structural Analysis Program, Wilson[5].

Para mensuração de resultados foram formuladas três estruturas com características diferentes, que permitissem a qualificação de cada versão, bem como a comparação dos resultados de performance entre elas.

Como equipamentos foram utilizados um microcomputador tipo PC modelo 486 DX de 33 Mhz e uma estação RISC 6000, modelo 320H da IBM.

2.1-Primeiro Exemplo Para Análise

Como primeira estrutura, utilizou-se um pequeno pórtico plano formado por um pilar e uma viga, que foi discretizado em quatro barras, gerando assim cinco nós, conforme apresentado na figura 2.1.1.

2.2-Segundo Exemplo Para Análise

Como segunda estrutura foi escolhido um pórtico de porte médio, tipicamente utilizado na indústria, contendo 69 barras conectadas a 38 nós, conforme podemos observar na figura 2.2.1.

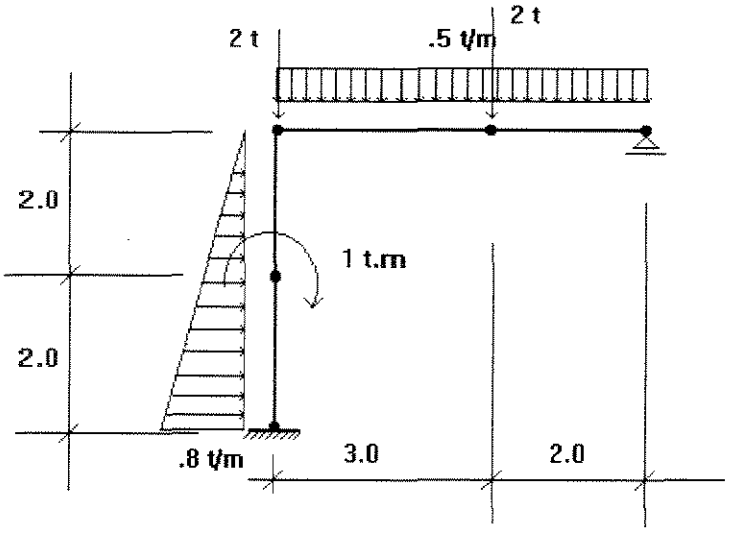

fig 2.1.1

medidas em metros

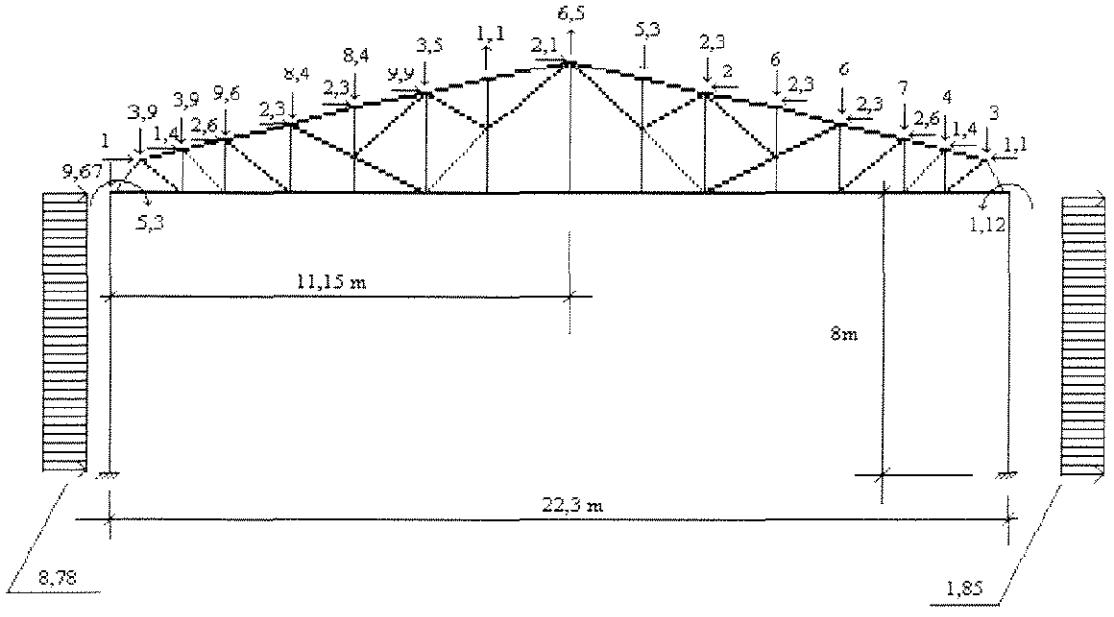

fig. 2.2.1

2.3-Terceiro Exemplo Para Análise

Como estrutura grande tomou-se a estrutura da figura 2.3.1, agora com 395 barras conectadas por 160 nós. Trata-se de uma estrutura especialmente formulada para não apresentar a formação de banda na matriz de rigidez global, fenomeno normalmente observado como se verá mais adiante.

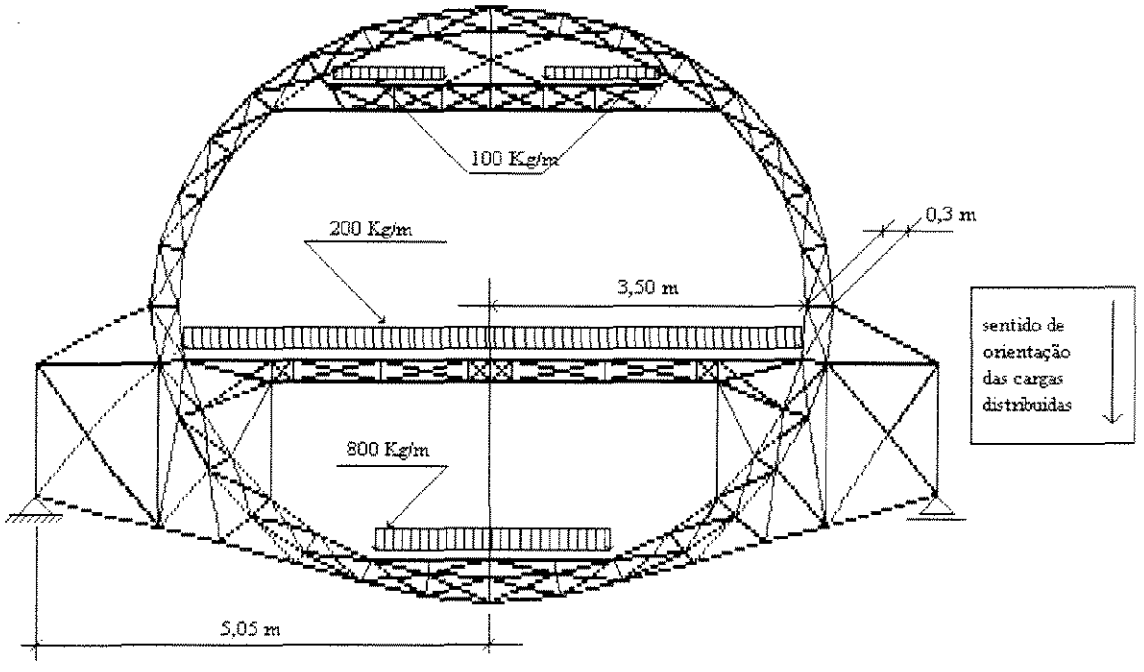

fig 2.3.1

3-APRESENTAÇÃO DO PROGRAMA BÁSICO

O programa básico foi desenvolvido em Borland-Pascal na sua versão 7.0 e parte da concepção de dois tipos básicos de registros, um para acomodar os dados de um nó e outro para acomodar os dados de uma barra. O registro (record) é um recurso existente tanto no Pascal como no C e C++, que permite a construção de vetores de conjuntos de mais de um tipo de dados. Por exemplo, para a montagem do arquivo dos alunos de uma escola, criar-se-ia um registro para cada aluno que abrigaria um conjunto de informações comuns a todos os alunos tais como: nome, filiação, endereço, etc. Dessa forma, ao se criar um registro e um vetor deles, estaria sendo formado o arquivo de alunos desejado. Assim, voltando ao caso do programa básico,foi dada a denominação de NOREC, ao registro que armazena os dados de um nó, o qual contém:

-número do nó;

-coordenadas (abscissas e ordenadas);

-grau de restrição a deslocamentos;

-carregamentos nodais (1-paralelo ao eixo das abscissas, 2-paralelo ao eixo das ordenadas,3-momento no plano que abriga o sistema em análise);

-deslocamentos(nas tres coordenadas globais citadas no ítem anterior);

-esforços reativos em cada um dos eixos coordenados globais;.

Os registros que armazenam os dados de uma barra foram denominados "BARREC" e contém:

-número da barra;

-nós de início e fim da barra;

-momento de inércia;

-módulo de elasticidade;

-comprimento da barra;

-área de secção transversal;

-cosenos diretores;

-carregamentos nas barras(cargas concentradas, uniformemente distribuidas, uniformemente variadas, axiais concentradas, axiais uniformemente distribuidas, momentos aplicados);

-ações nodais equivalentes(no sistema local, fig 1.2.1);

A exemplo do programa SAP90[5] , o modelo elaborado exige a criação por parte do usuário de um arquivo de entrada, no qual devem ser fornecidos dentro de uma ordem e forma pré-definida os dados que definem toda a geometria da estrutura a ser analisada, bem como o carregamento a que ela está submetida.

Alocado o espaço na área de dados para os vetores de registros, que armazenarão os dados de um problema estrutural em análise, o programa é iniciado com uma parte interativa via tela, na qual, primeiramente, ele pede ao usuário que identifique o arquivo de entrada e após sua busca, numa segunda chamada, que seja identificado o arquivo de saída, aonde serão gravados os resultados da análise em curso. Na sequência ele faz a leitura das coordenadas dos nós e seus graus de liberdade.

3.1-Obtenção da Matriz de Rigidez Global

Como já foi dito anteriormente, o programa básico tem por fim produzir um algoritmo a partir do qual sejam construídas as diversas versões. Nele armazena-se a matriz plena e sem reordenação de índices.

Após os procedimentos inciais já descritos, o programa entra num laço que se inicia na barra de número 1 e termina na última, no qual são feitas as seguintes operações em cada barra:

- leitura das características geométricas e elásticas das barras, tais como, número da barra, número do nó de início, nó final, módulo de elasticidade, momento de inércia e área de secção transversal .

- calcula o comprimento de cada barra a partir das coordenadas dos nós de início e fim :

$$C = \sqrt{r^2 + d^2} \quad (3.1.1) \quad \text{onde:}$$

C= comprimento da barra;

r=diferença de ordenadas dos nós de início e fim;

d=diferença de abscissas dos nós de início e fim;

- calcula os cosenos diretores:

$$\cos x = \frac{d}{C} \; ; \quad \cos y = \frac{r}{C} \quad (3.1.2)$$

- calcula a matriz de rigidez da barra conforme (1.3.1) e provoca sua transferência do sistema local para o global através da seguinte expressão matricial:

$$[k_G] = [\beta]^t . [k] . [\beta] \quad (3.1.3)$$

onde:

$[k_G]$ = matriz de rigidez da barra referida ao sistema global de coordenadas;

$[k]$ =matriz de rigidez local da barra, calculado como em (1.3.1);

$[\beta]$=matriz de incidência cinemática, que é aquela capaz de exprimir os deslocamentos nas coordenadas locais a partir de seus valores expressos nas coordenadas globais e pode ser obtida, para uma barra, como indicado na figura 3.1.1:

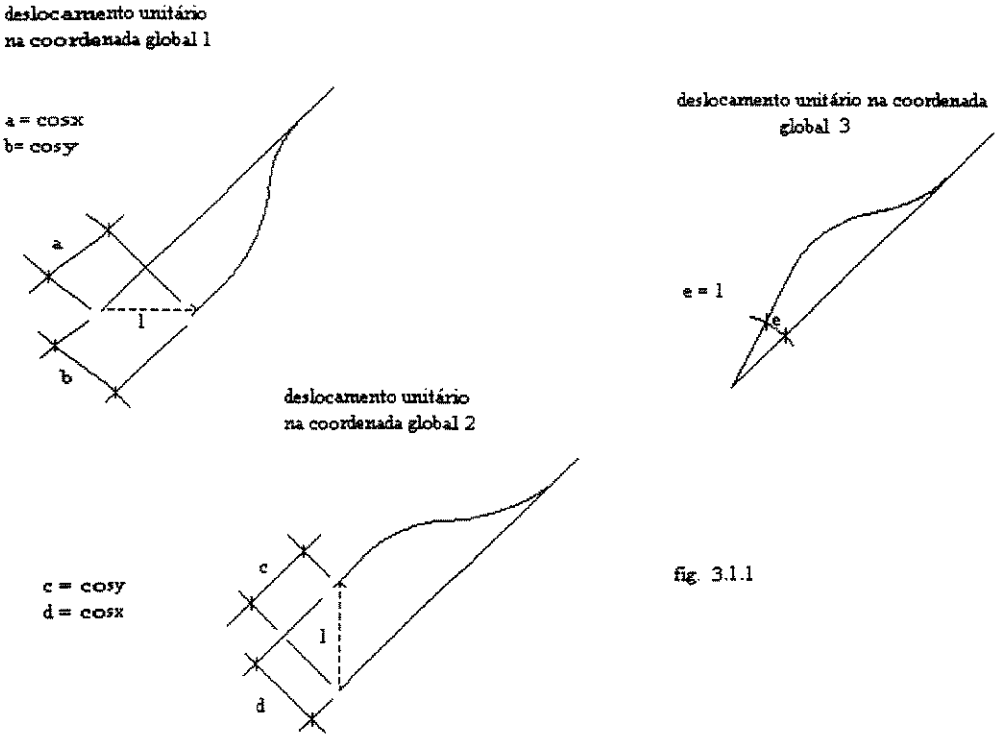

fig. 3.1.1

da figura 3.1.1 podemos ter:

$$\beta_{11} = \cos x; \quad \beta_{21} = -\cos y; \quad \beta_{33} = 1$$

$$\beta_{12} = \cos y; \quad \beta_{22} = \cos x;$$

(3.1.4)

desta forma podemos construir a matriz $[\beta]$ de cada barra como sendo:

$$[\beta] = \begin{bmatrix} \cos x & \cos y & 0 & 0 & 0 & 0 \\ -\cos y & \cos x & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos x & \cos y & 0 \\ 0 & 0 & 0 & -\cos y & \cos x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.1.5)$$

- processa a sobreposição com soma dos elementos da matriz de rigidez da barra aos elementos da matriz de rigidez global, endereçando a submatriz superior esquerda às posições correspondentes ao número do nó de início da barra processada e por analogia os demais quadrantes, montando assim a matriz de rigidez global. Para melhor esclarecer, supõe-se que esteja sendo processada uma barra que tenha por nó inicial o nó 1 e por nó final o nó 5 e que essa seja a primeira barra processada pelo programa. As contribuições que os elementos da matriz de rigidez local terão na matriz global, esquematicamente, serão como na figura abaixo:

	1	2	3	4	....	12	13	14	15	...	n
1	x	x	x				x	x	x		
2	x	x	x				x	x	x		
3	x	x	x				x	x	x		
4											
..											
12											
13	x	x	x				x	x	x		
14	x	x	x				x	x	x		
15	x	x	x				x	x	x		
....											
n											

fig. 3.1.2

3.2-A Obtenção do Vetor de Ações

Na seqüência o programa faz as leituras das cargas aplicadas em cada barra e calcula as suas ações de engastamento perfeito, armazenando-as nos campos dos registros das barras. Faz também a leitura dos dados relativos aos carregamentos nodais e preenche com eles os campos dos registros dos nós a eles reservados.

Em seguida, faz uma varredura de todas as barras, produzindo a transposição dos esforços equilibrantes de cada barra do sistema de coordenadas local para o global, mudando seu sinal para que se tenham os esforços nodais equivalentes ao carregamento das barras. Com os esforços nodais equivalentes assim calculados e armazenados nos registros das barras, procede-se a uma nova varredura das barras, procurando os esforços nodais armazenados e somando-os aos valores de carregamento nodal armazenados nos registros dos nós. Em seguida são armazenados num vetor, denominado de vetor das ações.

3.3-As Condições de Contorno

O sistema de equações lineares construído é indefinido e isso é perfeitamente compatível com a realidade física, uma vez que a falta de aparelhos de apoio, ainda não comentados aqui, que darão as chamadas condições de contorno à estrutura, farão com que ela não tenha equilíbrio definido. Os aparelhos de apoio, na verdade, impõem deslocamentos nulos nas coordenadas a eles relacionadas, o que pode ser simulado anulando-se os coeficientes expressos nas linhas e colunas definidas por essas coordenadas impedidas, o que é feito pelo algoritmo dentro do laço descrito em 3.1.

Agora a matriz dos coeficientes ficou singular tendo em vista a posição nula obtida na diagonal. O programa contorna esta questão dentro do mesmo laço, colocando 1 na posição nula da diagonal e depois das operações descritas em 3.2 anula a posição correspondente do vetor das ações (membro direito da equação1.1.1).

3.4-O Algoritmo Gaussiano

Uma vez conhecida a matriz [K] e o vetor das ações, já devidamente modificados pelas condições de contorno, o programa está apto a calcular os deslocamentos, bastando apenas resolver o sistema (1.1.1) já conhecido. Para tanto, ele se vale do processo de eliminação de Gauss.

3.4.1- O Método de Eliminação de Gauss

Para se poder mostrar o método da eliminação de Gauss, propõe-se um sistema de cinco equações a cinco incognitas, sem que com isso se perca a generalidade, como indicado a seguir:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + a_{15}x_5 &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + a_{25}x_5 &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 + a_{35}x_5 &= b_3 \\
a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + a_{45}x_5 &= b_4 \\
a_{51}x_1 + a_{52}x_2 + a_{53}x_3 + a_{54}x_4 + a_{55}x_5 &= b_5
\end{aligned} \qquad (3.4.1.1)$$

Multiplicando-se a primeira equação de (3.4.1.1) por:

$$-\frac{a_{21}}{a_{11}} \qquad (3.4.1.2)$$

obter-se-á:

$$-a_{21}x_1 - \frac{a_{12}.a_{21}}{a_{11}}x_2 - \frac{a_{13}.a_{21}}{a_{11}}x_3 - \frac{a_{14}.a_{21}}{a_{11}}x_4 - \frac{a_{15}.a_{21}}{a_{11}}x_5 = -\frac{b_1.a_{21}}{a_{11}} \qquad (3.4.1.3)$$

e se na sequência for somada a nova equação com a segunda equação do sistema, será obtida:

$$\left(a_{22} - \frac{a_{12}.a_{21}}{a_{11}}\right)x_2 + \left(a_{23} - \frac{a_{13}.a_{21}}{a_{11}}\right)x_3 + \left(a_{24} - \frac{a_{14}.a_{21}}{a_{11}}\right)x_4 + \left(a_{25} - \frac{a_{15}.a_{21}}{a_{11}}\right)x_5 = b_2 - \frac{b_1.a_{21}}{a_{11}}$$

(3.4.1.4)

Como se pode verificar foi eliminada a incógnita x_1. Analogamente multiplicando-se a primeira equação por:

$$-\frac{a_{31}}{a_{11}} \qquad (3.4.1.5)$$

e somando-a à terceira equação, será obtida uma nova equação como abaixo:

$$\left(a_{32} - \frac{a_{12}.a_{31}}{a_{11}}\right)x_2 + \left(a_{33} - \frac{a_{13}.a_{31}}{a_{11}}\right)x_3 + \left(a_{34} - \frac{a_{14}.a_{31}}{a_{11}}\right)x_4 + \left(a_{35} - \frac{a_{15}.a_{31}}{a_{11}}\right)x_5 = b_3 - \frac{b_1.a_{31}}{a_{11}}$$

(3.4.1.6)

sem a incógnita x_1. Por analogia se chega a um sistema de quatro equações a quatro incógnitas, cujas duas últimas equações serão:

$$\left(a_{42} - \frac{a_{12}.a_{41}}{a_{11}}\right)x_2 + \left(a_{43} - \frac{a_{13}.a_{41}}{a_{11}}\right)x_3 + \left(a_{44} - \frac{a_{14}.a_{41}}{a_{11}}\right)x_4 + \left(a_{45} - \frac{a_{15}.a_{41}}{a_{11}}\right)x_5 = b_4 - \frac{b_1.a_{41}}{a_{11}}$$

(3.4.1.7)

$$\left(a_{52}-\frac{a_{12}.a_{51}}{a_{11}}\right)x_2+\left(a_{53}-\frac{a_{13}.a_{51}}{a_{11}}\right)x_3+\left(a_{54}-\frac{a_{14}.a_{51}}{a_{11}}\right)x_4+\left(a_{55}-\frac{a_{15}.a_{51}}{a_{11}}\right)x_5=$$
$$b_5-\frac{b_1.a_{51}}{a_{11}}$$

(3.4.1.8)

se for chamado o primeiro coeficiente da primeira equação do novo sistema de quatro equações a quatro incógnitas de a_{22}^1, ou seja:

$$a_{22}^1=a_{22}-\frac{a_{12}.a_{21}}{a_{11}} \qquad (3.4.1.9)$$

e o segundo da primeira equação de a_{23}^1 e assim por diante, poderá ser reescrito o sistema de quatro equações a quatro incógnitas, como em (3.4.1.10).

$$\begin{aligned}
a_{22}^1.x_2+a_{23}^1.x_3+a_{24}^1.x_4+a_{25}^1.x_5&=b_2^1\\
a_{32}^1.x_2+a_{33}^1.x_3+a_{34}^1.x_4+a_{35}^1.x_5&=b_3^1\\
a_{42}^1.x_2+a_{43}^1.x_3+a_{44}^1.x_4+a_{45}^1.x_5&=b_4^1\\
a_{52}^1.x_2+a_{53}^1.x_3+a_{54}^1.x_4+a_{55}^1.x_5&=b_5^1
\end{aligned} \qquad (3.4.1.10)$$

O passo seguinte é aplicar novamente o processo sobre o sistema (3.4.1.10), ou seja, multiplicar a primeira equação por $-\frac{a_{32}^1}{a_{22}^1}$ e somar com a segunda, repetindo sucessivamente o processo já descrito, por todo o sistema, o que levará a um novo sistema de três equações a três incógnitas apresentado a seguir, em (3.4.1.11):

$$\left(a_{33}^1-\frac{a_{23}^1.a_{32}^1}{a_{22}^1}\right)x_3+\left(a_{34}^1-\frac{a_{24}^1.a_{32}^1}{a_{22}^1}\right)x_4+\left(a_{35}^1-\frac{a_{25}^1.a_{32}^1}{a_{22}^1}\right)x_5=b_3^1-\frac{b_2^1.a_{32}^1}{a_{22}^1}$$

$$\left(a_{43}^{1}-\frac{a_{33}^{1}.a_{42}^{1}}{a_{32}^{1}}\right)x_3+\left(a_{44}^{1}-\frac{a_{34}^{1}.a_{42}^{1}}{a_{32}^{1}}\right)x_4+\left(a_{45}^{1}-\frac{a_{35}^{1}.a_{42}^{1}}{a_{32}^{1}}\right)x_5=b_4^1-\frac{b_3^1.a_{42}^{1}}{a_{32}^{1}}$$

$$\left(a_{53}^{1}-\frac{a_{43}^{1}.a_{52}^{1}}{a_{42}^{1}}\right)x_3+\left(a_{54}^{1}-\frac{a_{44}^{1}.a_{52}^{1}}{a_{42}^{1}}\right)x_4+\left(a_{55}^{1}-\frac{a_{45}^{1}.a_{52}^{1}}{a_{42}^{1}}\right)x_5=b_5^1-\frac{b_4^1.a_{52}^{1}}{a_{42}^{1}}$$

(3.4.1.11)

Repetindo a estratégia anterior pode-se escrever o sistema (3.4.1.11), como em (3.4.1.12) abaixo:

$$\begin{aligned}
a_{33}^{2}.x_3+a_{34}^{2}.x_4+a_{35}^{2}.x_5&=b_3^2\\
a_{43}^{2}.x_3+a_{44}^{2}.x_4+a_{45}^{2}.x_5&=b_4^2\\
a_{53}^{2}.x_3+a_{54}^{2}.x_4+a_{55}^{2}.x_5&=b_5^2
\end{aligned}\qquad (3.4.1.12)$$

Aplicando novamente a mesma técnica chega-se ao seguinte sistema de duas equações a duas incógnitas:

$$\left(a_{44}^{2}-\frac{a_{34}^{2}.a_{43}^{2}}{a_{33}^{2}}\right)x_4+\left(a_{45}^{2}-\frac{a_{35}^{2}.a_{43}^{2}}{a_{33}^{2}}\right)x_5=b_4^2-\frac{b_3^2.a_{43}^{2}}{a_{33}^{2}}$$

$$\left(a_{54}^{2}-\frac{a_{44}^{2}.a_{53}^{2}}{a_{43}^{2}}\right)x_4+\left(a_{55}^{2}-\frac{a_{45}^{2}.a_{53}^{2}}{a_{43}^{2}}\right)x_5=b_5^2-\frac{b_4^2.a_{53}^{2}}{a_{43}^{2}}$$

(3.4.1.13)

que à exemplo das vezes anteriores, pode assim ser escrito:

$$\begin{aligned}
a_{44}^{3}.x_4+a_{45}^{3}.x_5&=b_4^3\\
a_{54}^{3}.x_4+a_{55}^{3}.x_5&=b_5^3
\end{aligned}\qquad (3.4.1.14)$$

finalmente, se for multiplicada a primeira equação do sistema (3.4.1.14) por $-\frac{a_{54}^{3}}{a_{44}^{3}}$ e somada com a segunda obter-se-á a equação (3.4.1.15).

$$\left(a_{55}^{3}-\frac{a_{45}^{3}.a_{54}^{3}}{a_{44}^{3}}\right)x_5=b_5^3-\frac{b_4^3.a_{54}^{3}}{a_{44}^{3}}\qquad (3.4.1.15)$$

o que vai permitir encontrar o valor de x_5 . Encontrado x_5, volta-se às equações anteriores, obtendo-se sucessivamente, x_4, x_3, x_2 e x_1. É claro que o que foi mostrado acima, no sistema de cinco equações a cinco incógnitas , pode ser generalizado para sistemas de qualquer número de equações, uma vez que o sistema de eliminação apresentado pode ser repetido indefinidamente.

3.4.2-A Fatorização L U

A eliminação de Gauss descrita na seção acima permite, como se pode ver em Golub [3] e em Duff et. al [4] , que se faça a fatorização da matriz dos coeficientes [K](1.1.1) em duas matrizes triangulares, uma denominada de [L] e outra de [U]. A primeira é triangular inferior e a outra triangular superior, de tal forma que a matriz chamada de [U], contenha os coeficientes resultantes das diversas eliminações do sistema gaussiano.

Escrevendo o sistema de equações em forma matricial tem-se:

$$[A].\{x\}=\{b\} \quad (3.4.2.1)$$

onde:

[A] = matriz dos coeficientes;

{b} = vetor das constantes conhecidas;

{x} = vetor das incógnitas;

Mais uma vez, sem perda da generalidade, tome-se um sistema 5×5 tendo em vista um melhor entendimento:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \times \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{vmatrix} = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{vmatrix} \qquad (3.4.2.2)$$

A fatorização da matriz de coeficientes como descrita acima, pode ser feita conforme apresentada em (3.4.2.3).

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 1 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 1 \end{bmatrix} \times \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{bmatrix}$$

(3.4.2.3)

Duff et al, op. cit p. 46, propõe que se construa uma matriz [L], fazendo para $i \rangle k$:

$$l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \qquad (3.4.2.4)$$

sendo essa uma relação perfeitamente válida, uma vez que para problemas estruturais elástico lineares a matriz dos coeficientes é sempre positiva definida e portanto $a_{kk}^{(k)} \neq 0$.

Pensando nessa propriedade (3.4.2.4), constrói-se agora a família de matrizes L de (3.4.2.3).

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -l_{21} & 1 & 0 & 0 & 0 \\ -l_{31} & 0 & 1 & 0 & 0 \\ -l_{41} & 0 & 0 & 1 & 0 \\ -l_{51} & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -l_{32} & 1 & 0 & 0 \\ 0 & -l_{42} & 0 & 1 & 0 \\ 0 & -l_{52} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -l_{43} & 1 & 0 \\ 0 & 0 & -l_{53} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -l_{54} & 1 \end{bmatrix}$$

(3.4.2.5)

Observando atentamente o conjunto de quatro matrizes de (3.4.2.5), pode-se verificar que o produto delas conduz à (3.4.2.6).

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -l_{21} & 1 & 0 & 0 & 0 \\ -l_{31} & -l_{32} & 1 & 0 & 0 \\ -l_{41} & -l_{42} & -l_{43} & 1 & 0 \\ -l_{51} & -l_{52} & -l_{53} & -l_{54} & 1 \end{bmatrix} \qquad (3.4.2.6)$$

Sabe-se também que :

$$\text{se } \left[L^{(2)}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & l_{32} & 1 & 0 & 0 \\ 0 & l_{42} & 0 & 1 & 0 \\ 0 & l_{52} & 0 & 0 & 1 \end{bmatrix} \text{,então } \left[L^{(2)}\right]^{(-1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -l_{32} & 1 & 0 & 0 \\ 0 & -l_{42} & 0 & 1 & 0 \\ 0 & -l_{52} & 0 & 0 & 1 \end{bmatrix} \qquad (3.4.2.7)$$

pois:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -l_{32} & 1 & 0 & 0 \\ 0 & -l_{42} & 0 & 1 & 0 \\ 0 & -l_{52} & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & l_{32} & 1 & 0 & 0 \\ 0 & l_{42} & 0 & 1 & 0 \\ 0 & l_{52} & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.4.2.8)$$

Pode-se então, escrever que :

$$\left[L^{(1)}\right]^{(-1)}.\left[L^{(2)}\right]^{(-1)}.\left[L^{(3)}\right]^{(-1)}.\left[L^{(4)}\right]^{(-1)}=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ l_{21} & 1 & 0 & 0 & 0\\ l_{31} & l_{32} & 1 & 0 & 0\\ l_{41} & l_{42} & l_{43} & 1 & 0\\ l_{51} & l_{52} & l_{53} & l_{54} & 1\end{bmatrix} \quad (3.4.2.9)$$

que é a matriz L de (3.4.2.3). Generalizando, pode-se escrever que se a matriz $\left[L^{(k)}\right]$, de (3.4.2.10) é o k-ésimo fator de um produto que leva à matriz L de (3.4.2.13)

$$\begin{bmatrix}1 & 0 & . & . & 0 & 0 & 0 & . & . & 0 & 0\\ 0 & 1 & . & . & 0 & 0 & 0 & . & . & 0 & 0\\ 0 & 0 & . & . & 0 & 0 & 0 & . & . & 0 & 0\\ . & . & . & . & . & . & . & . & . & . & .\\ . & . & . & . & . & . & . & . & . & . & .\\ 0 & 0 & . & . & 1 & 0 & 0 & . & . & 0 & 0\\ 0 & 0 & . & . & 0 & 1 & 0 & . & . & 0 & 0\\ 0 & 0 & . & . & 0 & -l_{k+1,k} & 1 & . & . & 0 & 0\\ 0 & 0 & . & . & 0 & -l_{k+2,k} & 0 & . & . & 0 & 0\\ . & . & . & . & . & . & . & . & . & . & .\\ 0 & 0 & . & . & 0 & -l_{n,k} & 0 & . & . & 0 & 1\end{bmatrix}$$

(3.4.2.10)

então, a relação :

$$\left[L^{(1)}\right]^{(-1)}.\left[L^{(2)}\right]^{(-1)}.....\left[L^{(n-1)}\right]^{(-1)}=[L] \quad (3.4.2.11),$$

é válida e a equação (3.4.2.12) conduz à matriz U desejada, por conter todas as linhas obtidas da eliminação gaussiana.

$$[L]^{(-1)}\times[K]=[U]$$

(3.4.2.12)

$$\begin{bmatrix} 1 & 0 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & 0 \\ -l_{2,1} & 1 & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & 0 \\ -l_{3,1} & -l_{3,2} & \cdot\cdot & 0 & 0 & 0 & \cdot\cdot & 0 & 0 \\ \cdot & \cdot & \cdot\cdot & \cdot & \cdot & \cdot & \cdot\cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot\cdot & \cdot & \cdot & \cdot & \cdot\cdot & \cdot & \cdot \\ -l_{k-1,1} & -l_{k-1,2} & \cdot\cdot & 1 & 0 & 0 & \cdot\cdot & 0 & 0 \\ -l_{k,1} & -l_{k,2} & \cdot\cdot & -l_{k,k-1} & 1 & 0 & \cdot\cdot & 0 & 0 \\ -l_{k+1,1} & -l_{k+1,2} & \cdot\cdot & -l_{k+1,k-1} & -l_{k+1,k} & 1 & \cdot\cdot & 0 & 0 \\ -l_{k+2,1} & -l_{k+2,2} & \cdot\cdot & -l_{k+2,k-1} & -l_{k+2,k} & -l_{k+2,k+1} & \cdot\cdot & 0 & 0 \\ \cdot & \cdot & \cdot\cdot & \cdot & \cdot & \cdot & \cdot\cdot & \cdot & \cdot \\ -l_{n,1} & -l_{n,2} & \cdot\cdot & -l_{n,k-1} & -l_{n,k} & -l_{n,k+1} & \cdot\cdot & -l_{n,n-1} & 1 \end{bmatrix} \quad (3.4.2.13)$$

Para compreender melhor a formulação, volte-se àquele sistema 5×5 de (3.4.1.1) e da sua forma matricial como em (3.4.2.2) e tome-se dela a matriz dos coeficientes.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \quad (3.4.2.14)$$

Construa-se agora, conforme proposto anteriormente, a matriz $[L^{(1)}]$,

$$[L^{(1)}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & 0 & 0 & 0 \\ -\frac{a_{31}}{a_{11}} & 0 & 1 & 0 & 0 \\ -\frac{a_{41}}{a_{11}} & 0 & 0 & 1 & 0 \\ -\frac{a_{51}}{a_{11}} & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.2.15)$$

Seja feita a sua multiplicação pela matriz [K] e se encontrará (3.4.2.16).

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}-\frac{a_{12}.a_{21}}{a_{11}} & a_{23}-\frac{a_{13}.a_{21}}{a_{11}} & a_{24}-\frac{a_{14}.a_{21}}{a_{11}} & a_{25}-\frac{a_{15}.a_{21}}{a_{11}} \\ 0 & a_{32}-\frac{a_{12}.a_{31}}{a_{11}} & a_{33}-\frac{a_{13}.a_{31}}{a_{11}} & a_{34}-\frac{a_{14}.a_{31}}{a_{11}} & a_{35}-\frac{a_{15}.a_{31}}{a_{11}} \\ 0 & a_{42}-\frac{a_{12}.a_{41}}{a_{11}} & a_{43}-\frac{a_{13}.a_{41}}{a_{11}} & a_{44}-\frac{a_{14}.a_{41}}{a_{11}} & a_{45}-\frac{a_{15}.a_{41}}{a_{11}} \\ 0 & a_{52}-\frac{a_{12}.a_{51}}{a_{11}} & a_{53}-\frac{a_{13}.a_{51}}{a_{11}} & a_{54}-\frac{a_{14}.a_{51}}{a_{11}} & a_{55}-\frac{a_{15}.a_{51}}{a_{11}} \end{bmatrix} \quad (3.4.2.16)$$

pode-se perceber claramente que a segunda linha é igual ao membro direito de (3.4.1.4), a terceira linha ao membro direito de (3.4.1.6), a quarta e a quinta aos membros direitos de (3.4.1.7) e (3.4.1.8) respectivamente, o que permite reescrever (3.4.2.16), como (3.4.2.17).

$$[U^{(1)}]=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a^1_{22} & a^1_{23} & a^1_{24} & a^1_{25} \\ 0 & a^1_{32} & a^1_{33} & a^1_{34} & a^1_{35} \\ 0 & a^1_{42} & a^1_{43} & a^1_{44} & a^1_{45} \\ 0 & a^1_{52} & a^1_{53} & a^1_{54} & a^1_{55} \end{bmatrix} \quad (3.4.2.17)$$

Na sequência pode-se efetuar o segundo produto, conforme indicado em (3.4.2.18).

$$[U^{(2)}]=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{a^1_{32}}{a^1_{22}} & 1 & 0 & 0 \\ 0 & -\frac{a^1_{42}}{a^1_{22}} & 0 & 1 & 0 \\ 0 & -\frac{a^1_{52}}{a^1_{22}} & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a^1_{22} & a^1_{23} & a^1_{24} & a^1_{25} \\ 0 & a^1_{32} & a^1_{33} & a^1_{34} & a^1_{35} \\ 0 & a^1_{42} & a^1_{43} & a^1_{44} & a^1_{45} \\ 0 & a^1_{52} & a^1_{53} & a^1_{54} & a^1_{55} \end{bmatrix} \quad (3.4.2.18)$$

e ter-se-á como resultado (3.4.2.19).

$$[U^{(2)}]=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} & a_{25}^{1} \\ 0 & 0 & a_{33}^{1}-\frac{a_{23}^{1}.a_{32}^{1}}{a_{22}^{1}} & a_{34}^{1}-\frac{a_{24}^{1}.a_{32}^{1}}{a_{22}^{1}} & a_{35}^{1}-\frac{a_{25}^{1}.a_{32}^{1}}{a_{22}^{1}} \\ 0 & 0 & a_{43}^{1}-\frac{a_{23}^{1}.a_{42}^{1}}{a_{22}^{1}} & a_{44}^{1}-\frac{a_{24}^{1}.a_{42}^{1}}{a_{22}^{1}} & a_{45}^{1}-\frac{a_{25}^{1}.a_{42}^{1}}{a_{22}^{1}} \\ 0 & 0 & a_{53}^{1}-\frac{a_{23}^{1}.a_{52}^{1}}{a_{22}^{1}} & a_{54}^{1}-\frac{a_{24}^{1}.a_{52}^{1}}{a_{22}^{1}} & a_{55}^{1}-\frac{a_{25}^{1}.a_{52}^{1}}{a_{22}^{1}} \end{bmatrix} \quad (3.4.2.19)$$

pode-se verificar que as três últimas linhas de (3.4.2.19) têm coeficientes idênticos aos do sistema (3.4.1.11), o que permite reescrever aquela primeira como em (3.4.2.20).

$$[U^{(2)}]=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} & a_{25}^{1} \\ 0 & 0 & a_{33}^{2} & a_{34}^{2} & a_{35}^{2} \\ 0 & 0 & a_{43}^{2} & a_{44}^{2} & a_{45}^{2} \\ 0 & 0 & a_{53}^{2} & a_{54}^{2} & a_{55}^{2} \end{bmatrix} \quad (3.4.2.20)$$

Por decorrência procede-se à operação (3.4.2.21),

$$[U^{(3)}]=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{a_{43}^{2}}{a_{33}^{2}} & 1 & 0 \\ 0 & 0 & -\frac{a_{53}^{2}}{a_{33}^{2}} & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} & a_{25}^{1} \\ 0 & 0 & a_{33}^{2} & a_{34}^{2} & a_{35}^{2} \\ 0 & 0 & a_{43}^{2} & a_{44}^{2} & a_{45}^{2} \\ 0 & 0 & a_{53}^{2} & a_{54}^{2} & a_{55}^{2} \end{bmatrix} \quad (3.4.2.21)$$

que dará como resposta (3.4.2.22).

$$[U^{(3)}]=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} & a_{25}^{1} \\ 0 & 0 & a_{33}^{2} & a_{34}^{2} & a_{35}^{2} \\ 0 & 0 & 0 & a_{44}^{2}-\frac{a_{34}^{2}.a_{43}^{2}}{a_{33}^{2}} & a_{45}^{2}-\frac{a_{35}^{2}.a_{43}^{2}}{a_{33}^{2}} \\ 0 & 0 & 0 & a_{54}^{2}-\frac{a_{34}^{2}.a_{43}^{2}}{a_{33}^{2}} & a_{55}^{2}-\frac{a_{35}^{2}.a_{53}^{2}}{a_{33}^{2}} \end{bmatrix} \quad (3.4.2.22)$$

Identifica-se mais uma vez as duas últimas linhas de (3.4.2.22) com os coeficientes do sistema (3.4.1.13), o que permite reescrever a primeira como em (3.4.2.23).

$$[U^{(3)}]=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} & a_{25}^{1} \\ 0 & 0 & a_{33}^{2} & a_{34}^{2} & a_{35}^{2} \\ 0 & 0 & 0 & a_{44}^{3} & a_{45}^{3} \\ 0 & 0 & 0 & a_{54}^{3} & a_{55}^{3} \end{bmatrix} \quad (3.4.2.23)$$

Por analogia pode-se escrever (3.4.2.24).

$$[U]=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\dfrac{a_{54}^{3}}{a_{44}^{3}} & 1 \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} & a_{25}^{1} \\ 0 & 0 & a_{33}^{2} & a_{34}^{2} & a_{35}^{2} \\ 0 & 0 & 0 & a_{44}^{3} & a_{45}^{3} \\ 0 & 0 & 0 & a_{54}^{3} & a_{55}^{3} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22}^{1} & a_{23}^{1} & a_{24}^{1} & a_{25}^{1} \\ 0 & 0 & a_{33}^{2} & a_{34}^{2} & a_{35}^{2} \\ 0 & 0 & 0 & a_{44}^{3} & a_{45}^{3} \\ 0 & 0 & 0 & 0 & a_{55}^{3}-\dfrac{a_{54}^{3}\cdot a_{45}^{3}}{a_{44}^{3}} \end{bmatrix} \quad (3.4.2.24)$$

A última linha do membro direito da equação (3.4.2.24) é idêntica ao membro direito de (3.4.1.15) e, portanto, chegou-se a uma matriz [L] cuja inversa multiplicada pela matriz dos coeficientes fornece uma matriz [U], que armazena as diversas etapas da eliminação gaussiana. Encontrada essa matriz [U], o problema está resolvido e as incógnitas determinadas de forma rápida e eficiente. Para se resolver de vez o problema por essa rota, basta que se multipliquem ambos os membros da equação (1.1.1) pela inversa da matriz [L] conforme aqui definida, ou seja:

$$[L]^{(-1)} \times [K] \times \{x\} = [L]^{(-1)} \times \{b\}$$

(3.3.2.25)

3.4.3- O Algoritmo

O programa básico resolve o sistema de equações (1.1.1) pelo método da eliminação gaussiana via fatorização LU, conforme apresentado em 3.3.2 e foi formulado conforme a seguir indicado:

```
L [1]:=1;   (*Vetor que será utilizado para armazenar as colunas que
                contenham os valores significativos da matriz L*)
For x:=2 to 3*N do      (* N= número de nós da estrutura em análise*)
L[x]:=0;
i:=1;
j:=1;
For y:=1 to 3*N-1
begin
   while j<3*N do
   begin

      L_{j+1} = a_{j+1,i} / a_{ii}   ;(* a_{ij} é elemento da matriz [K] dos coeficientes*)
      j:=j+1;
   end;
   For A:=i+1 to 3*N do
   b_A := b_A + b_i × L_A ;(* b_A é elemento do vetor {b} das ações*)
   For A:=y+1 to 3*N do
   begin
      If  L_A ≠ 0 then
      begin
         For k:=1 to 3*N do
         begin
            If  a_{yk} ≠ 0 then
            begin
               a_{A,k} := a_{Ak} + L_A × a_{yk}

            end;
         end;
      end;
   end;
   i:=i+1;
   j:=i;
end;                (3.4.3.1)
```

Ao final da execução de (3.4.3.1), ter-se-a a matriz [U] armazenada por sobre a matriz [K] dos coeficientes.

3.4.4- A Retro Substituição

Obtida a matriz [U] e o vetor {b} (das ações), já passado pelo eliminador gaussiano, pode-se calcular o vetor {x} através do algoritmo apresentado a seguir.

R:=0;
$L_{3\times N} := b_{3\times N} / a_{3\times N,3\times N}$;
(* Utiliza-se o mesmo vetor L, criado para (3.4.3.1), para armazenar provisoriamente os deslocamentos. O vetor {b} e a_{ij} são respectivamente o vetor das ações e elemento da matriz dos coeficientes [K] após (3.4.3.1)*)
$L_{3\times N-1} := b_{3\times N-1} - a_{3\times N-1,3\times N} \times L_{3\times N} / a_{3\times N-1,3\times N-1}$;
For K:=3*N-1 downto 2 do
begin
$b_{K-1} = b_{K-1} - a_{K-1,3\times N} \times L_{3\times N}$;
y:=1;
A:=3*N-K+1;
while y<A do
begin
$R := R + a_{K-1,3\times N-y} \times L_{3\times N-y}$;
y:=y+1;
end;
$L_{K-1} := (b_{K-1} - R) / a_{k-1,3\times N-A}$;
R:=0;
end;

(3.4.4.1)

Ao término da execução de (3.4.4.1) estarão armazenados em "L" os valores dos deslocamentos devidos ao carregamento a que a estrutura analisada está submetida. Seria, no entanto, conveniente, que se armazenassem esses deslocamentos nos campos

para tal existentes nos registros de cada nó, o que se faz através do procedimento (3.4.4.2).

```
For j:=1 to N do
begin
   vetno[i].dax:=L3xi-2     ;
     (* vetno é o vetor que armazena os registros do tipo no *)
   vetno[i].dnor:= L3xi-1 ;
   vetno[i].rot:= L3xi  ;
end;                                    (3.4.4.2)
```

Ora, se foram obtidos os deslocamentos sofridos pelos nós em cada coordenada, pode-se voltar à análise de cada barra.

$$\{aep\}+[k].\{d\}=\{b\} \quad (3.4.4.3)$$

onde: {aep}= ações de engastamento perfeito do carregamento da barra;

[k] = matriz de rigidez local;

{d} = vetor dos deslocamentos, que era das incógnitas que agora é conhecido;

{b} = vetor da ações, como visto anteriormente e que agora são as incognitas (os esforços nas barras).

Há que se lembrar antes, que os deslocamentos estão expressos nas coordenadas globais, o que obriga a reescrevê-los nas coordenadas locais para que possam ser lançados em (3.4.4.3) e assim permitir o cálculo dos esforços nas barras. Para tanto, basta que se construa uma rotina que processe as expressões (3.4.4.3) sabendo-se que:

$$\{d\}=[\beta].\{\bar{D}\} \quad (3.4.4.4)$$

onde:

$\{d\}$ = é um vetor de seis posições, no qual estarão armazenados os deslocamentos dos nós de início e fim de uma barra, expressos nas coordenadas locais dessa barra, de acordo com o ilustrado na figura (1.3.1).

$[\beta]$ = é a matriz de incidencia cinemática de (3.1.5);

$\{\bar{D}\}$= é um vetor de seis posições, que abriga os deslocamentos dos nós de início e fim da barra obtidos do vetor dos deslocamentos globais.

As ações calculadas nas extremidades de cada barra, conforne (3.4.3), contribuem para a obtenção, via equilíbrio dos nós, das reações de apoios.

Finaliza-se o programa gravando-se no arquivo de saída, em forma de tabela, os deslocamentos dos nós, os esforços nas barras e os esforços reativos.

4- EXPLORANDO A ALOCAÇÃO DE MEMÓRIA

Já foi visto nos capítulos anteriores que o grande problema a ser enfrentado é a quantidade de dados que serão gerados pelo desenvolvimento do Processo dos Deslocamentos e que terão que ser armazenados.

A forma de escapar da limitação do bloco de 64 Kbytes citado no capítulo 1.3 deste trabalho é a técnica denominada de "Alocação Dinâmica de Memória", permitida pelo Pascal.

Trata-se de criar uma estrutura para dados do mesmo tipo, na qual se armazena no bloco reservado pelo compilador para variáveis, apenas o endereço físico de memória da máquina, obtido pelo compilador fora de suas áreas reservadas, no qual foram ou estão gravados os dados de interesse do programa. Sendo assim, pode-se guardar na área limitada apenas um endereço e a partir dele construir uma pilha de dados como se verá a seguir.

4.1-Construindo Uma Pilha

É importante ter em mente que quando se está trabalhando dentro da área reservada pelo compilador para o armazenamento de dados, ele está preparado para cuidar deles, armazená-los e recuperá-los sempre que forem comandadas tais ações. Assim, se houver um conjunto de registros de um tal tipo, basta que lhe seja informado que o vetor "X" armazena variáveis do tipo daquele registro, que ele ao receber o comando de retorno do registro número "Y" em "X", forneça a informação que foi lá armazenada. Mas, quando se está fora de sua área de trabalho não se poderá contar com sua gerência e será necessária a criação de mecanismos de busca dos campos desejados

exigindo da parte do programador o conhecimento de como administrar uma "estrutura de dados".

A estrutura de dados parte da criação de um registro no qual serão gravados, no caso da matriz de rigidez, o valor que se deseja armazenar e o endereço do próximo registro. A operação se inicia com a definição do tipo do registro como ilustrado em 4.1.1

```
type
aij_ptr=^aij_rec;                    (*1*)
aij_rec=record                       (*2*)          (4.1.1)
        valor_aij:real;
        next:aij_ptr;
end;
```

em (*1*), foi criada uma variável do tipo ponteiro (aij_ptr), que servirá para armazenar o endereço no qual estará o registro aij_rec, ou pode-se ainda dizer que "aponta" para o registro mencionado. Em (*2*) criou-se o registro, com um campo do tipo real, que servirá para armazenar um valor pertencente a uma posição da matriz de rigidez e o campo denominado "next", que será utilizado para a guarda do ponteiro, ou endereço do próximo registro que armazenará a próxima posição.

Cada vez que se desejar armazenar uma posição, chama-se a subrotina (procedure) **new**, como em 4.1.2.

```
new(aij_ptr)          (4.1.2)
```

Por esse procedimento o compilador aloca um segmento de memória suficiente para armazenar um registro e dentro da área reservada para dados (64Kbytes), armazena o endereço do segmento alocado.

Agora pode-se guardar o valor a ser armazenado em **aij_ptr^.valor**.

O conceito de pilha se assemelharia, como bem ilustrado por JONES & HARROW[6], a uma pilha de pratos num restaurante, ou seja, o próximo sempre por sobre o anterior, de tal forma que o último a entrar na pilha será o primeiro a sair. Dessa forma ter-se-á, na memória reservada para dados pelo compilador, sempre o endereço do último segmento alocado, de tal forma a exigir a necessidade de se criar uma sistematica de ligação entre os segmentos, para se poder recuperá-los em algum instante e para isso será necessário aquele campo **next**.

Para melhor ilustrar, foi proposta uma pequena matriz 2×2, como indicada em (4.3), para se simular o seu armazenamento.

$$\begin{bmatrix} 425 & 382{,}3 \\ 647{,}1 & 12{,}9 \end{bmatrix} \qquad (4.1.3)$$

Ao se encontrar o primeiro valor a ser armazenado (425), faz-se como em (4.2) e no primeiro registro se lança:

```
aij_ptr^.valor:= 425;
next:=nil;
```

a palavra reservada **nil** indica que este é o primeiro registro e portanto não há nada em baixo, ou seja, ele será o último da pilha. Antes de se chamar novamente **new**, seria bom salvar-se o endereço, ou ponterio dessa primeira posição numa variável tipo ponteiro, que aqui foi denominada de **ultima**, para que não se perca.

Para se guardar a próxima posição, procede-se como abaixo:

```
new(aij_ptr) ;
aij_ptr^.valor:= 382,3;          (4.1.4)
next:=ultima;
ultima:=aij_ptr;
```

Dessa maneira alocou-se um novo registro, que guardou o novo valor, o ponteiro do registro anterior e ainda salvou-se o ponteiro atual para a próxima alocação. Repetindo o processo até a última posição do sistema (4.1.3), terá sido construída a pilha e a matriz estará armazenada como se fosse um vetor.

Pode-se ainda, a fim de melhor ilustrar, construir um diagrama de alocação como na figura (4.1.1).

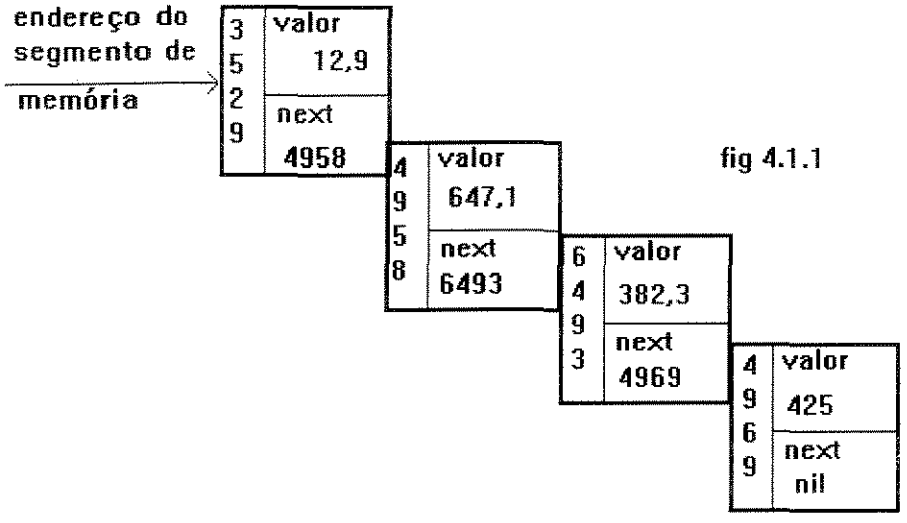

fig 4.1.1

conforme se pode constatar na ilustração da figura 4.1.1, foi construída uma pilha de elementos ligados uns aos outros, na qual estão armazenados os elementos da matriz 2×2 em forma de vetor.

4.2- Percorrendo uma Pilha

Para se recuperar um elemento da matriz armazenada como pilha, como visto em 4.1, necessita-se apenas saber seu número de ordem, uma vez que como ilustrado no exemplo 2×2, os elementos foram armazenados linha por linha e da primeira à última posição dentro de cada linha. Ora, para se recuperar então o valor de um elemento a_{ij}, basta calcular sua posição de ordem dentro da pilha que pode ser dada por:

$$O = n \times n - \left[(i-1) \times n + j\right] \qquad (4.2.1)$$

onde: O= número de vezes que se deve saltar registros na pilha para se chegar ao que interessa;

n= número de linhas ou colunas do sistema;

i= linha a que pertence o elemento procurado;

j= coluna a que pertence o elemento procurado;

Calculado "O", pode-se entrar num laço como indicado em 4.2.2.

```
topo:=aij_ptr;
for x:=1 to O do                    (4.2.2)
aij_ptr:= aij_ptr^.next;
aij_ptr:=topo;
```

onde: **topo**=variável tipo ponteiro criada para armazenar o endereço do topo da pilha.

É fácil verificar que ao final do laço, se terá em aij_ptr o endereço da posição procurada e nela o valor que se deseja.

Como se pode constatar, cada vez que houver a necessidade de recuperar o valor de uma posição da matriz na pilha, será necessário ir ao seu topo e percorrer através dela até que se chegue ao valor procurado. Só não será assim no caso particular em que se estiver num operador, no qual se busca ordenadamente as posições na sequência da pilha, ou seja, elemento a elemento, do fim para o começo. No caso do armazenamento pleno da matriz poderia ser montado o algoritmo da fatorização LU, como mostrado no Capítulo 3, de forma a se buscar sempre o próximo dentro da pilha. Mas como será visto a seguir, as técnicas de armazenamento da matriz de rigidez não são compatíveis com esse caso particular e, sendo assim, a busca como acima descrita torna o processamento

moroso demais. É bom lembrar que serão tratadas pilhas com dezenas de milhares de posições.

4.3-Acelerando a Varredura da Pilha

Como já se assinalou, os problemas comumente analisados, geram matrizes com dezenas de milhares de posições, tornando o processo de procura de elementos na pilha extremamente moroso ou até mesmo impraticável. Quando foram montadas as primeiras versões dos programas aqui apresentados, eles levavam horas para analisar a estrutura do Segundo Exemplo, capítulo 2.2, e o motivo como foi constatado, era o método de varredura da pilha. O caso mencionado (2.2), gera uma matriz de 14×114, ou seja, 12996 posições.

A solução encontrada foi criar mais um campo do tipo inteiro dentro do registro definido em 4.1.1 e estabelecer uma dupla ligação da pilha, criando mais um campo do tipo ponteiro naquele registro, de tal modo a se ter a indicação também do registro anterior. No novo campo inteiro lança-se o número de ordem do elemento da matriz transformada em vetor que poderia ser calculado como em 4.3.1.

$$\text{Ordem} := (i-1) \times n + j \qquad (4.3.1)$$

onde : Ordem = número de ordem do elemento a_{ij} na matriz armazenada como vetor;

i =número da linha em que está o elemento procurado;

j =número da coluna em que está o elemento procurado;

n =número de linhas e/ou colunas da matriz.

Aquela ilustração de 4.1.1 poderia ser reconstruída agora, como na figura 4.3.1.

A pilha de dados estruturada dessa forma permitirá uma varredura em busca de uma determinada posição de forma extremamente mais rápida. Dado um determinado a_{ij}, busca-se sua posição através de (4.3.1) e ao invés de se ir ao topo da pilha, simplesmente faz-se a verificação se no endereço constante do atual **aij_ptr** o valor de "ordem ", é menor ou maior que o procurado. Se for maior, percorre-se a pilha através dos ponteiros contidos em "next" e se menor se faz o caminho indicado pelos campos "anterior", até se chegar ao endereço desejado, ou seja, pode-se percorrer a pilha para frente ou para trás.

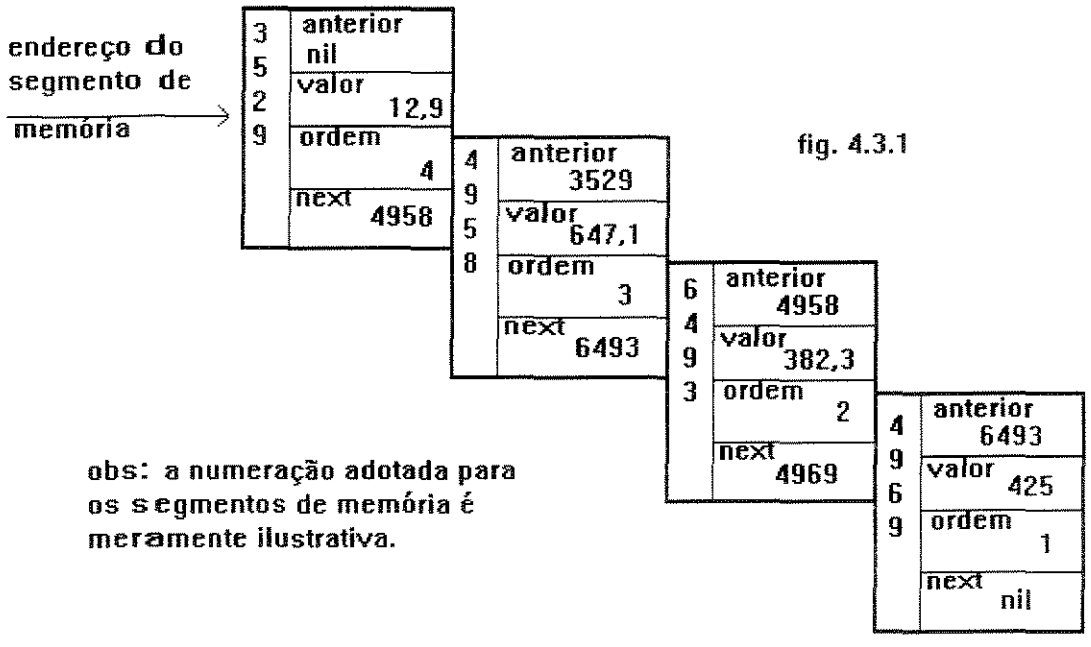

fig. 4.3.1

Assim um bom procedimento para busca de uma determinada posição seria como em (4.3.2).

```
Procedure Tripaij (posição;  var aij_ptr);
begin
while posição<aij_ptr do
   aij_ptr:=aij_ptr^.next;                    (4.3.2)
   while posição>aij_ptr do
```

```
            aij_ptr:=aij_ptr^.anterior
        end;
```

4.4-A Nova Capacidade de Armazenamento

Como foi visto no capítulo 1.3, a capacidade de armazenamento de dados na área reservada para tal pelo compilador é de 64 kbytes e, portanto, seria possível alocar um total de 8190 campos do tipo **double** (8 bytes cada um), sendo por essa via factível a análise de sistemas de no máximo 90×90. Se for considerada ainda a necessidade, como se viu no capítulo 3, de se armazenar os demais dados para a análise estrutural. o máximo que se conseguiria processar seriam estrututras de 23 nós, isso no caso bidimensional, pois se forem espaciais, não se chegaria aos 6 nós.

Pela técnica aqui apresentada, seria alocada a estrutura de dados fora do segmento de trabalho do sistema operacional, que é dimensionado por Roth,op. cit p.588, nos microcomputadores de 640Kbytes em 470.000 bytes. Como o registro estabelecido em 4.3 armazena 20 bytes (1 **double**, 2 **ponteiros**, 1 **longint**) para cada posição da matriz de rigidez, pode-se armazenar 21363 registros o que tornaria aqueles equipamentos aptos a analisar estruturas de cerca de 40 nós.

A versão do Pascal aqui contemplada, oferece três alternativas de operação, sendo que duas delas são apresentadas em forma de menu, quando acionado o ícone "compiler" da barra de menu superior da tela, na versão para DOS, que são :

- **REAL MODE**

Nessa opção o compilador se restringe à área abrangida pelo sistema operacional, ou seja, dentro dos 640 kbytes como já mencionado.

- **PROTECT MODE**

Já nessa situação o compilador proteje as áreas de atuação dele próprio e do sistema operacional, a fim de que não sejam acessadas pela alocação dinâmica tais segmentos e opera em toda a memória RAM do equipamento . O aparelho utilizado e apresentado no capítulo 2, possui 4Mbytes de memória RAM o que permitiria analisar estruturas planas de cerca de 100 nós.

A terceira opção é em ambiente Windows, quando o compilador além da memória RAM, ainda acessa o disco rígido para armazenamento, de forma automática, quando aquela se esgota.

5-A SIMETRIA E ESPARSIDADE DA MATRIZ DE RIGIDEZ

5.1-O Método Triangular

A matriz de rigidez global, da forma como é obtida pelo método, resulta simétrica e se assim é, não há necessidade de armazená-la por inteiro, pois sempre que for necessária a recuperação de uma posição não armazenada, basta que se invertam seus índices de localização na matriz plena (i e j) e se terá o valor procurado, sendo, portanto, necessário guardar pouco mais da metade da mesma.

Pode-se estocar desta forma os elementos não repetitivos, que assim o sejam por força da similaridade da matriz de rigidez, em um vetor, como proposto em Golub & Van Loan, op. cit. p.22, onde a correlação desejada pode ser formulada como em (5.1.1).

$$a_{ij} = \left((i-1).n - i.(i-1) \div 2 + j\right) \qquad \text{para todo } j \geq i \qquad (5.1.1)$$

onde:

a_{ij} = número de ordem do elemento da matriz de rigidez da linha i e coluna j, no vetor de armazenagem;

n = número de linhas ou de colunas da matriz;

Para melhor ilustrar, volte-se ao sistema 5×5, utilizado no capítulo 3, reescrevendo-o como em (5.1.2), tendo em vista a simetria aqui abordada.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{12} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{13} & a_{23} & a_{33} & a_{34} & a_{35} \\ a_{14} & a_{24} & a_{34} & a_{44} & a_{45} \\ a_{15} & a_{25} & a_{35} & a_{45} & a_{55} \end{bmatrix} \text{ a parte armazenada seria: } \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ & a_{22} & a_{23} & a_{24} & a_{25} \\ & & a_{33} & a_{34} & a_{35} \\ & & & a_{44} & a_{45} \\ & & & & a_{55} \end{bmatrix} \quad (5.1.2)$$

o vetor de armazenamento teria o aspecto da fig. 5.1.1.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{22}	a_{23}	a_{24}	a_{25}	a_{33}	a_{34}	a_{35}	a_{44}	a_{45}	a_{55}

fig. 5.1.1

O nome do método se deve à forma triangular dos elementos armazenados, como se vê em (5.1.2).

A partir do exposto no capitulo 4 e neste, foi construída a versão triangular do algoritmo apresentado no capítulo 3.

Nesse modelo foi estabelecido um processo de alocação dinâmica para os registros nós e barras e foi criado um novo registro, do tipo ilustrado na figura 4.3.1, para armazenar, como aqui apresentado, a matriz de rigidez.

Foi também estabelecido um processo de alocação dinâmica de memória, para o vetor das ações.

A construção da matriz U no algoritmo gaussiano, se faz por sobre a matriz de rigidez já armazenada.

O algoritmo completo pode ser visto no anexo 1.

5.2-O Método "Sparse"

Se for observada a figura 1.3.4, será de fácil percepção, que fora dos limites ali indicados as posições guardam zeros e mesmo dentro daqueles limites ainda haveriam muitos zeros. Se forem representadas por **x** as posições não-zero daquela estrutura sua matriz de rigidez poderia esquematicamente e de forma simplista ser apresentada como na fig.5.2.1.

x	0	O	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	x	x	0	x	x	0	0	0	0	0	0	0	0	0	0	0	0
		x	0	x	x	0	0	0	0	0	0	0	0	0	0	0	0
			x	0	0	x	0	0	0	0	0	0	0	0	0	0	0
				x	x	0	x	x	0	0	0	0	0	0	0	0	0
					x	0	x	x	0	0	0	0	0	0	0	0	0
						x	0	0	x	0	0	0	0	0	0	0	0
							x	x	0	x	x	0	0	0	0	0	0
								x	0	x	x	0	0	0	0	0	0
									x	0	0	x	0	0	0	0	0
										x	x	0	x	x	0	0	0
											x	0	x	x	0	0	0
												x	0	0	x	0	0
													x	x	0	x	x
														x	0	x	x
															x	0	0
																x	x
																	x

fig. 5.2.1

A constatação que fica é que os valores não-zero da matriz de rigidez são poucos e esparsos. Pode-se assim construir um vetor que armazene apenas as posições não nulas, dentro de um método triangular, como em 5.1.1, num processo de alocação dinâmica de memória como exposto no capítulo 4, criando um mecanismo que recupere a posição original na matriz de rigidez, desses elementos do vetor proposto posto sob a forma de um conjunto de registros.

Não é difícil criar o algoritmo imaginado quando se conhece a matriz, como ilustrado em (5.2.1). Mas se ela é conhecida, é sinal de que foi construída e armazenada por inteiro, não interessando mais qualquer recurso para economizar memória de armazenamento, ou seja, há que se criar esse mecanismo sonhado, sem conhecer a matriz de rigidez.

A forma encontrada para vencer esse aparente impasse foi a de montar a matriz linha por linha, ou seja, procedendo a uma varredura da estrutura em busca de barras, que contribuam para uma determinada linha, passando-se à construção dessa linha e armazenando-a no vetor mencionado. A partir da linha construida se faz a busca das posições não zero e o número de ordem que esse elemento teria se fosse armazenado pelo algoritmo (5.1.1), armazenando-os nos registros definidos no capítulo 4.3.

Dessa forma, definem-se as seguintes variáveis:

auxivet=um vetor auxiliar no qual serão reproduzidas cada linha da matriz de rigidez ;

linha=uma variável para registrar o número da linha que está sendo processada;

limin=uma variável para indicar o número de ordem que a primeira posição da linha teria, dentro de um hipotético vetor que armazenasse metade da matriz como em (5.1.1);

limax=uma variável para indicar o número de ordem que a última posição da linha teria, dentro de um hipotético vetor que armazenasse metade da matriz como em (5.1.1);

Z=uma variável para abrigar o número de ordem calculado por (5.1.1);

max=um contador;

gba=um vetor de ponteiros, aonde o processo de varredura descrito acima guarda os endereços dos registros que armazenam as barras que contribuem para a linha em construção;

numbar=número de barras cujos ponteiros estão em "gba";

Criadas as variáveis pode-se começar a construir a pilha de registros para armazenar os valores significativos da matriz de rigidez.

```
limax:=0;    linha:=0;    Z:=0;
max:=3*N;              (* N=número de nós da estrutura em análise*)
while max≥1 do
begin
    linha:=linha+1;    limax:=limax+max;    limin:=limax-max;
    numbar:=0;
(*************************************************************
        Faz-se a varredura da pilha que armazena as barras em busca
        de contribuintes da formação da linha que está sendo montada
        armazenando seus endereços em gba e incrementando numbar
*************************************************************)
    For contador:=1 to 3 do
    begin
                (* Cada nó contribue em tres linhas. portanto é preciso
              procurar qual das três contribuições interessa*)
        For cont:=3*N-max+1 to 3*N do
        auxivet(cont):=0;
                (* Lembrando que, no método triangular as linhas vão se
                   reduzindo, vamos preparar o auxivet para receber
                   apenas as posições que serão geradas*)
        g:=1;
        while g ≤ numbar g ≤ numbar do
        begin
            barra:=gba[g];
            assemblagem;
                (*O procedimento assemblagem se encontra ao final desta
                  rotina*)
            g:=g+1;
        end;
        for y:=linha to 3*N do
        begin
            (*percorre-se agora a linha construida e armazenada
              em auxivet, em busca das posições não-zero*)
            Z:=Z+1;
            (* incrementa-se o valor da ordem para o novo elemento*)
            If auxivet[y]<>0 then
            begin
                new(aij_ptr);
                (* O procedimento new, providencia a alocação de um registro
                   do tipo ilustrado em 4.3.1*)
                If y<>1 then
                begin
                    aij_ptr^.next:=última;    última^.ant:=aij_ptr;
                end
                else
                begin
                    If linha=1 then
                    aij_ptr^.next:=nil;
                end;
```

```
                    aij_ptr^.valor:=auxivet[y];          aij_ptr^.ordem:=Z;
                    última:=aij_ptr;                     posição:=posição+1;
                       (* A variável posição tem por objetivo contar a quantidade de
                          registros abertos para armazenar as posições não-zero*)
                  end;
              end;
               If contador <3 then
               begin
                  linha:=linha+1;          max:=max-1;
                  limin:=limax;            limax:=limax+max;
                   (*Incrementou-se tudo para a construção da próxima linha*)
               end;
          end;
           max:=max-1;
      end;
**********************************************************************************
**********************************************************************************
**********************************
    Procedure Assemblagem;
    begin
       sinal:=0;
       (*Percorre-se a pilha referente ao nó em busca dos endereços dos
         nós de inicio e fim da barra em questão armazenando-os em duas
         variáveis do tipo ponteiro que denominamos inno e finno*)
       (*Reproduziu-se aqui apenas a condição em que o nó início é menor
         que o nó final. Para a listagem completa. veja o anexo 2*)
       lin:=ni × 3-2;      col:=nf× 3-2;
       (*Esta é a condição que gera o menor número de ordem da contribuição da
         barra em análise ao hipotético vetor de armazenagem triangular*)
       vecalc;
       If XX<=limax then
       begin
            (* Se o menor número de ordem (XX). for maior que limax as contribuições da barra em
análise estarão fora da linha que está sendo construida e portanto essa barra não interessa.*)
          lin:=nf× 3;       col:=nf× 3;
             (*Esta é a condição que gera o maior número de ordem da contribuição
               da barra ao hipotético vetor de armazenagem triangular*)
          vecalc;
          If XX>=limin then
          begin
             For x:=2 downto 0 do
             For y:=2 downto o do
             begin
                lin:=ni× 3-X;          col:=ni× 3-Y;
                vecalc;
                If XX<=limax then
                If XX>limin then
                begin
                   If sinal:=0 then
                   begin
                      (* A variável sinal quando igual a 0. quer dizer que a matriz
                       de rigidez local da barra em análise ainda não foi calculada*)
                   constcc;
                  (* constroi a matriz de rigidez local que foi chamada de
```

```
                                CC[X.Y]*)
                            sinal:=1;
                        end;
                        auxivet[xx-limin+3 × n-max]:= auxivet[xx-limin+3 × n-max]+
                                                      cc[6-x.6-y]
                    end;
*************************************************************************
        Continua com as outras combinações para concluir a montagem do auxivet .
      procedure vecalc;
*************************************************************************
    (*calcula a posição do elemento num hipotético vetor do método triangular
      utilizando a relação (6.1.1)*)
    begin
      XX:= ((lin-1).3*n-lin.(lin-1)div2)+col;
        (*lin é o número da linha em que está o elemento e col o número da coluna*)
    end;
```

Cuidado especial deve ser tomado durante a eliminação gaussiana, pois a mesma altera a esparsidade da matriz de rigidez, de maneira que muitas posições que originalmente guardavam zeros passam a ter não-zeros exigindo assim um novo registro que deve ser inserido na pilha.

A exemplo do que se fez no método triangular, a matriz U terá seus valores gravados nos registros correspondentes da matriz de rigidez, além daqueles que serão criados tendo em vista o disposto no parágrafo anterior.

6-A FORMAÇÃO EM BANDA

6.1-O Método Retangular

Uma observação mais detalhada da matriz de rigidez mostra que para uma numeração estudada das malhas e nós, no caso de uma discretização genérica, ou no caso particular aqui enfocado das barras e nós, se verá que as posições não-zero da matriz de rigidez determinam uma formação tipo banda, tendo por eixo a diagonal principal conforme se pode observar na figura 5.2.1.

A partir dessa constatação podem ser apresentados alguns conceitos, tais como:

-Diz-se que uma matriz tem largura inferior de banda p, se:

$$a_{ij} = 0, \qquad \text{quando} \qquad j < i - p \qquad (6.1.1)$$

_Diz-se que uma matriz tem largura superior de banda q, se:

$$a_{ij} = 0, \qquad \text{quando} \qquad j > i + q \qquad (6.1.2)$$

Se for observada a figura 5.2.1, se verificará que ela tem q=4 e por força de sua simetria p=4. Não haveria nenhuma dificuldade em se montar um algoritmo que recuperasse as posições originais dos valores armazenados numa matriz auxiliar que guardasse apenas os elementos internos à banda de largura q , que para o exemplo da fig. 5.2.1 seria, como na fig. 6.1.3. A matriz armazenada seria retangular e daí o nome do método.

O método retangular acaba armazenando um triângulo de zeros, se assim se pode chamar, na base da matriz auxiliar, zeros nos limites da banda, além, naturalmente, dos

zeros internos `a banda. Essas posições de zeros podem ser em número bem grande se for lembrado que serão tratadas estruturas de grande porte, que acabam gerando grandes larguras de banda e pode até ser ineficiente se estiverem sendo trabalhadas estruturas fechadas, que inviabilizam a formação de banda, como se pode ver no exemplo da figura 6.1.2.

$$\begin{bmatrix} x & 0 & 0 & x & 0 \\ x & x & 0 & x & x \\ x & 0 & x & x & 0 \\ x & 0 & 0 & x & 0 \\ x & x & 0 & x & x \\ x & 0 & x & x & 0 \\ x & 0 & 0 & x & 0 \\ x & x & 0 & x & x \\ x & 0 & x & x & 0 \\ x & 0 & 0 & x & 0 \\ x & x & 0 & x & x \\ x & 0 & x & x & 0 \\ x & 0 & 0 & x & 0 \\ x & x & 0 & x & x \\ x & 0 & x & x & 0 \\ x & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 \\ x & 0 & 0 & 0 & 0 \end{bmatrix}$$

fig. 6.1.1

A simples estrutura de 4 barras da figura 6.1.2, não tem formação de banda suficiente para a formação de uma matriz retangular, como se pode constatar na ilustração da matriz de rigidez em 6.1.3.

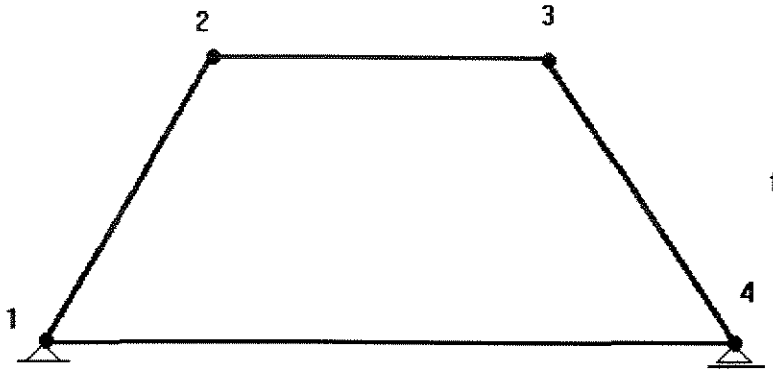

fig. 6.1.2

$$\begin{bmatrix}
x & x & x & x & x & x & & & & x & x & x \\
x & x & x & x & x & x & & & & x & x & x \\
x & x & x & x & x & x & & & & x & x & x \\
x & x & x & x & x & x & x & x & x & & & \\
x & x & x & x & x & x & x & x & x & & & \\
x & x & x & x & x & x & x & x & x & & & \\
 & & & x & x & x & x & x & x & x & x & x \\
 & & & x & x & x & x & x & x & x & x & x \\
 & & & x & x & x & x & x & x & x & x & x \\
x & x & x & & & & x & x & x & x & x & x \\
x & x & x & & & & x & x & x & x & x & x \\
x & x & x & & & & x & x & x & x & x & x
\end{bmatrix}$$

fig. 6.1.3

Como se pode constatar, praticamente não existe formação de banda esbelta em estruturas deste tipo o que torna o método retangular não vantajoso para esses casos, uma vez que vai armazenar em termos práticos, a mesma quantidade de posições do método triangular.

6.2-O Método "Skyline"

Um método muito utilisado e considerado por muitos autores como bastante eficiente é o denominado Skyline. Esse método explora, não de forma tão radical como o retangular, a formação em banda da matriz de rigidez. Bahte & Wilson [7] e Wilson et. al [8] discorrem sobre o método e como o utilisaram nos programas SAP4, NONSAP e ADINA.

O método consiste em determinar a altura de elementos não-zero em cada coluna, tendo como base a diagonal principal. Para melhor ilustrar pode-se voltar à figura esquemática ilustrada em 5.2.1. Poderia ser extraida dela apenas a parte das colunas que contenham não-zeros e reescreve-la, como na figura 6.2.1.

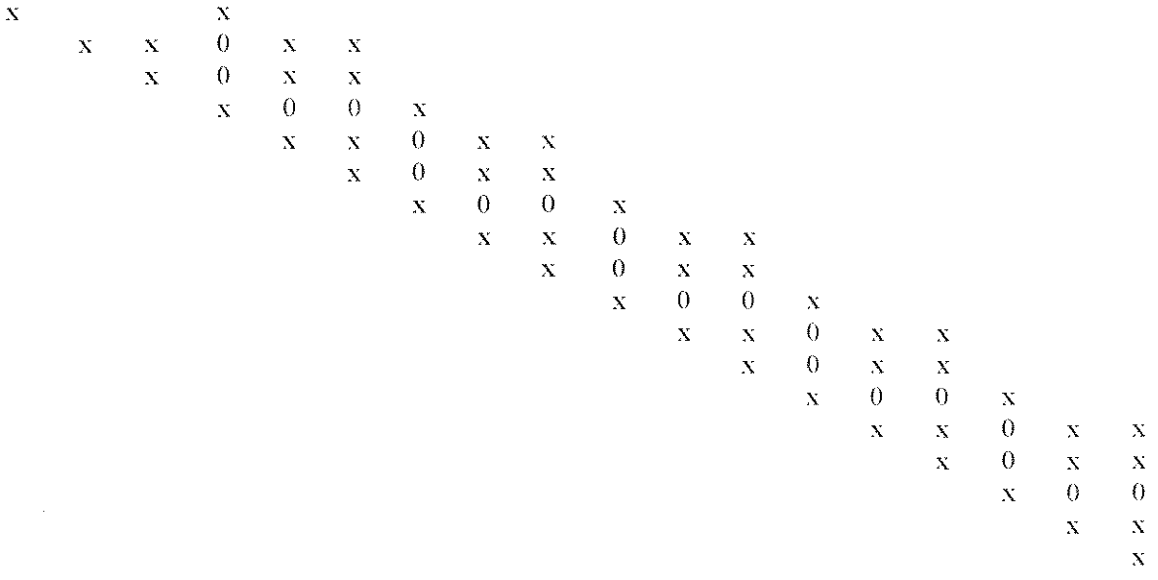

fig. 6.2.1

Pode-se agora construir um vetor que associe a cada coluna sua "altura útil", como apresentado na figura 6.2.2.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	2	4	4	5	4	4	5	4	4	5	4	4	5	4	4	5

fig. 6.2.2

Dessa maneira os únicos zeros armazenados são aqueles internos à banda, ou abaixo do teto estabelecido pela primeira posição não-zero de cada coluna, quando percorrida de cima para baixo.

A exemplo do já constatado no capítulo 5.2, será necessária a armazenagem dos elementos da matriz de rigidez pelo método Skyline, sem construir a matriz e a estratégia utilizada foi novamente construir linha por linha como lá foi feito.

Definiu-se previamente um vetor para o armazenamento das alturas de cada coluna. Se fosse para armazenar toda a matriz de rigidez e utilizando novamente o caso da figura 5.2.1, o vetor seria como ilustrado na figura 6.2.3.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

fig. 6.2.3

Ao ser construida a primeira linha, como já apresentado em 5.2, se terá de forma esquemática:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
x	0	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0

fig. 6.2.4

Observando a linha construída pode-se notar que a primeira coluna apresenta, já na primeira linha, um não-zero e, portanto, sua altura já está definida e este elemento será armazenado num registro, bem como seu número de ordem naquele hipotético vetor do método triangular, como se fez no caso Sparse, além de ser respeitada a altura original constante do vetor das alturas ilustrado na figura 6.2.3. A segunda coluna apresenta um zero e, portanto, sua altura deve ser subtraída de um e nada será armazenado. A terceira coluna também abriga um zero e sua altura constante do vetor das alturas será respeitada, pois serão armazenados todos os elementos da coluna e assim por diante. Ao final da varredura da primeira linha o vetor estará como na figura 6.2.5.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	2	4	4	5	6	7	8	9	10	11	12	13	14	15	16	17

fig 6.2.5

6.3-Otimização da Banda da Matriz de Rigidez

Como foi possível verificar ao longo de 6.1 e 6.2, a formação em banda de forma compacta, reduz significativamente a armazenagem de elementos da matriz de rigidez.

Pode-se no entanto adiantar que não existe um processo absolutamente ótimo para esse fim, mas recomendações que devem ser seguidas na busca de uma banda compacta conforme apresentado por Duff et al, op. cit p148-176.

A forma mais eficiente de buscar uma banda compacta foi proposta por Cuthil & McKee [9], e propõe que se dividam os nós em níveis de conjuntos. Mais compacta será a banda quanto maior for o número de conjuntos de nós.

Para melhor ilustrar, foi reproduzido o exemplo apresentado por Duff et al,op. cit, p.153:

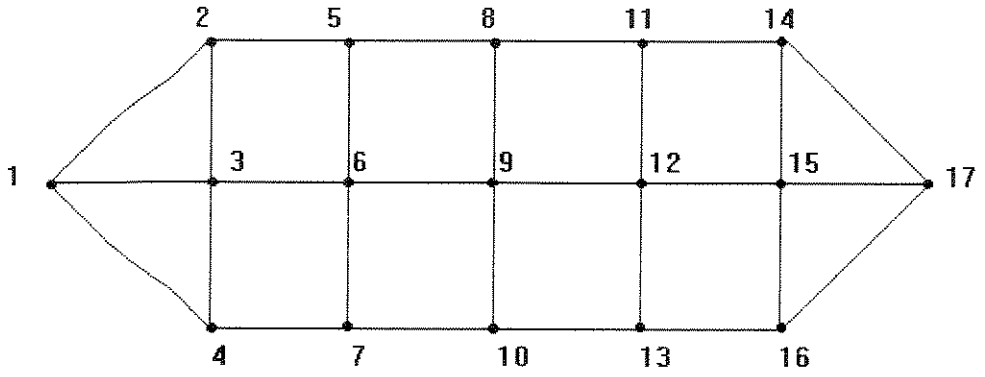

fig. 6.3.1

O primeiro conjunto de nós será formado por um único nó isolado. O segundo conjunto será formado pelos nós vizinhos àquele do primeiro conjunto, considerando-se vizinhos àqueles que têm barras em comum, ou que pertençam ao mesmo elemento de uma discretização. O terceiro conjunto será formado pelos vizinhos dos nós do segundo e assim por diante. Se for tomado como exemplo o ordenamento da figura 6.3.1, serão encontrados os seguintes conjuntos:

(1), (2,3,4), (5,6,7), (8,9,10), (11,12,13), (14,15,16), (17) (6.3.1)

A estrutura da figura 6.3.1, assim numerada, levaria à formação de uma matriz ilustrada de forma simplista, como na figura 6.3.2.

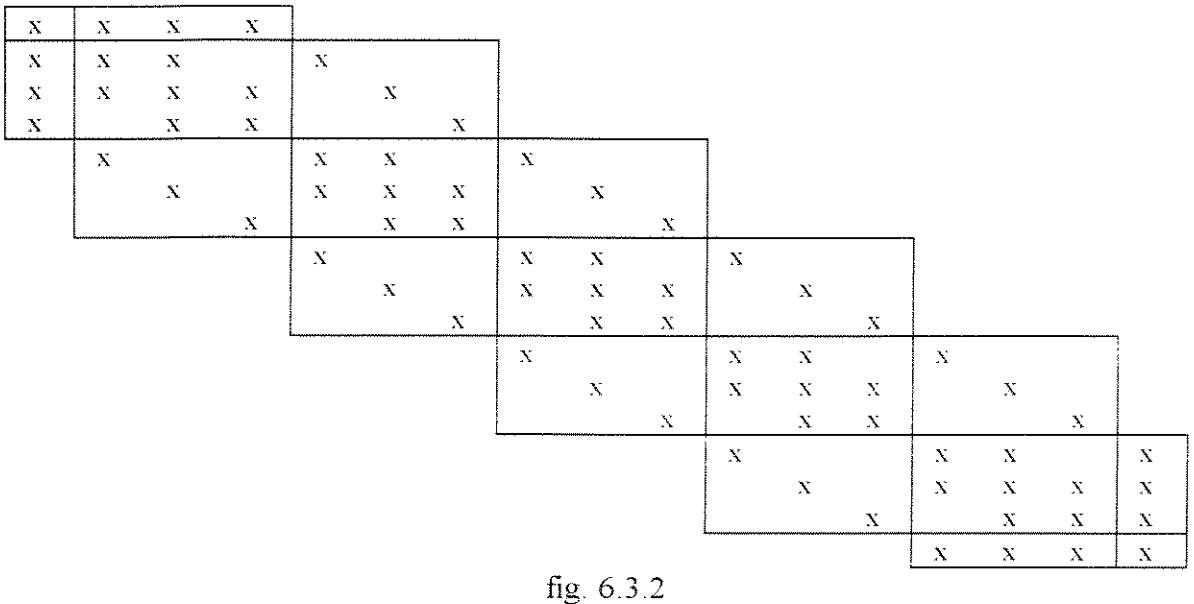

fig. 6.3.2

Pode-se perceber claramente, pela análise da figura (6.3.2), a formação de uma matriz chamada por Duff et al,op. cit, p.154, de bloco tridiagonal, na qual podem ser vistas submatrizes 1×1, 3×3, 3×3, 3×3, 3×3, 3×3 e 1×1, exatamente a sequência de (6.3.1). Golub & Van Loan [3], afirmam que os demais métodos que surgiram desde então, não oferecem vantagens significativas.

George [10], propõe uma inversão da ordenação de Cuthil & McKee, promovendo, não uma melhora no tamanho da banda, mas na exigência de armazenamento total e no número de operações aritméticas no processo de eliminação gaussiana. Para ilustrar a proposição, Duff et. all, op. cit, p. 154-155, apresentam o exemplo da figura 6.3.3.

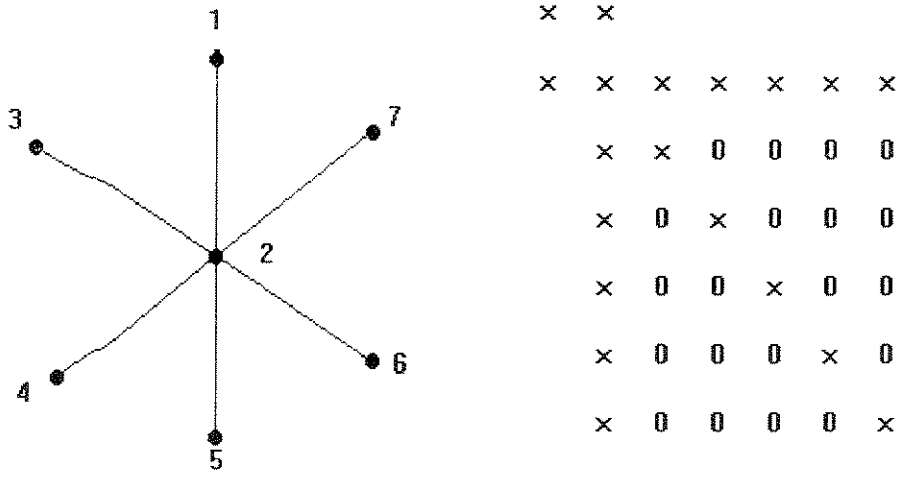

fig. 6.3.3

A figura 6.3.3, apresenta uma estrutura numerada segundo os critérios de Cuthil & McKee e a esquematização de sua matriz de rigidez simplificada. A figura 6.3.4, apresenta a mesma estrutura numerada de forma inversa como proposto por George[10] e a nova esquematização simplificada da matriz de rigidez.

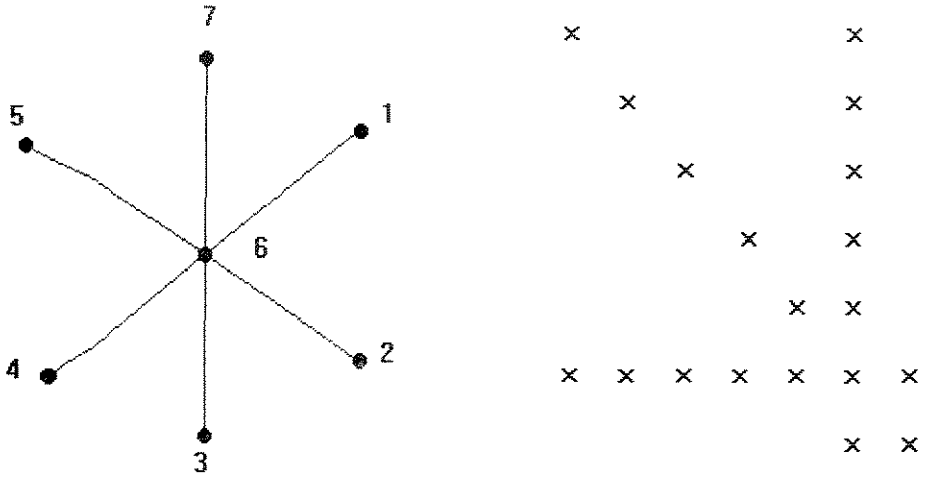

fig. 6.3.4

Se for considerado o mecanismo do método Skyline, poderá ser facilmente verificada a redução de armazenamento e consequente redução do número de operações do eliminador gaussiano.

7-RESULTADOS OBTIDOS E CONCLUSÕES

7.1-Resultados Obtidos

Do Programa Básico apresentado no capítulo 3, foram formuladas três versões:

- Triangular
- Sparse
- Skyline

As três versões são portanto o mesmo algoritmo adaptado para cada técnica, o que nos oferece uma confiável base de comparação. As três versões se utilizam da técnica de alocação dinâmica de memória, como descrito no capítulo 4, para armazenamento dos dados da estrutura em análise, da matriz de rigidez global e do vetor de ações. As três versões foram dotadas de um mecanismo de registro de tempo de processamento, de maneira a se poder aferir e comparar o desempenho, quando utilizado o microcomputador com a configuração descrita no capítulo 2.

7.1.1- Resultados Obtidos Para a Estrutura do Primeiro Exemplo

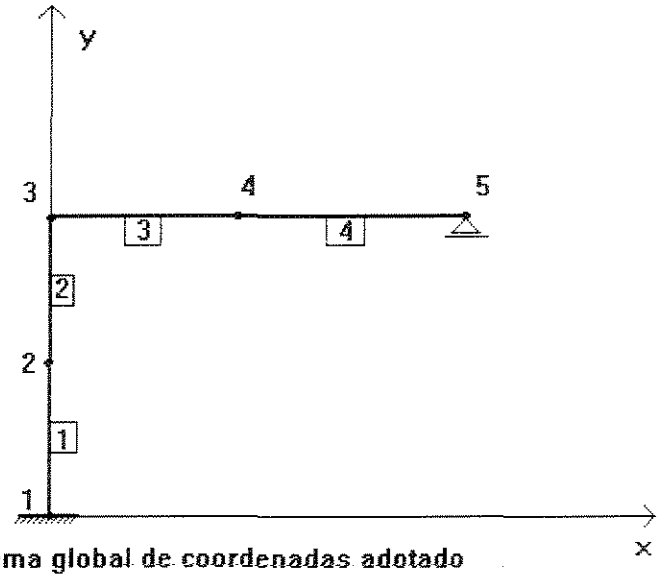

fig 7.1.1.1

x e y, sistema global de coordenadas adotado

A estrutura apresentada em 2.1, discretizada em 4 barras, teve seus nós e barras numerados conforme indicado na figura 7.1.1.

7.1.1.1- Método Triangular

O método triangular apresentou, para a estrutura proposta em 2.1, numerada e orientada conforme a fig 7.1.1, o seguinte arquivo de saída:

FOI ANALISADA A ESTRUTURA CUJOS DADOS FIGURAM NO ARQUIVO

teste1.pas

A matriz de rigidez do caso analisado, tem 225 posicoes

e portanto o metodo triangular armazenou 53,33 % da referida matriz de rigidez.

OS RESULTADOS ENCONTRADOS FORAM:

DESLOCAMENTOS

NO	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	0.000000	-0.000000	0.000000
2	0.000505	-0.000008	-0.000444
3	0.001457	-0.000016	-0.000501
4	0.001457	-0.000843	0.000151
5	0.001457	-0.000000	0.000565

ESFORÇOS NAS BARRAS

BARRA	NO INICIAL			NO FINAL		
	AXIAL	NORMAL	MOMENTO	AXIAL	NORMAL	MOMENTO
1	4.09	1.60	3.35	-4.09	-0.40	-1.49
2	4.09	0.40	0.49	-4.09	-0.00	-0.22
3	-0.00	2.09	0.22	0.00	-0.59	3.81

4	-0.00	-1.41	-3.81	0.00	2.41	0.00

REAÇÕES NOS APOIOS

NO	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	-1.60	4.09	3.35
5	0.00	2.41	0.00

O tempo total de processamento utilizado pelo metodo triangular foi de 0,28 segundos.

(7.1.1.1)

Os resultados observados em (7.1.1.1), são rigorosamente iguais aos obtidos pela análise via SAP90, conforme pode ser verificado em A5.1 (Anexo 5).

7.1.1.2- Método "Sparse"

O método sparse apresentou, para a estrutura proposta em 2.1, numerada e orientada conforme a fig 7.1.1, um arquivo de saída com os mesmos resultados apresentados em (7.1.1.1), diferindo apenas na quantidade de posições armazenadas para a matriz de rigidez e no tempo de processamento, que estão indicados abaixo:

Posições armazenadas da matriz de rigidez : 21

Posições armazenadas da matriz [U] : 37

Armazenamento relativo da Matriz [U] : 16,44 %

Tempo total de processamento : 0,17 segundos

7.1.1.3- Método "Skyline"

O método skyline apresentou, para a estrutura proposta em 2.1, numerada e orientada conforme a fig 7.1.1, a exemplo do observado em 7.1.1.2, um arquivo de saída com os mesmos resultados apresentados em (7.1.1.1), diferindo apenas na quantidade de posições armazenadas para a matriz de rigidez e no tempo de processamento, que estão indicados abaixo:

Posições armazenadas da matriz de rigidez : 43

Posições armazenadas da matriz [U] : 43

Armazenamento relativo da Matriz [U] : 19,11 %

Tempo total de processamento : 0,21 s

7.1.1.4- Diagramas

Com o intuito de construir os diagramas de forças normais, forças cortantes e momentos fletores, foi feita a rediscretização da estrutura contemplada nesta seção, dividindo-se cada barra em 5 outras. Com a nova configuração, procedeu-se à análise da estrutura atrvés do método sparse e foram construidos os gráficos apresentados em 7.1.1.4.1, 7.1.1.4.2 e 7.1.1.4.3.

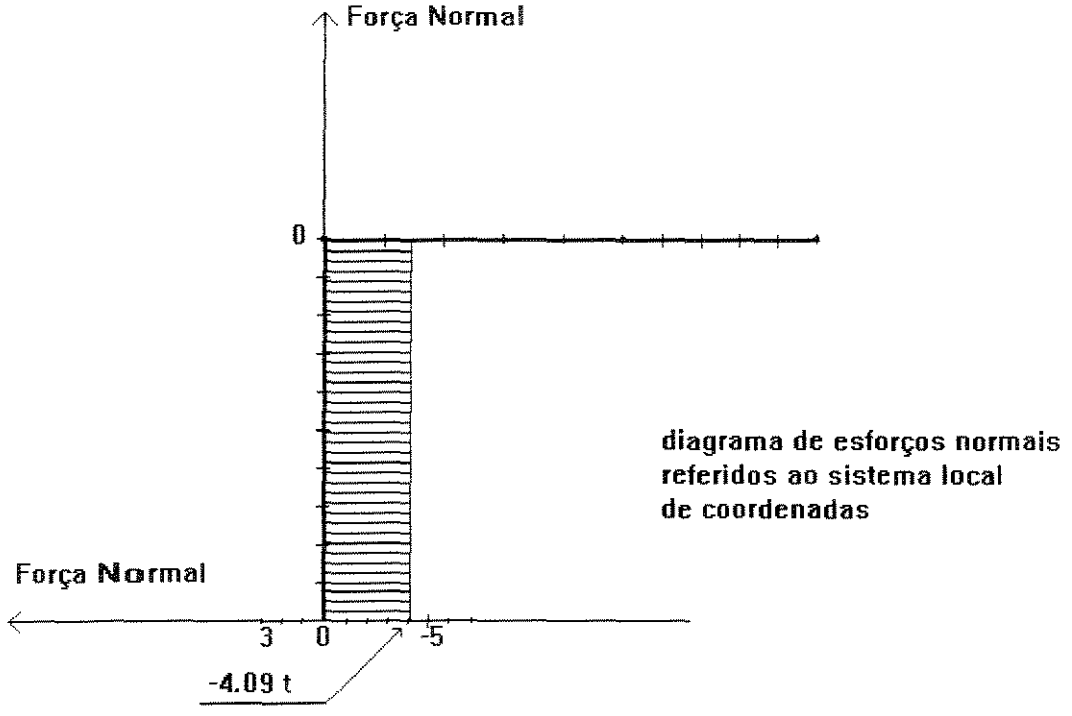

fig. 7.1.1.4.1
diagrama de forças cortantes referidas ao sistema local de coordenadas

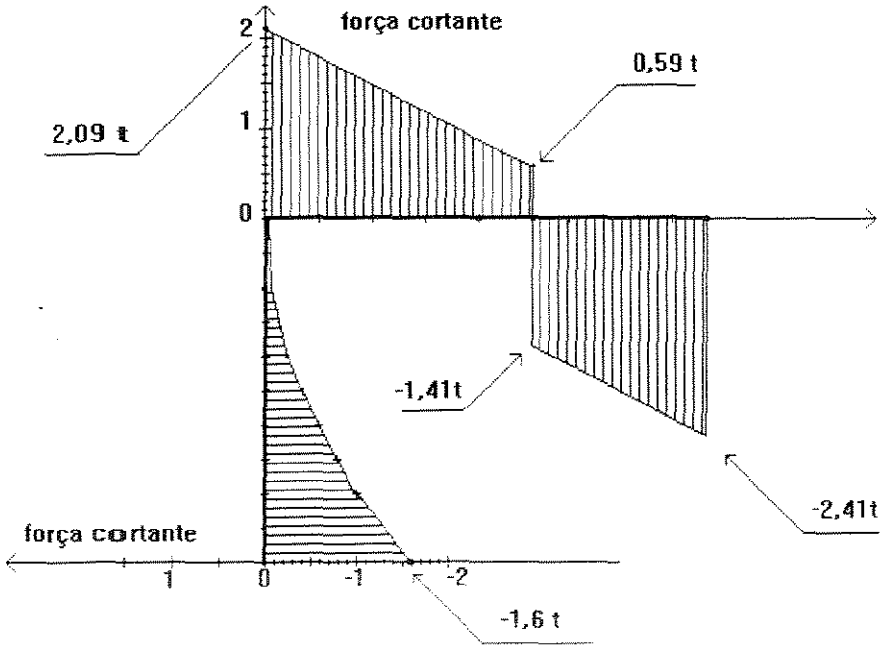

fig. 7.1.1.4.2

diagrama de momentos fletores referidos ao sistema local de coordenadas

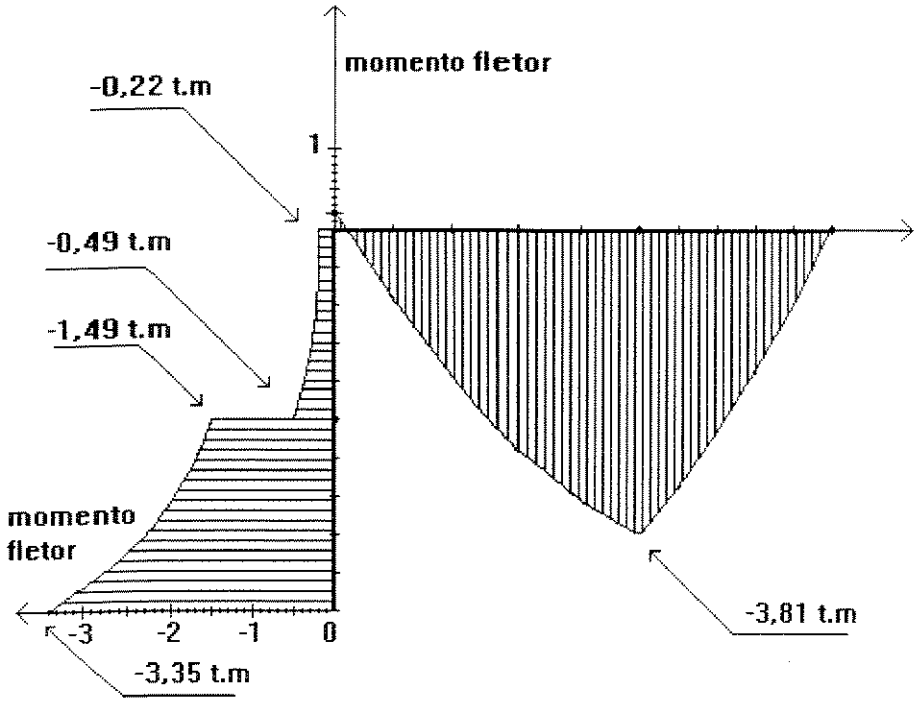

fig 7.1.1.4.3

7.1.2- Resultados Obtidos Para a Estrutura do Segundo Exemplo

A estrutura apresentada em 2.2, teve seus nós numerados, como apresentado na figura (7.1.2.1).

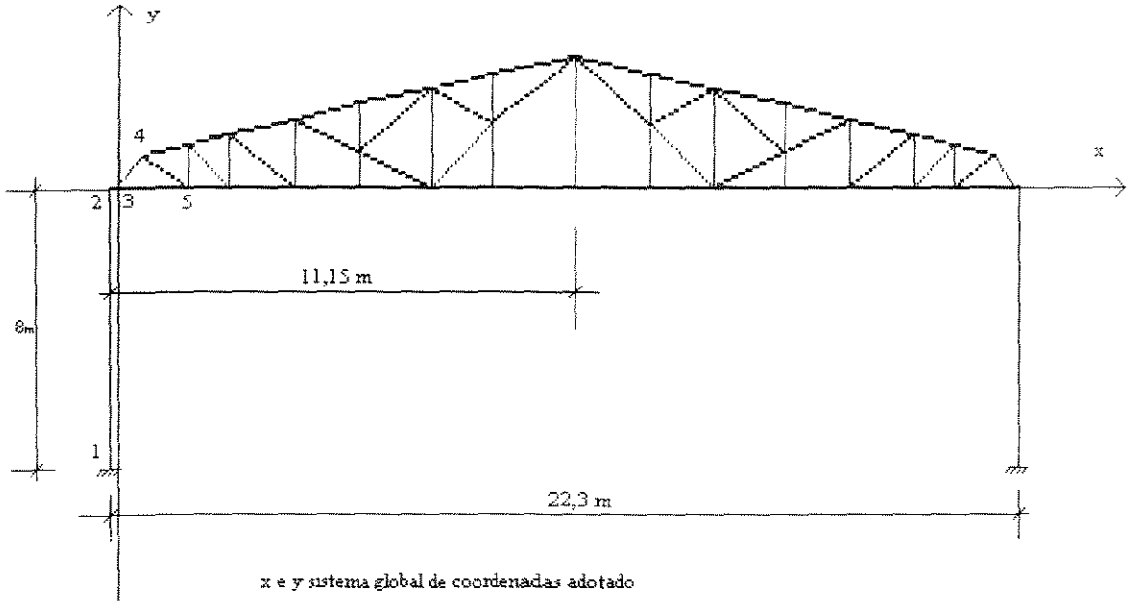

fig. (7.1.2.1)

7.1.2.1- Método Triangular

O método triangular apresentou, para a estrutura proposta em 2.2, numerada e orientada conforme a fig 7.1.2.1, conforme o seguinte arquivo de saída:

FOI ANALISADA A ESTRUTURA CUJOS DADOS FIGURAM NO ARQUIVO

reserra.pas

A matriz de rigidez do caso analisado tem 12996 posições e ,portanto, o método triangular armazenou 50,44% da referida matriz de rigidez.

OS RESULTADOS ENCONTRADOS FORAM:

DESLOCAMENTOS

NO	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	0.000000	0.000000	0.000000
2	0.043759	-0.000065	-0.007032
3	0.043759	-0.001119	-0.007032
4	0.045370	-0.002199	-0.001037
5	0.043793	-0.004250	-0.001327
6	0.045670	-0.004344	-0.001894
7	0.043919	-0.005993	-0.001468
8	0.045798	-0.006073	-0.001233
9	0.044191	-0.007804	-0.000873
10	0.045722	-0.007790	-0.000807
11	0.044450	-0.008861	-0.000352
12	0.044935	-0.008862	-0.000427
13	0.045604	-0.008933	-0.000349
14	0.044736	-0.008691	0.000025
15	0.045084	-0.008845	0.000095
16	0.044648	-0.008609	0.000315
17	0.045209	-0.008616	0.000105
18	0.044553	-0.008596	0.000796
19	0.044542	-0.006541	0.000468

20	0.043473	-0.006517	0.000440
21	0.044435	-0.006971	0.000276
22	0.043225	-0.006977	0.000413
23	0.042895	-0.007018	-0.000031
24	0.044346	-0.006114	0.000577
25	0.042725	-0.006212	0.000528
26	0.044288	-0.005145	0.000664
27	0.043820	-0.005148	0.000694
28	0.042595	-0.005165	0.000929
29	0.044230	-0.003139	0.001069
30	0.042759	-0.003194	0.001076
31	0.044116	-0.001430	0.000918
32	0.042880	-0.001510	0.001011
33	0.043992	-0.000230	0.001758
34	0.043025	-0.000337	0.000717
35	0.043234	0.000826	0.002326
36	0.043686	0.001138	-0.007816
37	0.043686	-0.000034	-0.007816
38	0.000000	0.000000	0.000000

ESFORÇOS NAS BARRAS

BARRA	NO INICIAL			NO FINAL		
	AXIAL	NORMAL	MOMENTO	AXIAL	NORMAL	MOMENTO
1	37.28	-4.96	-3.76	-37.28	4.96	-1.40

2	40.16	1.13	0.77	-40.16	-1.13	0.45
3	48.94	0.04	-0.10	-48.94	-0.04	0.15
4	50.44	-0.04	-0.09	-50.44	0.04	0.02
5	39.90	0.05	-0.01	-39.90	-0.05	0.10
6	42.04	-0.17	-0.21	-42.04	0.17	-0.11
7	44.98	0.20	0.07	-44.98	-0.20	0.25
8	44.70	-0.42	-0.36	-44.70	0.42	-0.44
9	37.73	0.38	0.41	-37.73	-0.38	0.30
10	37.99	-0.19	-0.22	-37.99	0.19	-0.08
11	34.54	0.14	0.09	-34.54	-0.14	0.18
12	34.15	-0.13	-0.13	-34.15	0.13	-0.09
13	31.10	0.05	0.05	-31.10	-0.05	0.04
14	24.64	-0.48	-0.20	-24.64	0.48	-0.31
15	15.05	0.91	0.18	-15.05	-0.91	0.80
16	11.82	-7.40	-1.86	-11.82	7.40	-5.85
17	-3.60	-2.27	-2.49	3.60	2.27	-0.94
18	-19.96	0.63	0.36	19.96	-0.63	0.30
19	-27.33	-0.05	-0.12	27.33	0.05	0.03
20	-26.11	0.03	-0.04	26.11	-0.03	0.09
21	-25.89	-0.19	-0.22	25.89	0.19	-0.13
22	9.20	0.12	0.05	-9.20	-0.12	0.13
23	9.62	-0.56	-0.53	-9.62	0.56	-0.49
24	9.65	0.46	0.44	-9.65	-0.46	0.39

25	9.37	-0.12	-0.13	-9.37	0.12	-0.05
26	5.27	0.06	0.05	-5.27	-0.06	0.07
27	5.79	-0.34	-0.33	-5.79	0.34	-0.22
28	11.51	-0.05	-0.02	-11.51	0.05	-0.06
29	19.43	0.42	0.05	-19.43	-0.42	0.38
30	33.22	-4.25	-1.91	-33.22	4.25	-4.51
31	-22.57	0.84	0.63	22.57	-0.84	0.55
32	-10.96	-0.17	-0.19	10.96	0.17	-0.08
33	1.68	0.01	-0.02	-1.68	-0.01	0.04
34	14.33	0.06	0.02	-14.33	-0.06	0.10
35	20.31	-0.13	-0.18	-20.31	0.13	-0.09
36	-7.46	-0.05	-0.10	7.46	0.05	-0.02
37	2.42	-0.02	-0.02	-2.42	0.02	-0.02
38	-25.90	0.09	0.09	25.90	-0.09	0.11
39	-23.81	-0.25	-0.36	23.81	0.25	-0.31
40	-0.62	0.21	0.28	10.62	-0.21	0.28
41	-7.35	-0.05	-0.07	7.35	0.05	-0.04
42	-2.17	-0.01	-0.02	2.17	0.01	0.01
43	-0.87	0.01	-0.00	0.87	-0.01	0.02
44	-0.94	0.06	0.05	0.94	-0.06	0.07
45	-2.54	-0.21	-0.24	2.54	0.21	-0.16
46	-7.42	0.04	0.05	7.42	-0.04	0.04
47	-12.08	-0.14	-0.09	12.08	0.14	-0.14

48	-16.25	1.41	0.90	16.25	-1.41	1.07
49	12.77	-0.09	0.04	-12.77	0.09	-0.15
50	8.89	-0.09	-0.10	-8.89	0.09	-0.04
51	-1.21	-0.03	-0.04	1.21	0.03	-0.02
52	0.23	0.22	0.13	-0.23	-0.22	0.09
53	8.63	0.15	0.09	-8.63	-0.15	0.11
54	8.90	0.06	0.07	-8.90	-0.06	0.08
55	0.68	0.43	0.40	-0.68	-0.43	0.35
56	-2.12	0.02	-0.08	2.12	-0.02	0.11
57	-1.02	0.03	0.06	1.02	-0.03	0.05
58	0.58	-0.28	-0.26	-0.58	0.28	-0.23
59	4.39	-0.03	0.04	-4.39	0.03	-0.08
60	5.66	-0.01	-0.00	-5.66	0.01	-0.01
61	0.40	0.52	0.26	-0.40	-0.52	0.27
62	2.05	-0.13	-0.12	-2.05	0.13	-0.05
63	4.68	0.20	0.20	-4.68	-0.20	0.20
64	8.78	0.15	0.10	-8.78	-0.15	0.12
65	14.44	0.74	0.62	-14.44	-0.74	0.27
66	17.64	28.82	-1.93	-17.64	-28.82	6.26
67	31.94	-18.08	10.36	-31.94	18.08	-13.07
68	28.77	62.27	220.55	-28.77	7.97	-3.36
69	18.13	29.94	14.20	-18.13	-44.74	284.53

REAÇOES NOS APOIOS

NO	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	-62.27	28.77	220.55
38	-44.74	18.13	284.53

O tempo total de processamento utilizado pelo método triangular foi de 4,67 segundos.

(7.1.2.1)

Os resultados observados em (7.1.2.1), são rigorosamente iguais aos obtidos pela análise via SAP90, conforme pode ser verificado em A5.2.

7.1.2.2- Método "Sparse"

O método sparse apresentou, para a estrutura proposta em 2.2, numerada e orientada conforme a fig 7.1.2.1, um arquivo de saída com os mesmos resultados apresentados em (7.1.2.1), diferindo apenas na quantidade de posições armazenadas para a matriz de rigidez e no tempo de processamento, que estão indicados abaixo:

Posições armazenadas da matriz de rigidez : 683

Posições armazenadas da matriz [U] : 944

Armazenamento relativo da Matriz [U] : 7,26 %

Tempo total de processamento : 2,47 s

7.1.2.3- Método "Skyline"

O método skyline apresentou, para a estrutura proposta em 2.2, numerada e orientada conforme a fig 7.1.2.1, com arquivo de entrada nomeado como reserra.pas, a exemplo do observado em 7.1.1.2, um arquivo de saída com os mesmos resultados

apresentados em (7.1.2.1), diferindo apenas na quantidade de posições armazenadas para a matriz de rigidez e no tempo de processamento, que estão indicados abaixo:

Posições armazenadas da matriz de rigidez : 945

Posições armazenadas da matriz [U] : 945

Armazenamento relativo da Matriz [U] : 7.27 %

Tempo total de processamento : 2.52 s

7.1.3- Resultados Obtidos Para a Estrutura do Terceiro Exemplo

A estrutura apresentada em 2.3, é uma estrutura fechada, que não permite uma formação compacta de sua banda e teve propositalmente seus nós numerados de forma aleatória, com o intuito de não termos qualquer formação em banda.

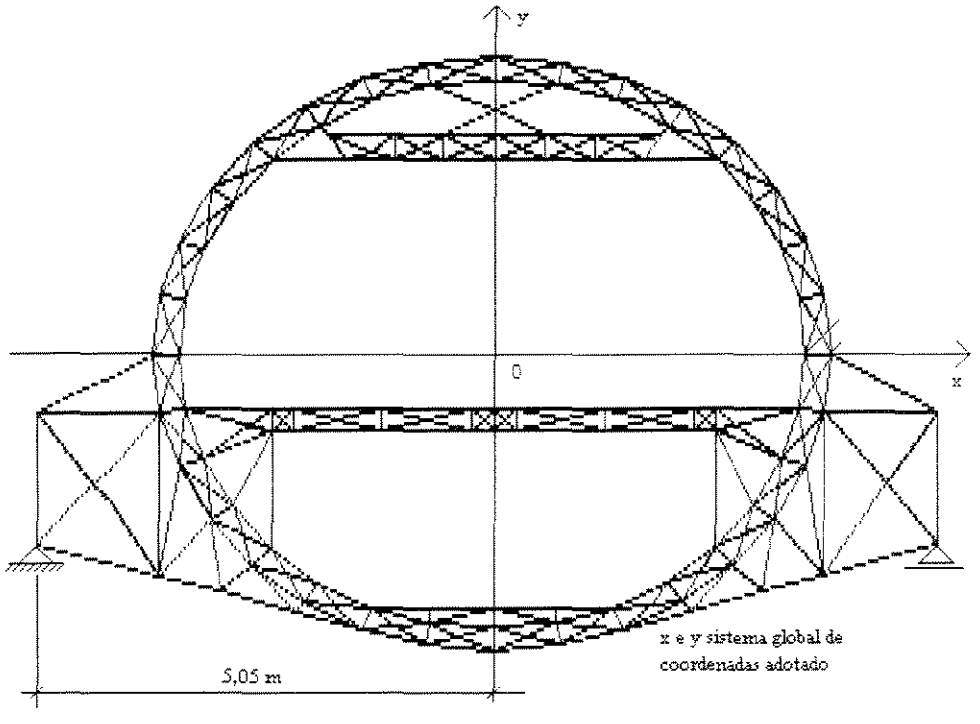

fig 7.1.3.1

7.1.3.1- Método Triangular

O método triangular não conseguiu processar a estrutura proposta em 2.3 no equipamento utilizado e apresentado no mesmo capítulo 2, apresentando excesso de armazenamento de dados no equipamento utilizado. (Heap Overflow)

7.1.3.2- Método "Sparse"

O método sparse apresentou, para a estrutura proposta em 2.3, numerada conforme descrito em 7.1.3 e orientada de acordo com a figura 7.1.3.1, um arquivo de saída conforme apresentado a seguir :

FOI ANALISADA A ESTRUTURA CUJOS DADOS FIGURAM NO ARQUIVO plan.pas

A matriz U armazenou 20879 posicoes.

A matriz de rigidez do caso analisado tem 230.400 posicoes

e portanto o metodo sparse armazenou 9,1 % da referida matriz de rigidez.

OS RESULTADOS ENCONTRADOS FORAM:

DESLOCAMENTOS

NO	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	0.000000	-0.000000	-0.010616
2	-0.002657	-0.014843	-0.008225
3	0.006459	-0.007314	-0.007669
4	0.011596	-0.002580	-0.007871

A planilha completa pode ser vista no anexo 6

157	0.010338	-0.012144	0.008884
158	-0.001215	-0.029387	0.001402
159	-0.002512	-0.025333	0.014227
160	0.005695	-0.019973	-0.015489

ESFORÇOS NAS BARRAS

BARRA	NO INICIAL			NO FINAL		
	AXIAL	NORMAL	MOMENTO	AXIAL	NORMAL	MOMENTO
1	-692.97	6.14	3.23	692.97	-6.14	4.79
2	1116.48	5.13	1.11	-1116.48	-5.13	3.76
3	778.64	-4.29	-4.33	-778.64	4.29	-2.95
4	-674.43	2.33	1.38	674.43	-2.33	1.74

A planilha completa pode ser vista no anexo 6

389	88.62	274.46	31.03	-88.62	271.80	-30.13
390	79.96	271.64	30.08	-79.96	274.62	-31.10
391	607.77	2.96	1.43	-607.77	-2.96	1.52
394	-500.34	-5.75	-1.21	500.34	5.75	-0.92
395	-497.59	-4.67	-0.74	497.59	4.67	-1.14

REAÇÕES NOS APOIOS

NÓ	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	0.00	1888.35	-0.00
41	-0.00	1888.35	0.00

O tempo total de processamento utilizado pelo metodo sparse foi de 44.1 segundos.

(7.1.2.3.1)

A coerência dos resultados pode ser verificada pela simetria da estrutura e do carregamento, perfeitamente repeitada pelos valores obtidos para as reações e facilmente verificável uma vez que a estrutura é isostática externamente.

Os resultados observados, para deslocamentos, têm uma diferença de aproximação, em relação aos obtidos pela análise via SAP90, conforme pode ser verificado no anexo 5 seção 3, na quarta casa decimal.

7.1.2.3- Método "Skyline"

O método skyline apresentou, para a estrutura proposta em 2.3, numerada conforme descrito em 7.1.3, um arquivo de saída com os mesmos resultados obtidos pelo método sparse, diferindo apenas na quantidade de posições armazenadas para a matriz de rigidez e no tempo de processamento, que estão indicados abaixo:

Posições armazenadas da matriz de rigidez : 20.948

Posições armazenadas da matriz [U] : 20.948

Armazenamento relativo da Matriz [U] : 9,09 %

Tempo total de processamento : 49,49 s

7.2- Conclusões

7.2.1- As Bases de Comparação

Tendo em vista a melhor visualização comparativa entre os métodos, foram reunidos alguns resultados em planilhas apresentadas a seguir.

A tabela 7.2.1.1 apresenta as bases de comparação entre os três métodos no processamento da estrutura chamada de Primeiro Exemplo, cujos resultados foram apresentados em 7.1.1.

Estrutura 1	número de elementos da matriz de rigidez	número de elementos armazenados antes da eliminação gaussiano	número de elementos armazenados após a eliminação gaussiana	percentagem de armazengem antes da eliminação gaussiana	percentagem de armazenagem após a eliminação gaussiana	tempo de processamento em segundos
Triangular	225	120	120	53,3	53,3	0.28
Skyline	225	43	43	19,11	19,11	0,21
Sparse	225	21	37	9,3	16,44	0.17

(7.2.1.1)

A tabela 7.2.1.2 apresenta as bases de comparação entre os três métodos no processamento da estrutura chamada de Segundo Caso, cujos resultados foram apresentados em 7.1.2.

Estrutura 2	número de elementos da matriz de rigidez	número de elementos armazenados antes da eliminação gaussiano	número de elementos armazenados após a eliminação gaussiana	percentagem de armazengem antes da eliminação gaussiana	percentagem de armazenagem após a eliminação gaussiana	tempo de processamento em segundos
Triangular	12.996	6.555	6.555	50,44	50,44	4,67
Skyline	12.996	945	945	7,27	7,27	2,52
Sparse	12.996	683	944	5,3	7,26	2,47

(7.2.1.2)

A tabela 7.2.1.3 apresenta as bases de comparação entre os três métodos no processamento da estrutura chamada de Terceiro Caso, cujos resultados foram apresentados em 7.1.3.

Estrutura 3	número de elementos da matriz de rigidez	número de elementos armazenados antes da eliminação gaussiano	número de elementos armazenados após a eliminação gaussiana	percentagem de armazengem antes da eliminação gaussiana	percentagem de armazenagem após a eliminação gaussiana	tempo de processamento em segundos
Triangular	230.400					
Skyline	230.400	20.948	20.948	9,09	9,09	49,49
Sparse	230.400	4.181	20.879	1,8	9,06	44,1

(7.2.1.3)

7.2.2- Comentários Finais

A primeira constatação que fica é a imensa quantidade de posições não significativas geradas pela matriz de rigidez, que torna esse estudo de técnicas de armazenagem, um imperativo, mesmo para equipamentos de grande porte. É bom lembrar que foram analisadas aqui estruturas planas, pois as espaciais quadruplicariam a matriz de rigidez tornando o problema de armazenamento ainda mais dramático. Se for considerado o cálculo estrutural de um edifício de quatro apartamentos por pavimento e vinte pavimentos, tão comum no cenário de nossas cidades, se constatará que facilmente se chegará a 3.000 nós, o que imporá uma matriz com 324.000.000 de posições, exigindo cerca de 2.6 Gb de memória armazenada, inviabilizando totalmente o método para os equipamentos que os escritórios de cálculo normalmente possuem. Armazenando apenas os não-zero se chegará, para esse caso exemplificado, em menos de 260Mb, o que já viabilizará estações do tipo Risc para esse cálculo, ou a partição da estrutura em duas ou tres subestruturas..., enfim, será possível seu tratamento.

A segunda constatação se refere à grande alteração provocada pelo algoritmo gaussiano na esparsidade da matriz de rigidez. Como pode ser observado na análise do terceiro caso pelo método Sparse, após a intervenção do algoritmo, o número de posições armazenadas foi quintuplicado. Ou seja o número de posições zero no interior da banda definida pelo método Skyline se reduz a quantidades quase insignificantes na matriz [U].

A comparação entre os tempos de processamento, que aponta para uma maior eficiência do método Sparse, obriga a comentar um pouco sobre a elaboração dos programas. Quando foi construida a primeira versão do Sparse, a pilha dinamicamente

alocada, como já foi comentado anteriormente, era simplesmente ligada e o tempo gasto para processar a estrutura do caso 2, foi de mais de três horas. Promovendo a dupla ligação entre os elementos da pilha o tempo de processamento caiu para três minutos e armazenando convenientemente alguns endereços ao longo da eliminação gaussiana, com o intuito de reduzir as "viagens" através da estrutura de dados, chegou-se aos pouco mais de 2 segundos observados na versão final. Isso conduz à conclusão de que é uma operação demorada em termos computacionais o manuseio de uma grande estrutura de dados, daí a vantagem do método Sparse sobre o Skyline, pois o segundo inicia a eliminação gaussiana com uma pilha cinco vezes maior.

Para finalizar, pode-se indicar uma pequena vantagem em termos de tempo de processamento, em favor do método Sparse, uma vez que em termos de armazenamento, Skyline e Sparse praticamente se equiparam, e que dotando os programas de ferramentas que permitam a gravação dos elementos da matriz no disco rígido, como se aborda no Anexo 1, essa técnica permitiria a análise em computadores pessoais, da maioria das estruturas de uso corrente.

8- REFERENCIAS BIBLIOGRAFICAS

8.1- Obras Citadas no Texto

[1] - MOREIRA, Domício Falcão. **Análise matricial de estruturas**. Rio de Janeiro, Universidade de São Paulo Livros Técnicos e Científicos, 1977.

[2] - ROTH, P. N. A fast pointer based-solved solver for simultaneous linear equations. **Computer & Structures**. Great Britain, Pergamo, **36**(4): 585-612, 1990.

[3] - GOLUB, Gene H. & VAN LOAN, CHARLES F. **Matrix computations**. Baltimore, The John Hopkins Univresity, 1989.

[4] - DUFF, I. S., ERISMAN, A. M., REID, J. K. **Direct methods for sparse matrices**. Great Britain, Oxford Science Publications, 1986.

[5] - WILSON, Edward L., Habibullah, A. SAP90: A series of computer programs for the static and dinamic finite element analysis of structure, users manual. Bekerley, 1988.

[6] - JONES, Jaqueline A. & HARROW, Keith. **Problem solving with turbo Pascal**. New Jersey, Prentice Hall, 1986.

[7] - BAHTE, K. J. & WILSON, E. L. **Numerical methods in finite element analysis**. Englewood Cliffs, Prentice Hall, 1976.

[8] - WILSON, E. L., BAHTE, K. J. , DOHERTY, W. P. Direct solution of large of linear equations. **Computer & Structures**. Great Britain, 4:363 372, 1974.

[9] - CUTHIL, E. & McKEE, J. Reducing the bandwidth of sparse symetric matrices. **Proceedings 24th National Conference of the Association for Computing Machinery.** New Jersey, Brandon. 157-172, 1969.

[10]- GEORGE, A. **Computer implementation on the finite element method. Ph. D thesis.** California, Department of Computer Science-Stanford University, 1971.

8.2- Obras de Consulta

BAHTE, K. J. & WILSON, E. L. **Numerical methods in finite element analysis.** Englewood Cliffs, Prentice Hall, 1976.

BARRET, Richard, et alii . **Templates for the solution of linear systems**; building blocks for iterative methods. Philadelphia, Society For Industrial And Aplied Mathematics, 1993.

BECKER , Eric B. et alii. **Finite elements an introduction.** Englewood Cliffs, Prentice Hall, 1981

BORLAND Int. Inc. **Borland Pascal manual.** Scotts Valley, 1988.

DEVLOO, P. & HAYES, L. J. **A fast vector algorithm for a matrix-vector multiplication in the finite element method.** Ticom-Texas Institute For Computational Mechanics. Austin. 1985

HUGHES, Thomas J. R. **The finite element method**. Englewood Cliffs, Prentice Hall, 1987.

JAMSA, Kris & NAMEROFF, Seteven. **Turbo Pascal - programmer's library.** Berckeley, Borland Osborne/ McGraw Hill, 1988.

LO, S. H. On bandsolver using skyline storage . **Computer & Structures.** Great Britain, Pergamo, **44**(6): 1187-1196. 1992.

LUO, J. C. Algorithms for reducing the bandwidth and profile of a sparse

matrices. **Computer & Structures**, Great Britain, **44**(3): 535-548. 1992.

MACKAY, D. R. et alii. An implementation of a generalized sparse/profile finite element solution method. **Computer & Structures**, Great Britain, **41**(4): 723-737.

ANEXO 1
Outros Métodos

1- ARMAZENAGEM POR BLOCOS

1.1- O Conceito

Trata-se de uma técnica que procura viabilizar o tratamento de estruturas de porte muito grande, fazendo uso de gravação externa no disco rígido.

O método estabelece um número de linhas da matriz de rigidez que compõe um bloco, processa a construção desse bloco dentro da memória RAM, efetua a eliminação gaussiana e armazena o bloco no disco rígido. Constrói um novo bloco com as linhas seguintes e repete a operação até que toda a matriz esteja, em sua primeira forma eliminada, armazenada no disco rígido.

O método pode ainda se valer do processo de armazenagem abordado no corpo principal desse trabalho, ou seja triangular ou skyline, fazendo com que o bloco seja armazenado dinamicamente, por um dos processos abordados, na memória RAM, tratado pelo algoritmo gaussiano e depois armazenado no disco rígido.

Por essa via, a capacidade de processamento fica quase ilimitada, tornando, no entanto, como poderemos ver adiante, extremamente demorada sua operação.

1.2- Descrição das Etapas de Programação

Já que se pretende enfrentar gravações e buscas em disco rígido, que são operações morosas, seria de boa tecnica que o programa se iniciasse com uma verificação do tamanho da estrutura, dirigindo para uma rotina alternativa, como aquelas apresentadas no corpo principal deste trabalho, as estruturas menores, capazes de serem tratadas dentro da memória RAM. Constatada a efetiva necessidade de aplicação do

método, ou seja, o conjunto de dados a serem armazenados não cabe em RAM, dar-se-ia início, após os procedimentos iniciais de leitura dos dados geométricos, físicos e de carregamento da estrutura em análise, à construção do primeiro bloco dentro de uma pilha alocada dinamicamente, tendo em vista a impossibilidade de se guardar dentro do segmento de memória os coeficientes relativos a mais de um nó, ou até mesmo de mais de uma linha, de estruturas que exijam essa rota. Após a construção do bloco, aplicar-se-ia a eliminação gaussiana sobre o mesmo, procedendo-se seu armazenamento numa segunda pilha, agora pelo método triangular ou skyline, apagando-se em seguida a pilha de dados do bloco original. Na seqüência, far-se-ia a construção do segundo bloco, sobre o qual se procederia a eliminação gaussiana, guardando-se os resultados por sobre a pilha já existente do primeiro e seguindo o mesmo procedimento, triangular ou skyline.

A operação se repetiria até que a pilha obtida da primeira eliminação atingisse um determinado número de elementos, previamente determinado. Encontrado esse limite proceder-se-ia à armazenagem da pilha no disco rígido, tomando-se o cuidado de se deixar a primeira linha em RAM, pois será necessária para o prosseguimento da primeira eliminação nos blocos seguintes.

Terminada a construção da matriz de rigidez, já tendo inclusive sofrido a primeira eliminação, far-se-ia à recuperação do disco rígido para RAM, de um primeiro bloco colocado sob a forma de pilha alocada dinamicamente. Sobre essa nova pilha se aplicaria a segunda eliminação do processo de Gauss, fazendo sua devolução para o disco rígido, tomando um novo bloco para prosseguimento da segunda eliminação, e assim por diante, até que se completasse a matriz U.

Obtida a matriz U, o programa partiria para a finalização já apresentada anteriormente.

Não indicamos o método Sparse, na construção da pilha eliminada, pois o mesmo, como pudemos verificar na seção correspondente, altera ao longo da eliminação gaussiana, o tamanho da pilha, criando posições intermediárias devidas ao fato de que ele cria posições significativas que antes, por serem zero, não haviam sido objeto de armazenagem. Esse reordenamento de dados no disco rígido nos parece de difícil operacionalização.

Não é difícil perceber que o método torna extremamente moroso o processo, uma vez que as tecnicas utilizadas para acelerar a varredura das pilhas que em RAM já são demoradas, aqui não são possíveis aliada ao fato da interferencia do processo mecânico de leitura do disco rígido. Talvez valesse mesmo a pena partir a estrutura em trechos e calcula-la por partes.

2-MÉTODOS ITERATIVOS

São métodos de cálculo de sistemas lineares de equações sem a construção da matriz dos coeficientes, procurando a solução através de processos iterativos, ou seja,de aproximações sucessivas dos valores das incógnitas. De um modo geral eles partem daescolha de um conjunto de valores pré escolhidos como primeira aproximação das variáveis, constroem a primeira linha da matriz dos coeficientes e aplicam um algoritmo que fazendo uso do valor atribuído à primeira variável calcula um novo valor para ela

mais próximo do valor real. A seguir, constroem a segunda linha da matriz e utilizando-se do valor atribuído à segunda variável, e o valor calculado para a primeira,entram novamente no algoritmo que calcula um novo valor para a segunda variável, e assim por diante, até que se aplique a mesma operação sobre a última linha, obtendo-se assim um novo valor para a última incógnita. Em cada incógnita é feita a verificação do grau de aproximação em função de um erro admitido previamente como aceitável, e em função desse erro o processo é repetido quantas vezes for necessário.Como primeira aproximação pode ser usado zero para todas as variáveis, uma vez que os valores procurados de um modo geral devem ser menores do que zero, pois as incógnitas são deslocamentos e não se espera para eles valores maiores.

Pela própria descrição acima pode-se deduzir que embora o problema de armazenamento esteja solucionado, uma vez que a matriz de rigidez não é construida em momento algum, o processo se torna tão moroso que o inviabiliza como se verá adiante.

Há, ainda, a questão da convergência que pode não ocorrer dependendo da configuração da matriz dos coeficientes, pois como condição básica para a convergência é preciso que o coeficiente da diagonal seja maior que os demais coeficientes da mesma linha. No caso estrutural parece que a convergência deve acontecer sempre.

Dentre os diversos métodos apresentados por Barret et al[14], podem ser destacados os seguintes:

- **Jacobi** ;
- **Gauss-Seidel** ;
- **Successive Overrelaxation** ;
- **Symmetric Successive Overrelaxation** ;

Nesse trabalho escolheu-se o Método de Gauss-Seidel para ser objeto de construção de uma versão do programa básico, para que fosse possível avaliar a oportunidade do processo.

2.1- O Programa Gauss-Seidel

Construir linha a linha a matriz dos coeficientes já é uma técnica utilizada neste trabalho restando, portanto, criar um laço que execute o algoritmo, apresentado por Barret et. al, op. cit,p. 10, e reproduzido aqui em A4-2.1, verificando a convergência e o grau de aproximação em relação ao desejado e fazendo-o voltar a executar o algoritmo até que a aproximação procurada se verifique.

for k = 1 ,2,.....
  for i = 1,2,.....,n
    $\sigma = 0$
    for j = 1,2,.....,i-1 (A4-2.1)
      $\sigma = \sigma + a_{ij} x_j^{(k)}$
    end
    for j = i+1,.....,n
      $\sigma = \sigma + a_{ij} x_j^{(k-1)}$
  end
    verifica a convergencia, continua se necessário
end

A operação do programa construido correspondeu à expectativa inicial, pois para a execução da estrutura do primeiro caso, aquele pequeno pórtico de 4 barras, foram necessárias cerca de 5000 execuções do loop para se chegar à precisão da quinta casa decimal, tendo sido dispendida mais de uma hora para seu processamento. Como a oportunidade do método se deve ao fato de deixar o equipamento apto a analisar estruturas de porte ilimitado o tempo de processamento o inviabilisa.

ANEXO 2
Listagem do Programa Versão Triangular

```
PROGRAM PORTICO_PLANO:     (*armazena metade da matriz global e aloca as variaveis
                             dinamicamente*)
Uses CRT.DOS;
type
   matriz=array[1..6,1..6] of double;
   tarray=array[1..4] of double;
   vetors=array[1..6] of double;
   vetor =array[1..2000] of double;
   arquivo=text;
   AEptr=^AErec;
   AErec=record
         Eval:double;
         pos:longint;
         next:AEptr;
         ant:AEptr;
   end;
   kbalptr=^krec;
   krec=record
         kval:double;
         ordem:longint;
         ant:kbalptr;
         next:kbalptr;
   end;
   barptr=^barrec;
   barrec=record
          noinicio:longint;
          nofim:longint;
          minercia:double;
          modelast:double;
          area:double;
          comp:double;
          cosx:double;
          cosy:double;
          cc:tarray;   (*carga concentrada*)
          di:tarray;   (*distancia do no inicio ao ponto de
                     aplicacao*)
          df:tarray;   (*dist. do no fim ao ponto de
                     aplicacao*)
          ud:tarray;   (*carga uniformemnte distribuida*)
          uddf:tarray; (*dist. do no fim ao ponto medio
                     da carga*)
          uddi:tarray; (*dist. do no inicio ao ponto
                     medio da carga*)
          tam:tarray;  (*comprimento da carga uniformemente
                      dist.*)
          uv:double;     (*valor maior da carga
                     uniformemente variada*)
          uvni:double;  (*valor da uv no no inicio*)
          axc:tarray;  (*carga axial aplicada na barra*)
          axcdi:tarray:(*distancia do no inicio ao ponto de
                     aplicacao*)
          axcdf:tarray:(*distancia do no fim ao ponto de
                      aplicacao
                   da carga*)
          axud:tarray; (**)
```

```
            atam:tarray;
            axudi:tarray;
            axudf:tarray;
            axuv:double;
            axuvni:double;
            mab:tarray;
            mabdi:tarray;
            mabdf:tarray;
            norin:double;
            norfim:double;
            axin:double;
            axfim:double;
            momin:double;
            mofim:double;
            ant:barptr;
            next:barptr;
            num:longint;
    end;
    noptr=^norec;
    norec=record
          nu:longint;
          absno:double;
          ordno:double;
          norno:double;
          axno:double;
          mono:double;
          restri:integer;
          dax:double;
          dnor:double;
          rot:double;
          reax:double;
          renor:double;
          remon:double;
          ant:noptr;
          next:noptr;
    end;
     guarda=array[1..20] of noptr;
     guarba=array[1..20] of barptr;

var
AE.Topae.lastae:AEptr;
proxk,auxk,kglobal,topk,lastk:kbalptr;
primbar.lastbarra.barra.topbar:barptr;
noi.lastno.no.topno:noptr;
A.B.G.I.K.L.N.X.Y.W.Z.NI.NF.AAA.GG.OPTION.busk.total:longint;
C.D.E.F.H.J.M.O.P.Q.R.S.T.U.CONC.COMPR.COSENO.SENO.UVAR.NIN.NFIM.MIN.MFIM.
    AXIC.AXICDI.AXVAR.AXVARNI.UNI.AIN.AFIM.DIST.ud.tam.uddi.
    momb.mombdi.val1:double;
t1.t2.sec. hund :real;
tempo:real;
Hour. Minute. Second. Sec100: word;
gno:guarda;
gba:guarba;
CC.AA.BB.MBETA.MRIGB: MATRIZ;
ELE:VETOR;
CA:VETORS;
ARQE.ARQS:ARQUIVO;
NOMEARQE.NOMEARQS:STRING;
```

```
procedure rest1 ;
var
x,y,z:integer;
begin
  y:=(3*w-3)*3*N-(3*w-3)*(3*w-2) div 2 +3*w-2;
  z:=(3*w-3)*3*N-(3*w-3)*(3*w-2) div 2 +3*N;
  kglobal:=topk;
  while kglobal^.ordem>z+1 do
  kglobal:=kglobal^.next;
  for x:=z downto y+1 do
  begin

    kglobal:=kglobal^.next;
    kglobal^.kval:=0;
  end;
  proxk:=kglobal^.next;
  proxk^.kval:=1;
  for x:=3*w-3 downto 1 do
  begin
    y:=(x-1)*3*N-x*(x-1) div 2 +3*w-2;
    while kglobal^.ordem>y do
    kglobal:=kglobal^.next;
    kglobal^.kval:=0;
  end;
end;
procedure rest2;
var
x,y,z:integer;
begin
  y:=(3*w-2)*3*N-(3*w-2)*(3*w-1) div 2 +3*w-1;
  z:=(3*w-2)*3*N-(3*w-2)*(3*w-1) div 2 +3*N;
  kglobal:=topk;
  while kglobal^.ordem>z+1 do
  kglobal:=kglobal^.next;
  for x:=z downto y+1 do
  begin
    kglobal:=kglobal^.next;
    kglobal^.kval:=0;
  end;
  proxk:=kglobal^.next;
  proxk^.kval:=1;
  for x:=3*w-2 downto 1 do
  begin
    y:=(x-1)*3*N-x*(x-1) div 2 +3*w-1;
    while kglobal^.ordem>y do
    kglobal:=kglobal^.next;
    kglobal^.kval:=0;
  end;
end;
procedure rest3;
var
x,y,z:integer;
begin
  y:=(3*w-1)*3*N-(3*w-1)*(3*w) div 2 +3*w;
  z:=(3*w-1)*3*N-(3*w-1)*(3*w) div 2 +3*N;
  kglobal:=topk;
  while kglobal^.ordem>z+1 do
```

```
    kglobal:=kglobal^.next;
    for x:=z downto y+1 do
    begin
      kglobal:=kglobal^.next;
      kglobal^.kval:=0;
    end;
    proxk:=kglobal^.next;
    proxk^.kval:=1;
    for x:=3*w-1 downto 1 do
    begin
      y:=(x-1)*3*N-x*(x-1) div 2 +3*w;
      while kglobal^.ordem>y do
      kglobal:=kglobal^.next;
      kglobal^.kval:=0;
    end;
end;
procedure remete;
begin
    nin:=barra^.norin;
    nfim:=barra^.norfim;
    ain:=barra^.axin;
    afim:=barra^.axfim;
    min:=barra^.momin;
    mfim:=barra^.mofim;
    compr:=barra^.comp;
end;
procedure recebe;
begin
    barra^.norin:=nin;
    barra^.norfim:=nfim;
    barra^.axin:=ain;
    barra^.axfim:=afim;
    barra^.momin:=min;
    barra^.mofim:=mfim;
end;
procedure Limpa;
begin
     lastk:=topk^.next;
     kglobal:=topk;
     while kglobal^.next<>nil do
     begin
       dispose(kglobal);
       kglobal:=lastk;
       lastk:=kglobal^.next;
     end;
     barra:=topbar;
     lastbarra:=topbar^.next;
     while barra^.next<>nil do
     begin
       dispose(barra);
       barra:=lastbarra;
       lastbarra:=barra^.next;
     end;
     no:=topno;
     lastno:=no^.next;
     while no^.next<>nil do
     begin
       dispose(no);
```

```
      no:=lastno;
      lastno:=no^.next;
    end;
end;

procedure matrizbeta(t,u:double; Var mbeta:matriz);
begin
  MBETA[1,1]:=T;
  MBETA[1,2]:=U;
  MBETA[1,3]:=0;
  MBETA[1,4]:=0;
  MBETA[1,5]:=0;
  MBETA[1,6]:=0;
  MBETA[2,1]:=-U;
  MBETA[2,2]:=T;
  MBETA[2,3]:=0;
  MBETA[2,4]:=0;
  MBETA[2,5]:=0;
  MBETA[2,6]:=0;
  MBETA[3,1]:=0;
  MBETA[3,2]:=0;
  MBETA[3,3]:=1;
  MBETA[3,4]:=0;
  MBETA[3,5]:=0;
  MBETA[3,6]:=0;
  MBETA[4,1]:=0;
  MBETA[4,2]:=0;
  MBETA[4,3]:=0;
  MBETA[4,4]:=T;
  MBETA[4,5]:=U;
  MBETA[4,6]:=0;
  MBETA[5,1]:=0;
  MBETA[5,2]:=0;
  MBETA[5,3]:=0;
  MBETA[5,4]:=-U;
  MBETA[5,5]:=T;
  MBETA[5,6]:=0;
  MBETA[6,1]:=0;
  MBETA[6,2]:=0;
  MBETA[6,3]:=0;
  MBETA[6,4]:=0;
  MBETA[6,5]:=0;
  MBETA[6,6]:=1;
END;

procedure matrizrig(C,J,S,P:double;Var mrigb:matriz);
    (*C=COMP. J=MOM. DE INERCIA. S=AREA. P=MODULO DE EL.*)
begin
  MRIGB[1,1]:=P*S/C;
  MRIGB[1,2]:=0;
  MRIGB[1,3]:=0;
  MRIGB[1,4]:=-P*S/C;
  MRIGB[1,5]:=0;
  MRIGB[1,6]:=0;
  MRIGB[2,1]:=0;
  MRIGB[2,2]:=12*P*J/(SQR(C)*C);
  MRIGB[2,3]:=6*P*J/(SQR(C));
  MRIGB[2,4]:=0;
```

```
  MRIGB[2,5]:=-12*P*J/(SQR(C)*C);
  MRIGB[2,6]:=6*P*J/(SQR(C));
  MRIGB[3,1]:=0;
  MRIGB[3,2]:=6*P*J/(SQR(C));
  MRIGB[3,3]:=4*P*J/C;
  MRIGB[3,4]:=0;
  MRIGB[3,5]:=-6*P*J/(SQR(C));
  MRIGB[3,6]:=2*P*J/C;;
  MRIGB[4,1]:=-P*S/C;
  MRIGB[4,2]:=0;
  MRIGB[4,3]:=0;
  MRIGB[4,4]:=P*S/C;
  MRIGB[4,5]:=0;
  MRIGB[4,6]:=0;
  MRIGB[5,1]:=0;
  MRIGB[5,2]:=-12*P*J/(SQR(C)*C);
  MRIGB[5,3]:=-6*P*J/(SQR(C));
  MRIGB[5,4]:=0;
  MRIGB[5,5]:=12*P*J/(SQR(C)*C);
  MRIGB[5,6]:=-6*P*J/(SQR(C));
  MRIGB[6,1]:=0;
  MRIGB[6,2]:=6*P*J/(SQR(C)) ;
  MRIGB[6,3]:=2*P*J/C;
  MRIGB[6,4]:=0;
  MRIGB[6,5]:=-6*P*J/(SQR(C)) ;
  MRIGB[6,6]:=4*P*J/C;
end;

procedure produmax(aa,bb:matriz; Var cc:matriz) ;
Var
x,y,k:integer;

begin
  for x:=1 to 6 do
  for y:=1 to 6 do
  begin
    cc[x,y]:=0;
     for k:=1 to 6 do
    begin
      cc[x,y]:=cc[x,y]+aa[x,k]*bb[k,y];
    end;
  end;
end;

procedure Tripno(I:longint; Var no:noptr);
begin
  while I<no^.nu do
  no:=no^.next;
  while I>no^.nu do
  no:=no^.ant;
end;
procedure assemhalf(n,B,G:integer; barra,topbar:barptr; no,topno:noptr;
                    Kglobal,topk:kbalptr);

Var
x,k,y,ni,nf,ww,xx,yy,v,I,JJ:longint;
cc,bb,aa,mbeta,mrigb:matriz;
t,u,p,s,j,C,r,d:real;
```

```
Procedure vecalc;
begin
  XX:=((I-1)*3*N-I*(I-1) DIV 2) +JJ;
  If XX<kglobal^.ordem then
  begin
    while xx<kglobal^.ordem do
    kglobal:=kglobal^.next;
  end
  else
  begin
    while xx>kglobal^.ordem do
    kglobal:=kglobal^.ant;
  end;
end;
BEGIN
  Tripbar (G,barra);
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T,U,MBETA);
  P:=BARRA^.MODELAST;
  J:=BARRA^.MINERCIA;
  S:=BARRA^.AREA;
  C:=BARRA^.COMP;
  MATRIZRIG(C,J,S,P,MRIGB);
  FOR X:=1 TO 6 DO
  FOR K:=1 TO 6 DO
  AA[X,K]:=MBETA[K,X];
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BB[X,Y]:=MRIGB[X,Y];
  PRODUMAX(AA,BB,CC);
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BEGIN
    AA[X,Y]:=CC[X,Y];
    BB[X,Y]:=MBETA[X,Y];
  END;
  PRODUMAX(AA,BB,CC);
  NI:=BARRA^.NOINICIO;
  NF:=BARRA^.NOFIM;
  For Y:=2 downto 0 do
  For x:=2 downto 0 do
  begin
    I:=ni*3-X;
    JJ:=ni*3-y;
    If JJ>=I then
    begin
      vecalc;
      kglobal^.kval:=kglobal^.kval+cc[3-x,3-y];
    end;
    I:=ni*3-X;
    JJ:=nf*3-y;
    If JJ>=I then
    begin
      vecalc;
      kglobal^.kval:=kglobal^.kval+cc[3-x,6-y];
    end;
    I:=nf*3-X;
```

```
    JJ:=ni*3-y;
    If JJ>=I then
    begin
      vecalc;
      kglobal^.kval:=kglobal^.kval+cc[6-x,3-y];
    end;
    I:=nf*3-X;
    JJ:=nf*3-y;
    If JJ>=I then
    begin
      vecalc;
      kglobal^.kval:=kglobal^.kval+cc[6-x,6-y];
    end;
  end;
end;

procedure assemblagem(n,B,G:integer; barra.topbar:barptr; no.topno:noptr;
                      Kglobal.topk:kbalptr);

Var
x,k,y,ni,nf,ww,xx,yy,v,I:integer;
cc,bb,aa,mbeta,mrigb:matriz;
t,u,p,s,j,C,r,d:real;
Procedure vecalc;
begin
  v:=(ww-1)*3*n-(ww-1)*ww div 2+yy;
end;

BEGIN
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T,U,MBETA);
  P:=BARRA^.MODELAST;
  J:=BARRA^.MINERCIA;
  S:=BARRA^.AREA;
  C:=BARRA^.COMP;
  MATRIZRIG(C,J,S,P,MRIGB);
  FOR X:=1 TO 6 DO
  FOR K:=1 TO 6 DO
  AA[X,K]:=MBETA[K,X];
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BB[X,Y]:=MRIGB[X,Y];
  PRODUMAX(AA,BB,CC);
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BEGIN
    AA[X,Y]:=CC[X,Y];
    BB[X,Y]:=MBETA[X,Y];
  END;
  PRODUMAX(AA,BB,CC);
  NI:=BARRA^.NOINICIO;
  NF:=BARRA^.NOFIM;
  IF NI<NF THEN
  BEGIN
    ww:=NI*3-2;
    yy:=ww;
    vecalc;
```

```
kglobal:=topk;
while kglobal^.ordem>v+3 do
kglobal:=kglobal^.next;
for xx:=3 downto 1 do
begin
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[1,xx];
end;
ww:=NI*3-1;
yy:=ww;
vecalc;
kglobal:=topk;
while kglobal^.ordem>v+2 do
kglobal:=kglobal^.next;
for xx:=3 downto 2 do
begin
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[2,xx];
end;
ww:=NI*3;
yy:=ww;
vecalc;
kglobal:=topk;
while kglobal^.ordem>v do
kglobal:=kglobal^.next;
kglobal^.kval:=kglobal^.kval+cc[3,3];
ww:=NI*3-2;
yy:=NF*3-2;
vecalc;
kglobal:=topk;
while kglobal^.ordem>v+3 do
kglobal:=kglobal^.next;
for xx:=6 downto 4 do
begin
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[1,xx];
end;
ww:=NI*3-1;
yy:=NF*3-2;
vecalc;
kglobal:=topk;
while kglobal^.ordem>v+3 do
kglobal:=kglobal^.next;
for xx:=6 downto 4 do
begin
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[2,xx];
end;
ww:=NI*3;
yy:=NF*3-2;
vecalc;
kglobal:=topk;
while kglobal^.ordem>v+3 do
kglobal:=kglobal^.next;
for xx:=6 downto 4 do
begin
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[3,xx];
```

```
  end;
  ww:=NF*3-2;
  yy:=ww;
  vecalc;
  kglobal:=topk;
  while kglobal^.ordem>v+3 do
  kglobal:=kglobal^.next;
  for xx:=6 downto 4 do
  begin
    kglobal:=kglobal^.next;
    kglobal^.kval:=kglobal^.kval+cc[4.xx];
  end;
  ww:=NF*3-1;
  yy:=ww;
  vecalc;
  kglobal:=topk;
  while kglobal^.ordem>v+1 do
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[5.6];
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[5.5];
  ww:=NF*3;
  yy:=ww;
  vecalc;
  kglobal:=topk;
  while kglobal^.ordem>v do
  kglobal:=kglobal^.next;
  kglobal^.kval:=kglobal^.kval+cc[6.6];
END
ELSE
BEGIN
  ww:=nf*3-2;
  yy:=ww;
  vecalc;
  kglobal:=topk;
  while kglobal^.ordem>v+3 do
  kglobal:=kglobal^.next;
  for xx:=6 downto 4 do
  begin
    kglobal:=kglobal^.next;
    kglobal^.kval:=kglobal^.kval+cc[4.xx];
  end;
  ww:=NF*3-2;
  yy:=NI*3-2;
  vecalc;
  kglobal:=topk;
  while kglobal^.ordem>v+3 do
  kglobal:=kglobal^.next;
  for xx:=3 downto 1 do
  begin
    kglobal:=kglobal^.next;
    kglobal^.kval:=kglobal^.kval+cc[4.xx];
  end;
  ww:=NF*3-1;
  yy:=ww;
  vecalc;
  kglobal:=topk;
  while kglobal^.ordem>v do
```

```
    kglobal:=kglobal^.next;
    kglobal^.kval:=kglobal^.kval+cc[5,6];
    proxk:=kglobal^.next;
    proxk^.kval:=proxk^.kval+cc[5,5];
    ww:=NF*3-1;
    yy:=NI*3-2;
    vecalc;
    kglobal:=topk;
    while kglobal^.ordem>v+3 do
    kglobal:=kglobal^.next;
    for xx:=3 downto 1 do
    begin
      kglobal:=kglobal^.next;
      kglobal^.kval:=kglobal^.kval+cc[5,xx];
    end;
    ww:=NF*3;
    yy:=ww;
    vecalc;
    kglobal:=topk;
    while kglobal^.ordem>v+4 do
    kglobal:=kglobal^.next;
    kglobal^.kval:=kglobal^.kval+cc[6,6];
    for xx:=3 downto 1 do
    begin
      kglobal:=kglobal^.next;
      kglobal^.kval:=kglobal^.kval+cc[6,xx];
    end;
    ww:=NI*3-2;
    yy:=ww;
    vecalc;
    kglobal:=topk;
    while kglobal^.ordem>v+3 do
    kglobal:=kglobal^.next;
    for xx:=3 downto 1 do
    begin
      kglobal:=kglobal^.next;
      kglobal^.kval:=kglobal^.kval+cc[1,xx];
    end;
    ww:=NI*3-1;
    yy:=ww;
    vecalc;
    kglobal:=topk;
    while kglobal^.ordem>v+2 do
    kglobal:=kglobal^.next;
    for xx:=3 downto 1 do
    begin
      kglobal:=kglobal^.next;
      kglobal^.kval:=kglobal^.kval+cc[2,xx];
    end;
    ww:=NI*3;
    yy:=ww;
    vecalc;
    kglobal:=topk;
    while kglobal^.ordem>v do
    kglobal:=kglobal^.next;
    kglobal^.kval:=kglobal^.kval+CC[3,3];
  end;
end;
```

```
procedure ENGASTUV(OPTION,E,F,COMPR,COSENO,SENO,UVAR,UNI:double;
                   Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);

begin

  IF OPTION=0 THEN
  BEGIN                (*SITEMA GLOBAL DE COORD.*)
    IF UNI=0 THEN
    BEGIN
      NIN:=NIN-(3/20)*E*COMPR*COSENO;
      NFIM:=NFIM-(7/20)*E*COMPR*COSENO;
      AIN:=AIN+(F*COMPR*SENO/2)*(COMPR*SENO/3)/COMPR;
      AFIM:=AFIM+(F*COMPR*SENO)*(COMPR*SENO/3)/COMPR;
      MIN:=MIN-E*SQR(COMPR*COSENO)/30;
      MFIM:=MFIM+E*SQR(COMPR*COSENO)/20;
    END
    ELSE
    BEGIN
      NIN:=NIN-(7/20)*E*COMPR*COSENO;
      NFIM:=NFIM-(3/20)*E*COMPR*COSENO;
      AIN:=AIN+(F*COMPR*SENO)*(COMPR*SENO/3)/COMPR;
      AFIM:=AFIM+(F*COMPR*SENO/2)*(COMPR*SENO/3)/COMPR;
      MIN:=MIN-E*SQR(COMPR*COSENO)/20;
      MFIM:=MFIM+E*SQR(COMPR*COSENO)/30;
    END;
  END
  ELSE
  BEGIN          (*SISTEMA LOCAL*)
    IF UNI=0 THEN
    BEGIN
      NIN:=NIN-(3/20)*UVAR*COMPR;
      NFIM:=NFIM-(7/20)*UVAR*COMPR;
      MIN:=MIN-UVAR*SQR(COMPR)/30;
      MFIM:=MFIM+UVAR*SQR(COMPR)/20;
    END
    ELSE
    BEGIN
      NIN:=NIN-(7/20)*UVAR*COMPR;
      NFIM:=NFIM-(3/20)*UVAR*COMPR;
      MIN:=MIN-UVAR*SQR(COMPR)/20;
      MFIM:=MFIM+UVAR*SQR(COMPR)/30;
     END;
  END;
END;

procedure ENGASTCC(OPTION,E,F,COMPR,CONC,DIST:double;
                   Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
var
cont:longint;
begin
  if option=0 then
  begin
    nin:=nin-e*(sqr((compr-dist)/compr)*(3-2*(compr-dist)/compr));
     nfim:=nfim-e*(sqr(dist/compr)*(3-2*dist/compr));
     min:=min-e*(dist*sqr((compr-dist)/compr));
     mfim:=mfim+e*(compr-dist)*sqr(dist/compr);
     ain:=ain-f*(compr-dist)/compr;
```

```
    afim:=afim-f*dist/compr;
  end
  else
  begin
 nin:=nin-conc*(sqr((compr-dist)/compr)*(3-2*(compr-dist)/compr));
    nfim:=nfim-conc*(sqr(dist/compr)*(3-2*dist/compr));
    min:=min-conc*(dist*sqr((compr-dist)/compr));
    mfim:=mfim+conc*(compr-dist)*sqr(dist/compr);
  end;

end;

procedure ENGASTUD(OPTION,COMPR,UD,TAM,UDDI,COSENO,SENO:double;
            Var E,F,M,H,NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
begin
  if option=0 then
  begin
    E:=ud*tam*sqr(coseno);
    F:=ud*tam*coseno*seno;
    M:=uddi/compr;
    H:=sqr(tam/(2*compr));
    NIN:=NIN-E*(1-3*sqr(M)-H+2*M*(sqr(M)+H));
    NFIM:=NFIM-E*(3*sqr(M)+H-2*M*(sqr(M)+H));
    MIN:=MIN-E*(UDDI*sqr((COMPR-UDDI)/COMPR)-H*(3*(COMPR-UDDI)-
       COMPR)/3);
    MFIM:=MFIM+E*((COMPR-UDDI)*sqr(M)-H*(3*UDDI-COMPR)/3);
    AIN:=AIN-F*(COMPR-UDDI)/COMPR;
    AFIM:=AFIM-F*UDDI/COMPR;
  end
  else
  begin
    E:=ud*tam;
    M:=uddi/compr;
    H:=sqr(tam/(2*compr));
    NIN:=NIN-E*(1-3*sqr(M)-H+2*M*(sqr(M)+H));
    NFIM:=NFIM-E*(3*sqr(M)+H-2*M*(sqr(M)+H));
    MIN:=MIN-E*(UDDI*sqr((COMPR-UDDI)/COMPR)-H*(3*(COMPR-UDDI)-
       COMPR)/3);
    MFIM:=MFIM+E*((COMPR-UDDI)*sqr(M)-H*(3*UDDI-COMPR)/3);
  end;
end;

PROCEDURE ENGASTAXUV(AXVAR,AXVARNI,COMPR:double;
            Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
Begin
                (*NAO HA OPCAO PARA O SISTEMA GLOBAL*)
  IF AXVARNI=0 THEN
  BEGIN
    AIN:=AIN+AXVAR*(COMPR/3)/COMPR;
    AFIM:=AFIM+AXVAR*(COMPR*2/3)/COMPR;
    END
    ELSE
    BEGIN
      AIN:=AIN+AXVAR*(COMPR*2/3)/COMPR;
      AFIM:=AFIM+AXVAR*(COMPR/3)/COMPR;
    END;
  END;
```

```
PROCEDURE ENGASTAXC(OPTION:longint; AXIC,AXICDI,COMPR,
                    COSENO,SENO:double;
                   Var E,F,NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
BEGIN
  IF OPTION=0 THEN
  BEGIN       (* OSISTEMA ESTA NAS COORDENADAS GLOBAIS*)
             (*AXIC=CARGA CONCENTRADA NA COORDENADA GLOBAL 1*)
    F:=AXIC*COSENO;  (*COMPONENTE NA COORDENADA LOCAL 1*)
    E:=AXIC*SENO;    (*COMPONENTE NA COORDENADA LOCAL 2*)
    NIN:=NIN-
     E*(SQR((COMPR-AXICDI)/COMPR)*(3-2*((COMPR-AXICDI)/COMPR)));
    NFIM:=NFIM-E*(SQR(AXICDI/COMPR))*(3-2*(AXICDI/COMPR));
    MIN:=MIN-E*(AXICDI*SQR((COMPR-AXICDI)/COMPR));
    MFIM:=MFIM+E*((COMPR-AXICDI)*SQR(AXICDI/COMPR));
    AIN:=AIN-F*(COMPR-AXICDI)/COMPR;
    AFIM:=AFIM-F*AXICDI/COMPR;
  END
  ELSE
  BEGIN
    AIN:=AIN-AXIC*(COMPR-AXICDI)/COMPR;
    AFIM:=AFIM-AXIC*AXICDI/COMPR;
  END;
END;

Procedure ENGASTMB(MOMB,MOMBDI,COMPR:double;
                   Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
Begin
  nin:=nin-momb*6*mombdi*(compr-mombdi)/(compr*sqr(compr));
  nfim:=nfim+momb*6*mombdi*(compr-mombdi)/(compr*sqr(compr));
  min:=min-momb*(compr-mombdi)*(2-3*(compr-mombdi)/compr)/compr;
  mfim:=mfim-momb*mombdi*(2-3*mombdi/compr)/compr;
end;

Procedure Cabenos;
Var
R:real;
y:integer;
begin
  writeln(arqs,
'FOI ANALISADA  A ESTRUTURA CUJOS DADOS FIGURAM NO ARQUIVO '.
       NOMEARQE,' ');
  writeln(arqs);
  Y:=sqr(3*N);
  R:= ((3*N-1)*3*N-(3*N*(3*N-1) DIV 2) +3*N)/Y*100;
  writeln(arqs.'A matriz de rigidez do caso analisado tem '.Y.' posicoes');
  writeln(arqs.'  e portanto o metodo triangular armazenou '.R:5:2.' % da referida matriz de rigidez.');
  writeln(arqs.'OS RESULTADOS ENCONTRADOS FORAM:');
  writeln(arqs);
  writeln(arqs);
  writeln(arqs.'                          DESLOCAMENTOS');
  writeln(arqs);
  write(arqs.'  NO    COORDENADA 1        COORDENADA 2');
  writeln(arqs.'          COORDENADA 3');
end;

Procedure Mostradesloc(C,D,E:double; g:longint; Var X:longint);
Var
A,y:integer;
```

```
begin
  write(arqs,' ',g:3,'    ',C:9:6,'          ',D:9:6);
  writeln(arqs,'          ',E:9:6);
end;

Procedure Roda  (B:longint; Topbar,barra:barptr; mbeta:matriz);
Var
T,U:double;
G:longint;
CA:Array[1..6] of double;

begin
   FOR G:=1 TO B DO
   BEGIN
     Tripbar(G,barra);
     T:=BARRA^.COSX;
     U:=BARRA^.COSY;
     matrizbeta(T,U,mbeta);
     CA[1]:=BARRA^.AXIN;
     CA[2]:=BARRA^.NORIN;
     CA[3]:=BARRA^.MOMIN;
     CA[4]:=BARRA^.AXFIM;
     CA[5]:=BARRA^.NORFIM;
     CA[6]:=BARRA^.MOFIM;
     barra^.axin:=mbeta[1,1]*CA[1]+mbeta[1,2]*CA[2]+
              mbeta[1,3]*CA[3]+mbeta[1,4]*CA[4];
     barra^.axin:=barra^.axin+mbeta[1,5]*CA[5]+mbeta[1,6]*CA[6];
     barra^.norin:=mbeta[2,2]*CA[2]+mbeta[2,1]*CA[1]+
                mbeta[2,3]*CA[3]+
                mbeta[2,4]*CA[4];
     barra^.norin:=barra^.norin+mbeta[2,5]*CA[5]+mbeta[2,6]*CA[6];
     barra^.momin:=mbeta[3,3]*CA[3]+mbeta[3,1]*CA[1]+
                mbeta[3,2]*CA[2]+
                mbeta[3,4]*CA[4];
     barra^.momin:=barra^.momin+mbeta[3,5]*CA[5]+
                mbeta[3,6]*CA[6];
     barra^.axfim:=mbeta[4,4]*CA[4]+mbeta[4,1]*CA[1]+
                mbeta[4,2]*CA[2]+
                mbeta[4,3]*CA[3];
     barra^.axfim:=barra^.axfim+mbeta[4,5]*CA[5]+
                mbeta[4,6]*CA[6];
     barra^.norfim:=mbeta[5,1]*CA[1]+mbeta[5,2]*CA[2]+
                 mbeta[5,3]*CA[3]+
                mbeta[5,4]*CA[4];
     barra^.norfim:=barra^.norfim+mbeta[5,5]*CA[5]+
                 mbeta[5,6]*CA[6];
     barra^.mofim:=mbeta[6,1]*CA[1]+mbeta[6,2]*CA[2]+
                 mbeta[6,3]*CA[3]+
                mbeta[6,4]*CA[4];
     barra^.mofim:=barra^.mofim+mbeta[6,6]*CA[6]+
                mbeta[6,5]*CA[5];
   end;
end;



Procedure Tripae (busk:longint; Var AE:AEptr);
begin
```

```
  while busk<AE^.pos do
  AE:=AE^.next;
  while busk>AE^.pos do
  Ae:=AE^.ant;
end;


Procedure Carga  (N,B:longint; Topbar,barra:barptr; mbeta:matriz;
                  no,topno:noptr;     lastae,AE:aeptr;  Var topAE:aeptr);
Var
X,Y,I,busk,W:longint;
T,U:double;
G:longint;
AA:matriz;
CA:Array[1..6] of double;

begin
  barra:=topbar;
  FOR G:=1 TO B DO
  BEGIN
    T:=BARRA^.COSX;
    U:=BARRA^.COSY;
    matrizbeta(T,U,mbeta);
    For x:=1 to 6 do
    For y:=1 to 6 do
    AA[x,y]:=mbeta[y,x];
    CA[1]:=BARRA^.AXIN;
    CA[2]:=BARRA^.NORIN;
    CA[3]:=BARRA^.MOMIN;
    CA[4]:=BARRA^.AXFIM;
    CA[5]:=BARRA^.NORFIM;
    CA[6]:=BARRA^.MOFIM;
    barra^.axin:=AA[1,1]*CA[1]+AA[1,2]*CA[2]+AA[1,3]*CA[3]+
            AA[1,4]*CA[4];
    barra^.axin:=barra^.axin+AA[1,5]*CA[5]+AA[1,6]*CA[6];
    barra^.norin:=AA[2,2]*CA[2]+AA[2,1]*CA[1]+AA[2,3]*CA[3]+
             AA[2,4]*CA[4];
    barra^.norin:=barra^.norin+AA[2,5]*CA[5]+AA[2,6]*CA[6];
    barra^.momin:=AA[3,3]*CA[3]+AA[3,1]*CA[1]+AA[3,2]*CA[2]+
            AA[3,4]*CA[4];
    barra^.momin:=barra^.momin+AA[3,5]*CA[5]+AA[3,6]*CA[6];
    barra^.axfim:=AA[4,4]*CA[4]+AA[4,1]*CA[1]+AA[4,2]*CA[2]+
            AA[4,3]*CA[3];
    barra^.axfim:=barra^.axfim+AA[4,5]*CA[5]+AA[4,6]*CA[6];
    barra^.norfim:=AA[5,1]*CA[1]+AA[5,2]*CA[2]+AA[5,3]*CA[3]+
             AA[5,4]*CA[4];
    barra^.norfim:=barra^.norfim+AA[5,5]*CA[5]+AA[5,6]*CA[6];
    barra^.mofim:=AA[6,1]*CA[1]+AA[6,2]*CA[2]+AA[6,3]*CA[3]+
            AA[6,4]*CA[4];
    barra^.mofim:=barra^.mofim+AA[6,6]*CA[6]+AA[6,5]*CA[5];
    If g<>b then
    barra:=barra^.next;
  END;
  new(AE);
  AE^.eVAL:=0;
  AE^.next:=nil;
  AE^.pos:=1;
  FOR Y:=2 TO 3*N DO
```

```
begin
  lastAE:=AE;
  new(AE);
  lastAE^.ant:=AE;
  AE^.Eval:=0;
  AE^.pos:=Y;
  AE^.next:=lastAE;
end;
topAE:=AE;
I:=0;
FOR G:=1 TO B DO
BEGIN
  Tripbar(G.barra);
  busk:=BARRA^.noinicio*3-2;
  tripae(busk.ae);
  AE^.eval:=AE^.eval-BARRA^.AXIN;
  busk:=BARRA^.noinicio*3-1;
  tripae(busk.ae);
  AE^.eval:=AE^.eval-BARRA^.NORIN;
  busk:=BARRA^.noinicio*3;
  tripae(busk.ae);
  AE^.eval:=AE^.eval-BARRA^.MOMIN;
  busk:=BARRA^.nofim*3-2;
  tripae(busk.ae);
  AE^.eval:=AE^.eval-BARRA^.AXFIM;
  busk:=BARRA^.nofim*3-1;
  tripae(busk.ae);
  AE^.eval:=AE^.eval-BARRA^.NORFIM;
  busk:=BARRA^.nofim*3;
  tripae(busk.ae);
  AE^.eval:=AE^.eval-BARRA^.MOFIM;
END;
FOR X:=1 TO N DO
BEGIN
  i:=x;
  tripno(I.no);
  busk:=3*X-2;
  tripAE(busk.AE);
  AE^.Eval:=AE^.Eval+NO^.AXNO;
  busk:=3*X-1;
  tripAE(busk.AE);
  AE^.Eval:=AE^.Eval+NO^.norno;
  busk:=3*X;
  tripAE(busk.AE);
  AE^.Eval:=AE^.Eval+NO^.mono;
END;
FOR W:=1 TO N DO
BEGIN
  I:=w;
   tripno(I.no);
  CASE NO^.RESTRI OF
  3:BEGIN
      busk:=3*W-2;
      tripAE(busk.AE);
      AE^.Eval:=0;
      busk:=3*W-1;
      tripAE(busk.AE);
      AE^.Eval:=0;
```

```
          busk:=3*W;
          tripAE(busk,AE);
          AE^.Eval:=0;
        END;
      2:BEGIN
          busk:=3*W-2;
          tripAE(busk,AE);
          AE^.Eval:=0;
          busk:=3*W-1;
          tripAE(busk,AE);
          AE^.Eval:=0;
        END;
      1:BEGIN
          busk:=3*W-1;
          tripAE(busk,AE);
          AE^.Eval:=0;
        END;
      4:BEGIN
          busk:=3*W-2;
          tripAE(busk,AE);
          AE^.Eval:=0;
        END;
      0:begin
          no^.restri:=0;
        end;
      END;
    END;
end;



BEGIN
   WRITELN('PROGRAMA  PORTICO  PLANO');
   writeln('versao triangular');
   WRITE ('ARQUIVO DE DADOS= ');
   writeln;
   READLN(NOMEARQE);
   ASSIGN(ARQE,NOMEARQE):RESET(ARQE);
   WRITELN(NOMEARQE);
   WRITE ('ARQUIVO DE SAIDA= ');
   READln(NOMEARQS);
     GetTime(hour,minute,second,sec100);
    tempo:=60*minute+second+sec100/100;
   ASSIGN(ARQs,NOMEARQs):REwrite(ARQs);
   WRITELN(NOMEARQS);
   READLN(ARQE,B);
   READLN(ARQE,N);
   new(no);
   lastno:=no;
   X:=0;
   FOR I:=1 TO N DO
   BEGIN
     READLN(ARQE,no^.nu,no^.ABSNO,no^.ORDNO,no^.RESTRI);
     If no^.restri<>0 then
     begin
       x:=x+1;
       gno[X]:=no;
```

```
    end;
    if I=1 then
    begin
      no^.next:=nil;
      new(no);
    end
    else
    begin
      if I<N then
      begin
        no^.next:=lastno;
        lastno^.ant:=no;
        lastno:=no;
        new(no);
      end
      else
      begin
        no^.next:=lastno;
        lastno^.ant:=no;
        no^.ant:=nil;
      end;
    end;
END;
topno:=no;
new(barra);
lastbarra:=barra;
primbar:=barra;

FOR G:=1 TO B DO
BEGIN
    READLN(ARQE,barra^.num,BARRA^.NOINICIO,BARRA^.NOFIM,
          BARRA^.MINERCIA,
          BARRA^.MODELAST,BARRA^.AREA);
    I:=barra^.noinicio;
    tripno(I.no);
    R:=-no^.ordno;
    D:=-no^.absno;
    I:=barra^.nofim;
    tripno(I.no);
    R:=R+no^.ordno;
    D:=D+no^.absno;
    C:=sqrt(sqr(R)+sqr(D));
    barra^.comp:=C;
    barra^.cosx:=D/C;
    barra^.cosy:=R/C;
    if g=1 then
    begin
      barra^.next:=nil;
      new(barra);
    end
    else
    begin
      if g<b then
      begin
        barra^.next:=lastbarra;
        lastbarra^.ant:=barra;
        lastbarra:=barra;
        new(barra);
```

```
      end
      else
      begin
        barra^.next:=lastbarra;
        lastbarra^.ant:=barra;
      end;
    end;
END;
topbar:=barra;
FOR X:=1 TO ((3*N-1)*3*N-(3*N*(3*N-1) DIV 2) +3*N)  DO
begin
  new(kglobal);
  kglobal^.kval:=0;
  kglobal^.ordem:=X;
  If X=1 then
  begin
    lastk:=kglobal;
    auxk:=kglobal;
    kglobal^.next:=nil;
  end
  else
  begin
    lastk^.ant:=kglobal;
    kglobal^.next:=lastk;
    lastk:=kglobal;
  end;
  If x= ((3*N-1)*3*N-(3*N*(3*N-1) DIV 2) +3*N) then
  begin
    kglobal^.ant:=nil;
  end;
end;
topk:=kglobal;
barra:=primbar;
FOR G:=1 TO B DO
BEGIN
    assemhalf(n,B,G,barra,topbar,no,topno,Kglobal,topk);
   If G<>b then
   begin
     barra:=barra^.ant;
   end;
end;
W:=1;
WHILE W<=N DO
BEGIN
  I:=W;
  Tripno(I,no);
  CASE NO^.RESTRI OF
    3:BEGIN
        rest1;
        rest2;
        rest3;
    end;
    2:BEGIN
        rest1;
        rest2;
    end;
    1:BEGIN
        rest2;
```

```
      end;
      4:BEGIN
          rest1;
      END;
      0:begin
          no^.restri:=0;
      end;
    END;
    W:=W+1;
END;
writeln('Foi construida a matriz de rigidez e armazenadas '.
        ((3*N-1)*3*N-(3*N*(3*N-1) DIV 2) +3*N),' posicoes');
         kglobal:=topk;
barra:=topbar;
FOR G:=1 TO B DO
BEGIN
  BARRA^.NORIN:=0;
  BARRA^.NORFIM:=0;
  BARRA^.AXIN:=0;
  BARRA^.AXFIM:=0;
  BARRA^.MOMIN:=0;
  BARRA^.MOFIM:=0;
  BARRA^.UV:=0;
  BARRA^.UVNI:=0;
  BARRA^.AXUV:=0;
  BARRA^.AXUVNI:=0;
  FOR X:=1 TO 4 DO
  BEGIN
    barra^.cc[x]:=0;
    barra^.di[x]:=0;
    barra^.axc[x]:=0;
    barra^.axcdi[x]:=0;
    barra^.ud[x]:=0;
    barra^.tam[x]:=0;
    barra^.uddi[x]:=0;
    barra^.mab[x]:=0;
    barra^.mabdi[x]:=0;
  end;
  If g<>b then
  barra:=barra^.next;
END;

READLN(ARQE,OPTION);  (*ESTA VARIAVEL. DA AO USUARIO A OPCAO DE
                COLOCAR SEU CARREGAMENTO SEGUNDO O SISTEMA
                GLOBAL(0) DE COORDENADAS. OU LOCAL(1)*)

READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.G);
    tripbar(G.barra);
    for x:=1 to 4 do
    BEGIN
      READ(ARQE.BARRA^.CC[X],BARRA^.DI[X]);
        (*CC:=CARGA CONCENTRADA NA COORDENADA GLOBAL 2*)
      E:=BARRA^.CC[X]*BARRA^.COSX; (*agora na coord.
```

```
                                local 2*)
        F:=BARRA^.CC[X]*BARRA^.COSY; (*coord. local 1*)
        CONC:=BARRA^.CC[X];
        DIST:=BARRA^.DI[X];
        remete;
        ENGASTCC(OPTION,E,F,COMPR,CONC,DIST,NIN,NFIM,MIN,
          MFIM, AIN,AFIM);
        recebe;
      END;
      READLN(ARQE);
    end;
  end;
READLN(ARQE,W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE,G);
    tripBAR(G,BARRA);
    for x:=1 to 4 do
    begin
      READ(ARQE,BARRA^.AXC[X],BARRA^.AXCDI[X]);
      AXIC:=BARRA^.AXC[X];
      AXICDI:=BARRA^.AXCDI[X];
      COSENO:=BARRA^.COSX;
      SENO:=BARRA^.COSY;
      REMETE;
      ENGASTAXC(OPTION,AXIC,AXICDI,COMPR,COSENO,SENO,E,F,
            NIN,NFIM,MIN,MFIM,
        AIN,AFIM);
      RECEBE;
    end;
    READLN(ARQE);
  end;
END;
READLN(ARQE,W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE,G);
    TripBAR(G,BARRA);
    read(arqe,total);(*total=1 significa
                que a carga se extende por toda a
      barra*)
    If total=1 then
    begin
      read(arqe,barra^.ud[1]);
      ud:=barra^.ud[1];
      tam:=barra^.comp;
      uddi:=tam/2;
      coseno:=barra^.cosx;
      seno:=barra^.cosy;
      remete;
      ENGASTud(OPTION,COMPR,ud,tam,uddi,COSENO,SENO,e,f,m,
            h,NIN,NFIM,MIN,
        MFIM, AIN,AFIM);
      recebe;
```

```
      end
      else
      begin
        for x:=1 to 4 do
        begin
          READ(ARQE,BARRA^.UD[X],BARRA^.TAM[X],
              BARRA^.UDDI[X]);
              (*UD:=CARGA UN. DISTRIB. COORDENADA GLOBAL
                    2*)
          ud:=BARRA^.UD[X];
          tam:=BARRA^.TAM[X];
          uddi:=BARRA^.UDDI[X];
          coseno:=BARRA^.COSX;
          seno:=BARRA^.COSY;
          remete;
          ENGASTud(OPTION,COMPR,ud,tam,uddi,COSENO,SENO,e,f,
               m,h,NIN,NFIM,MIN,
               MFIM,AIN,AFIM);
          recebe;
        end;
      end;
      READLN(ARQE);
    end;
  END;
  READLN(ARQE,W);
  if w<>0 then
  begin
    for y:=1 to w do
    begin
      READ(ARQE,G);
      TripBAR(G,BARRA);
      read(arqe,BARRA^.UV,BARRA^.UVNI);
      E:=BARRA^.UV*BARRA^.COSX;
      F:=BARRA^.UV*BARRA^.COSY;
      COSENO:=BARRA^.COSX;
      SENO:=BARRA^.COSY;
      UVAR:=BARRA^.UV;
      remete;
      UNI:=BARRA^.UVNI;
      ENGASTUV(OPTION,E,F,COMPR,COSENO,SENO,UVAR,UNI,NIN,NFIM,
            MIN,
           MFIM,AIN,AFIM);
      recebe;
      readln(arqe);
    end;
  end;
  READLN(ARQE,W);
  if w<>0 then
  begin
    for y:=1 to w do
    begin
      READ(ARQE,G);
      TripBAR(G,BARRA);
      for x:=1 to 4 do
      begin
        READ(ARQE,BARRA^.MAB[X],BARRA^.MABDI[X]);
        momb:=BARRA^.MAB[X];
        mombdi:=BARRA^.MABDI[X];
```

```
        REMETE;
        ENGASTMB(MOMB.MOMBDI.COMPR.NIN.NFIM.MIN.MFIM.AIN.
                 AFIM);
        RECEBE;
      end;
      READLN(ARQE);
    end;
end;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  READ(ARQE.G);
  TripBAR(G.BARRA);
  READLN(ARQE.BARRA^.AXUV.BARRA^.AXUVNI);
     (*NAO HA OPCAO PARA O SISTEMA GLOBAL*)
  AXVAR:=BARRA^.AXUV;
  AXVARNI:=BARRA^.AXUVNI;
  REMETE;
  ENGASTAXUV(AXVAR.AXVARNI.COMPR.
             NIN.NFIM.MIN.MFIM.AIN.AFIM);
  RECEBE;
end;
FOR I:=1 TO N DO
BEGIN
   tripno(I.no);
  NO^.AXNO:=0;
  NO^.NORNO:=0;
  NO^.MONO:=0;
END;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.I);
     tripno(I.no);
    READLN(arqe.NO^.AXNO.NO^.NORNO.NO^.MONO);
  end;
end;
Carga (N.B.topbar.barra.mbeta.no.topno.lastAE.AE.topAE);
writeln('Foi construido o vetor de carga');
ELE[1]:=1;
FOR X:=2 TO 3*N DO
ELE[X]:=0;
X:=1;
G:=1;
ae:=topae;
FOR Y:=1 TO 3*N-1  DO
BEGIN
  W:=((X-1)*3*N-X*(X-1) DIV 2)+X;
  kglobal:=auxk;
   while W<kglobal^.ordem do
  kglobal:=kglobal^.next;
  while W>kglobal^.ordem do
  kglobal:=kglobal^.ant;
  val1:=kglobal^.kval;
   Z:=((Y-1)*3*N-Y*(Y-1) div 2)+3*N;
```

```
  While Kglobal^.ordem<Z do
  kglobal:=kglobal^.ant;
  If kglobal^.ordem>auxk^.ordem then
  auxk:=kglobal;
  WHILE G<3*N DO
  BEGIN
    ELE[3*N-G+X]:=-KGLOBAL^.kval/val1;
    kglobal:=kglobal^.next;
    G:=G+1;
  END;
  busk:=X;
  tripAE(busk,AE);
  D:=AE^.eval;
  If D<>0 then
  begin
    FOR A:=X+1 TO 3*N DO
    begin
      If ELE[A]<>0 then
      begin
        busk:=A;
        tripAE(busk,AE);
        AE^.eval:=AE^.eval+D*ELE[A];
      end;
    end;
  end;
  FOR A:=Y+1 TO 3*N DO
  BEGIN
    If ELE[A]<>0 then
    begin
      FOR K:=A TO 3*N DO
      BEGIN
        W:=((Y-1)*3*N-Y*(Y-1) DIV 2)+K;
        while W<kglobal^.ordem do
        kglobal:=kglobal^.next;
        while W>kglobal^.ordem do
        kglobal:=kglobal^.ant;
        If kglobal^.kval<>0 then
        begin
          R:=kglobal^.kval;
          Z:=((A-1)*3*N-A*(A-1) DIV 2)+K;
          while Z<kglobal^.ordem do
          kglobal:=kglobal^.next;
          while Z>kglobal^.ordem do
          kglobal:=kglobal^.ant;
          Kglobal^.kval:=KGLOBAL^.kval+ELE[A]*R;
        end;
      end;
    END;
  end;
  X:=X+1;
  G:=X;
END;
writeln('Foi construida a matriz U ');
AE:=topAE;
kglobal:=topk;
R:=0;
ELE[3*N]:=TOPAE^.EVAL/TOPK^.KVAL;
AE:=AE^.next;
```

```
kglobal:=kglobal^.next;
AE^.EVAL:=AE^.EVAL-kglobal^.kval*ELE[3*N];
kglobal:=kglobal^.next;
ELE[3*n-1]:=AE^.EVAL/kglobal^.kval;
kglobal:=kglobal^.next;
For K:=3*N-1 downto 2 do
begin
  AE:=AE^.next;
  AE^.eval:=Ae^.eval-kglobal^.kval*ELE[3*N];
  Y:=1;
  A:=3*N-K+1;
  while Y<A do
  begin
    kglobal:=kglobal^.next;
    R:=R+kglobal^.kval*ELE[3*N-Y];
    Y:=Y+1;
  end;
  kglobal:=kglobal^.next;
  ELE[K-1]:=(AE^.eval-R)/kglobal^.kval;
  If K>2 then
  begin
    R:=0;
    kglobal:=kglobal^.next;
  end;
end;
FOR G:=1 TO N DO
BEGIN
  I:=G;
  tripno(I.no);
  NO^.DAX:=ele[3*G-2];
  NO^.DNOR:=ele[3*G-1];
  NO^.ROT:=ele[3*G];
end;
cabenos;
x:=0;
for g:=1 to N do
begin
  I:=G;
  tripno(I.no);
  C:=no^.dax;
  D:=no^.dnor;
  E:=no^.rot;
  mostradesloc(C,D,E,g,X);
end;
Roda (B.topbar.barra.mbeta);
FOR G:=1 TO B DO
BEGIN
  TripBAR(G.BARRA);
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T,U.MBETA);
  FOR X:=1 TO 6 DO
  BEGIN
    I:=BARRA^.NOINICIO;
     tripno(I.no);
    R:=MBETA[X.1]*NO^.DAX+MBETA[X.2]*NO^.DNOR+
       MBETA[X.3]*NO^.ROT;
    I:=BARRA^.NOFIM;
```

```
    tripno(I.no);
    CA[X]:=R+MBETA[X,4]*NO^.DAX+MBETA[X,5]*NO^.DNOR+
        MBETA[X,6]*NO^.ROT;
  END;
  P:=BARRA^.MODELAST;
  J:=BARRA^.MINERCIA;
  S:=BARRA^.AREA;
  C:=BARRA^.COMP;
  MATRIZRIG(C,J,S,P,MRIGB);
  FOR X:=1 TO 6 DO
  BARRA^.AXIN:=BARRA^.AXIN+MRIGB[1,X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.NORIN:=BARRA^.NORIN+MRIGB[2,X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.MOMIN:=BARRA^.MOMIN+MRIGB[3,X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.AXFIM:=BARRA^.AXFIM+MRIGB[4,X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.NORFIM:=BARRA^.NORFIM+MRIGB[5,X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.MOFIM:=BARRA^.MOFIM+MRIGB[6,X]*CA[X];
END;
 WRITELN(ARQS);
writeln(arqs,'                    ESFORCOS NAS BARRAS');
WRITELN(ARQS);
writeln(arqs,' BARRA  NO INICIAL                NO FINAL');
writeln(arqs,'       AXIAL   NORMAL  MOMENTO  AXIAL   NORMAL  MOMENTO');
G:=1;
FOR G:=1 TO B DO
BEGIN
  tripbar(G.barra);
  WRITE(ARQS,'  ',g:3,' ',BARRA^.AXIN:8:2,' ',BARRA^.NORIN:8:2);
  WRITE(ARQS,' ',BARRA^.MOMIN:8:2,' ',BARRA^.AXFIM:8:2);
  WRITELN(ARQS,' ',BARRA^.NORFIM:8:2,' ',BARRA^.MOFIM:8:2);
END;
FOR I:=1 TO N DO
BEGIN
   tripno(I.no);
  NO^.REAX:=0;
  NO^.RENOR:=0;
  NO^.REMON:=0;
END;
FOR G:=1 TO B DO
BEGIN
  TripBAR(G.BARRA);
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T,U,MBETA);
  FOR X:=1 TO 6 DO
  FOR K:=1 TO 6 DO
  AA[X,K]:=MBETA[K,X];
  FOR X:=1 TO 6 DO
  BEGIN
    R:=AA[X,1]*BARRA^.AXIN+AA[X,2]*BARRA^.NORIN+
        AA[X,3]*BARRA^.MOMIN;
    CA[X]:=R+AA[X,4]*BARRA^.AXFIM+AA[X,5]*BARRA^.NORFIM+
         AA[X,6]*BARRA^.MOFIM;
  END;
```

```
    I:=BARRA^.NOINICIO;
     tripno(I.no);
    NO^.REAX:=NO^.REAX+CA[1];
    NO^.RENOR:=NO^.RENOR+CA[2];
    NO^.REMON:=NO^.REMON+CA[3];
    I:=BARRA^.NOFIM;
     tripno(I.no);
    NO^.REAX:=NO^.REAX+CA[4];
    NO^.RENOR:=NO^.RENOR+CA[5];
    NO^.REMON:=NO^.REMON+CA[6];
  END;
  x:=0;
  writeln(arqs);
  writeln(arqs.'                    REACOES NOS APOIOS');
 writeln(arqs);
 write(arqs.' NO   COORDENADA 1      COORDENADA 2');
 writeln(arqs.'        COORDENADA 3');
  FOR G:=1 TO N DO
  BEGIN
    I:=G;
    tripno(I.no);
    IF NO^.RESTRI>0 THEN
    BEGIN
      write(arqs.' ',G:3,'     ',NO^.REAX:9:2);
      WRITE(arqs.'          ',no^.renor:9:2,'          ');
      WRITELN(arqs,no^.remon:9:2);
    END;
  END;
  t1:=tempo;
  GetTime(hour,minute,second,sec100);
  writeln('hora= ',hour,' ',minute,' ',second,' ',sec100);
  tempo:=60*minute+second+sec100/100;
  t2:=tempo-t1;
  writeln(arqs,'O tempo total de processamento utilizado pelo metodo');
  writeln(arqs,'triangular foi de ',t2:7:2,' segundos.');
  LIMPA;
  close(arqs);
  for g:=1 to 6 do
  writeln;
  writeln('Foi analisada a estrutura constante do arquivo. ',nomearqe,'.');
  writeln('Os resultados encontrados estao no arquivo. ',nomearqs);
  writeln;
  writeln;
  writeln('Para sair digite qualquer tecla');
  repeat until keypressed;
end.
```

ANEXO 3
Listagem do Programa Versão "Sparse"

```
PROGRAM PORTICO_PLANO;     (*armazena metade da matriz global sem zeros e aloca
dinamicamente

                                        as variáveis*)
Uses CRT.DOS;
type
   matriz=array[1..6,1..6] of double;
   tarray=array[1..4] of double;
   vetors=array[1..6] of double;
   vetor =array[1..2000] of double;
   arquivo=text;
   AEptr=^AErec;
   AErec=record
         Eval:double;
         pos:longint;
         next:AEptr;
         ant:AEptr;
   end;
   kbalptr=^krec;
   krec=record
         kval:double;
         ordem:longint;
         ant:kbalptr;
         next:kbalptr;
   end;
   barptr=^barrec;
   barrec=record
          noinicio:longint;
          nofim:longint;
          minercia:double;
          modelast:double;
          area:double;
          comp:double;
          cosx:double;
          cosy:double;
          cc:tarray;  (*carga concentrada*)
          di:tarray;  (*distancia do no inicio ao ponto deaplicacao*)
          df:tarray;  (*dist. do no fim ao ponto de aplicacao*)
          ud:tarray;  (*carga uniformemnte distribuida*)
          uddf:tarray; (*dist. do no fim ao ponto medio da carga*)
          uddi:tarray; (*dist. do no inicio ao ponto  medio da carga*)
          tam:tarray;  (*comprimento da carga uniformemente dist.*)
          uv:double;     (*valor maior da carga uniformemente variada*)
          uvni:double;   (*valor da uv no no inicio*)
          axc:tarray;  (*carga axial aplicada na barra*)
          axcdi:tarray;(*distancia do no inicio ao ponto de aplicacao*)
          axcdf:tarray;(*distancia do no fim ao ponto de aplicacao da carga*)
          axud:tarray; (**)
          atam:tarray;
          axudi:tarray;
          axudf:tarray;
          axuv:double;
          axuvni:double;
          mab:tarray;
          mabdi:tarray;
```

```
            mabdf:tarray;
            norin:double;
            norfim:double;
            axin:double;
            axfim:double;
            momin:double;
            mofim:double;
            ant:barptr;
            next:barptr;
            num:longint;
    end;
    noptr=^norec;
    norec=record
          nu:longint;
          absno:double;
          ordno:double;
          norno:double;
          axno:double;
          mono:double;
          restri:integer;
          dax:double;
          dnor:double;
          rot:double;
          reax:double;
          renor:double;
          remon:double;
          ant:noptr;
          next:noptr;
    end;
     guarda=array[1..20] of noptr;
     guarba=array[1..20] of barptr;
var
  lastAE,AE,topAE:AEptr;
  primbar,lastbarra,barra,topbar:barptr;
  noi,inno,fino,lastno,no,topno:noptr;
  proxk,auxk,kglobal,lastk,topk:kbalptr;
  gno:guarda;
  gba:guarba;
  iaux,max,maxres,linha,numbar,total:longint;
  ABC,numno,contador,alfa,B,G,I,L,X,Y,W,NI,NF,AAA,GG,OPTION,limax,
          limin,ender,sig,zero,bbb,ADVERT,SINAL,alarme,bip,fla,
          busK,comp0,A,N,K,Z,ord,posicao,limite,lon:longint;
  C,D,E,F,H,J,M,O,P,Q,R,S,T,U,CONC,COMPR,COSENO,SENO,UVAR,NIN,NFIM,
     axic,axicdi,axvar,axvarni,uni,ain,afim,dist,ud,tam,uddi,min,
     mfim,momb,mombdi,valor,val1,val2,numer,denom,aux:double;
    t1,t2,sec, hund :real;
tempo:real;
Hour, Minute, Second, Sec100: word;
  CC,AA,BB,MBETA,MRIGB: MATRIZ;
  auxivet:Vetor;
  ELE:Vetor;
  CA:Vetors;
  ARQE,ARQS:ARQUIVO;
  NOMEARQE,NOMEARQS:STRING;

procedure remete;
begin
   nin:=barra^.norin;
```

```
  nfim:=barra^.norfim;
  ain:=barra^.axin;
  afim:=barra^.axfim;
  min:=barra^.momin;
  mfim:=barra^.mofim;
  compr:=barra^.comp;
end;
procedure recebe;
begin
  barra^.norin:=nin;
  barra^.norfim:=nfim;
  barra^.axin:=ain;
  barra^.axfim:=afim;
  barra^.momin:=min;
  barra^.mofim:=mfim;
end;

procedure Limpa;
begin
   lastk:=topk^.next;
   kglobal:=topk;
   while kglobal^.next<>nil do
   begin
     dispose(kglobal);
     kglobal:=lastk;
     lastk:=kglobal^.next;
   end;
   barra:=topbar;
   lastbarra:=topbar^.next;
   while barra^.next<>nil do
   begin
     dispose(barra);
     barra:=lastbarra;
     lastbarra:=barra^.next;
   end;
   no:=topno;
   lastno:=no^.next;
   while no^.next<>nil do
   begin
     dispose(no);
     no:=lastno;
     lastno:=no^.next;
   end;
end;

Procedure Tripbar (busk:longint; Var barra:barptr);
begin
  while busk<barra^.num do
  barra:=barra^.next;
  while busk>barra^.num do
  barra:=barra^.ant;
end;

procedure matrizbeta(t,u:double; Var mbeta:matriz);
begin
  MBETA[1,1]:=T;
  MBETA[1,2]:=U;
  MBETA[1,3]:=0;
```

```
  MBETA[1,4]:=0;
  MBETA[1,5]:=0;
  MBETA[1,6]:=0;
  MBETA[2,1]:=-U;
  MBETA[2,2]:=T;
  MBETA[2,3]:=0;
  MBETA[2,4]:=0;
  MBETA[2,5]:=0;
  MBETA[2,6]:=0;
  MBETA[3,1]:=0;
  MBETA[3,2]:=0;
  MBETA[3,3]:=1;
  MBETA[3,4]:=0;
  MBETA[3,5]:=0;
  MBETA[3,6]:=0;
  MBETA[4,1]:=0;
  MBETA[4,2]:=0;
  MBETA[4,3]:=0;
  MBETA[4,4]:=T;
  MBETA[4,5]:=U;
  MBETA[4,6]:=0;
  MBETA[5,1]:=0;
  MBETA[5,2]:=0;
  MBETA[5,3]:=0;
  MBETA[5,4]:=-U;
  MBETA[5,5]:=T;
  MBETA[5,6]:=0;
  MBETA[6,1]:=0;
  MBETA[6,2]:=0;
  MBETA[6,3]:=0;
  MBETA[6,4]:=0;
  MBETA[6,5]:=0;
  MBETA[6,6]:=1;
END;

procedure matrizrig(C,J,S,P:double;Var mrigb:matriz);
    (*C=COMP. J=MOM. DE INERCIA. S=AREA. P=MODULO DE EL.*)
begin
  MRIGB[1,1]:=P*S/C;
  MRIGB[1,2]:=0;
  MRIGB[1,3]:=0;
  MRIGB[1,4]:=-P*S/C;
  MRIGB[1,5]:=0;
  MRIGB[1,6]:=0;
  MRIGB[2,1]:=0;
  MRIGB[2,2]:=12*P*J/(SQR(C)*C);
  MRIGB[2,3]:=6*P*J/(SQR(C));
  MRIGB[2,4]:=0;
  MRIGB[2,5]:=-12*P*J/(SQR(C)*C);
  MRIGB[2,6]:=6*P*J/(SQR(C));
  MRIGB[3,1]:=0;
  MRIGB[3,2]:=6*P*J/(SQR(C));
  MRIGB[3,3]:=4*P*J/C;
  MRIGB[3,4]:=0;
  MRIGB[3,5]:=-6*P*J/(SQR(C));
  MRIGB[3,6]:=2*P*J/C;;
  MRIGB[4,1]:=-P*S/C;
  MRIGB[4,2]:=0;
```

```
  MRIGB[4,3]:=0;
  MRIGB[4,4]:=P*S/C;
  MRIGB[4,5]:=0;
  MRIGB[4,6]:=0;
  MRIGB[5,1]:=0;
  MRIGB[5,2]:=-12*P*J/(SQR(C)*C);
  MRIGB[5,3]:=-6*P*J/(SQR(C));
  MRIGB[5,4]:=0;
  MRIGB[5,5]:=12*P*J/(SQR(C)*C);
  MRIGB[5,6]:=-6*P*J/(SQR(C));
  MRIGB[6,1]:=0;
  MRIGB[6,2]:=6*P*J/(SQR(C)) ;
  MRIGB[6,3]:=2*P*J/C;
  MRIGB[6,4]:=0;
  MRIGB[6,5]:=-6*P*J/(SQR(C)) ;
  MRIGB[6,6]:=4*P*J/C;
end;

procedure produmax (aa,bb:matriz; Var cc:matriz) ;
Var
x,y,k:integer;

begin
  for x:=1 to 6 do
  for y:=1 to 6 do
  begin
    cc[x,y]:=0;
     for k:=1 to 6 do
    begin
      cc[x,y]:=cc[x,y]+aa[x,k]*bb[k,y];
    end;
  end;
end;

Procedure Tripno(I:longint; Var no:noptr);
begin
  while I<no^.nu do
  no:=no^.next;
  while I>no^.nu do
  no:=no^.ant;
end;

procedure assemblagem(n,B,limax,limin:longint; barra:barptr;
      no,topno:noptr;  max,maxres,linha,numbar:longint;
      gno:guarda;  Var auxivet:vetor; Var G:longint);

Var
x,k,y,ni,nf,ww,yy,v,I,sinal,numno,bip,lin,col:longint;
cont,Ind,xx:longint;
cc,bb,aa,mbeta,mrigb:matriz;
t,u,p,s,j,C,r,d:double;
inno,fino:noptr;

Procedure vecalc(lin,col:longint; Var xx:longint);
begin
  XX:=((lin-1)*3*N-lin*(lin-1) DIV 2) +col;
end;
Procedure ConstCc;
```

```
Var
x,y,k:longint;
BEGIN
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T,U,MBETA);
  P:=BARRA^.MODELAST;
  J:=BARRA^.MINERCIA;
  S:=BARRA^.AREA;
  C:=BARRA^.COMP;
  MATRIZRIG(C,J,S,P,MRIGB);
  FOR X:=1 TO 6 DO
  FOR K:=1 TO 6 DO
  AA[X,K]:=MBETA[K,X];
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BB[X,Y]:=MRIGB[X,Y];
  PRODUMAX(AA,BB,CC);
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BEGIN
    AA[X,Y]:=CC[X,Y];
    BB[X,Y]:=MBETA[X,Y];
  END;
  PRODUMAX(AA,BB,CC);

end;
Procedure Dirichilet;
Var
Y,X,bbb,sinal:longint;
begin
  Y:=2;
  sinal:=0;
  Case no^.restri of
  3:begin
      while Y>=0 do
      begin
        X:=2;
        While X>=0 do
        begin
          lin:=numno*3-X;
          col:=numno*3-y;
          If col=lin then
          begin
            vecalc(lin,col,xx);
            If (XX<=limax) and (xx>limin) then
            begin
              auxivet[xx-limin+3*N-max]:=1;
              For bbb:=xx-limin+3*N-max+1 to 3*N do
              auxivet[bbb]:=0;
              G:=numbar+1;
              sinal:=1;
              X:=-1;
            end
            else
              x:=x-1;
          end
          else
```

```
          X:=X-1;
        end;
        If sinal=1 then
        begin
          Y:=-1;
        end
        else
        begin
          Y:=Y-1;
        end;
      end;
end;
2:begin
    while Y>=1 do
    begin
      X:=2;
      while X>=1 do
      begin
        lin:=numno*3-X;
        col:=numno*3-y;
        If col=lin then
        begin
          vecalc(lin,col,xx);
          If (XX<=limax) and (xx>limin) then
          begin
            auxivet[xx-limin+3*N-max]:=1;
            For bbb:=xx-limin+3*N-max+1 to 3*N do
            auxivet[bbb]:=0;
            bip:=1;
            G:=numbar+1;
            X:=0;
            sinal:=1;
          end
          else
            x:=x-1;
        end
        else
          X:=X-1;
      end;
      If sinal=1 then
      begin
        Y:=-1;
      end
      else
      begin
        Y:=Y-1;
      end;
    end;
end;
1:begin
    lin:=numno*3-1;
    col:=numno*3-1;
    If col=lin then
    begin
      vecalc(lin,col,xx);
      If XX<=limax then
      If xx>limin then
      begin
```

```
          auxivet[xx-limin+3*N-max]:=1;
          For bbb:=xx-limin+3*N-max+1 to 3*N do
          auxivet[bbb]:=0;
          G:=numbar+1;
        end;
      end;
    end;
  end;
end;
Procedure Assem;
Var
X,Y:integer;
begin
  For X:=2 downto 0 do
  For Y:=X downto 0 do
  begin
    lin:=ni*3-X;              {4}
    col:=ni*3-y;
    vecalc(lin,col,xx);
    If XX<=limax then
    If xx>limin then
    begin
      If sinal=0 then
      begin
        constCC;
        sinal:=1;
      end;
      auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                    cc[3-x,3-y];
    end;
    lin:=nf*3-X;          {3}
    col:=nf*3-y;
    vecalc(lin,col,xx);
    If XX<=limax then
    If xx>limin then
    begin
      If sinal=0 then
      begin
        constCC;
        sinal:=1;
      end;
      auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                        cc[6-x,6-y];
    end;
  end;
  For X:=2 downto 0 do
  For Y:=2 downto 0 do
  begin
    lin:=ni*3-X;   {1}
    col:=nf*3-y;
    If col>=lin then
    begin
      vecalc(lin,col,xx);
      If XX<=limax then
      If xx>limin then
      begin
        If sinal=0 then
        begin
```

```
          constCC;
          sinal:=1;
        end;
        auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                    cc[3-x,6-y] ;
      end;
    end;
    lin:=nf*3-X;      {2}
    col:=ni*3-y;
    If col>=lin then
    begin
      vecalc(lin,col,xx);
      If XX<=limax then
      If xx>limin then
      begin
        If sinal=0 then
        begin
          constCC;
          sinal:=1;
        end;
        auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                      cc[6-x,3-y];
      end;
    end;
  end;
end;
BEGIN
  sinal:=0;
  NI:=BARRA^.NOINICIO;
  NF:=BARRA^.NOFIM;
  I:=Barra^.noinicio;
  tripno(I,no);
  inno:=no;
  I:=Barra^.nofim;
  tripno(I,no);
  fino:=no;
  If ni<nf then
  begin
    lin:=ni*3-2;          {4}
    col:=ni*3-2;
    vecalc(lin,col,xx);
    If XX<=limax then
    begin
      lin:=nf*3;
      col:=nf*3;
      vecalc(lin,col,xx);
      If XX>=limin then
      begin
        assem;
      end;
    end;
  end
  else
  begin
    lin:=nf*3-2;          {4}
    col:=nf*3-2;
    vecalc(lin,col,xx);
    If XX<=limax then
```

```
  begin
    lin:=ni*3;
    col:=ni*3;
    vecalc(lin,col,xx);
    If XX>=limin then
    begin
      assem;
    end;
  end;
end;
if fino^.restri<>0 then
begin
   no:=fino;
   numno:=barra^.nofim;
   dirichilet;
end;
If inno^.restri<>0 then
begin
  no:=inno;
  numno:=barra^.noinicio;
  richilet;
end;
If G=numbar then
begin
  For K:=1 to maxres do
  begin
    iaux:=gno[k]^.restri;
    Case iaux  of
    3:begin
        For Y:=2 downto 0 do
        begin
          cont:=gno[k]^.nu;
          If (cont*3-Y)>(3*N-max+1) then
          begin
            col:=cont*3-Y;
            lin:=linha;
            vecalc(lin,col,xx);
            auxivet[xx-limin+3*N-max]:=0;
          end;
        end;
      end;
    2:begin
        For Y:=2 downto 1 do
        begin
          cont:=gno[k]^.nu;
          If (cont*3-Y)>(3*N-max+1) then
          begin
            col:=cont*3-Y;
            lin:=linha ;
            vecalc(lin,col,xx);
            auxivet[xx-limin+3*N-max]:=0;
          end;
        end;
      end;
    1:begin
        cont:=gno[k]^.nu;
        If (cont*3-1)>(3*N-max+1) then
        begin
```

```
            col:=cont*3-1;
            lin:=linha;
            vecalc(lin,col,xx);
            auxivet[xx-limin+3*N-max]:=0;
          end;
        end;
      end;
    end;
  end;
end;


procedure ENGASTUV(OPTION,E,F,COMPR,COSENO,SENO,UVAR,UNI:double;
            Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);

begin

  IF OPTION=0 THEN
  BEGIN            (*SITEMA GLOBAL DE COORD.*)
    IF UNI=0 THEN
    BEGIN
      NIN:=NIN-(3/20)*E*COMPR*COSENO;
      NFIM:=NFIM-(7/20)*E*COMPR*COSENO;
      AIN:=AIN+(F*COMPR*SENO/2)*(COMPR*SENO/3)/COMPR;
      AFIM:=AFIM+(F*COMPR*SENO)*(COMPR*SENO/3)/COMPR;
      MIN:=MIN-E*SQR(COMPR*COSENO)/30;
      MFIM:=MFIM+E*SQR(COMPR*COSENO)/20;
    END
    ELSE
    BEGIN
      NIN:=NIN-(7/20)*E*COMPR*COSENO;
      NFIM:=NFIM-(3/20)*E*COMPR*COSENO;
      AIN:=AIN+(F*COMPR*SENO)*(COMPR*SENO/3)/COMPR;
      AFIM:=AFIM+(F*COMPR*SENO/2)*(COMPR*SENO/3)/COMPR;
      MIN:=MIN-E*SQR(COMPR*COSENO)/20;
      MFIM:=MFIM+E*SQR(COMPR*COSENO)/30;
    END;
  END
  ELSE
  BEGIN        (*SISTEMA LOCAL*)
    IF UNI=0 THEN
    BEGIN
      NIN:=NIN-(3/20)*UVAR*COMPR;
      NFIM:=NFIM-(7/20)*UVAR*COMPR;
      MIN:=MIN-UVAR*SQR(COMPR)/30;
      MFIM:=MFIM+UVAR*SQR(COMPR)/20;
    END
    ELSE
    BEGIN
      NIN:=NIN-(7/20)*UVAR*COMPR;
      NFIM:=NFIM-(3/20)*UVAR*COMPR;
      MIN:=MIN-UVAR*SQR(COMPR)/20;
      MFIM:=MFIM+UVAR*SQR(COMPR)/30;
     END;
  END;
END;

procedure ENGASTCC(OPTION,E,F,COMPR,CONC,DIST:double;
```

```
                Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
var
cont:longint;
begin
  if option=0 then
  begin
    nin:=nin-e*(sqr((compr-dist)/compr)*(3-2*(compr-dist)/compr));
    nfim:=nfim-e*(sqr(dist/compr)*(3-2*dist/compr));
    min:=min-e*(dist*sqr((compr-dist)/compr));
    mfim:=mfim+e*(compr-dist)*sqr(dist/compr);
    ain:=ain-f*(compr-dist)/compr;
    afim:=afim-f*dist/compr;
  end
  else
  begin
 nin:=nin-conc*(sqr((compr-dist)/compr)*(3-2*(compr-dist)/compr));
    nfim:=nfim-conc*(sqr(dist/compr)*(3-2*dist/compr));
    min:=min-conc*(dist*sqr((compr-dist)/compr));
    mfim:=mfim+conc*(compr-dist)*sqr(dist/compr);
  end;

end;

procedure ENGASTUD(OPTION,COMPR,UD,TAM,UDDI,COSENO,SENO:double;
                Var E,F,M,H,NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
begin
  if option=0 then
  begin
    E:=ud*tam*sqr(coseno);
    F:=ud*tam*coseno*seno;
    M:=uddi/compr;
    H:=sqr(tam/(2*compr));
    NIN:=NIN-E*(1-3*sqr(M)-H+2*M*(sqr(M)+H));
    NFIM:=NFIM-E*(3*sqr(M)+H-2*M*(sqr(M)+H));
    MIN:=MIN-E*(UDDI*sqr((COMPR-UDDI)/COMPR)-H*(3*(COMPR-UDDI)-
       COMPR)/3);
    MFIM:=MFIM+E*((COMPR-UDDI)*sqr(M)-H*(3*UDDI-COMPR)/3);
    AIN:=AIN-F*(COMPR-UDDI)/COMPR;
    AFIM:=AFIM-F*UDDI/COMPR;
  end
  else
  begin
    E:=ud*tam;
    M:=uddi/compr;
    H:=sqr(tam/(2*compr));
    NIN:=NIN-E*(1-3*sqr(M)-H+2*M*(sqr(M)+H));
    NFIM:=NFIM-E*(3*sqr(M)+H-2*M*(sqr(M)+H));
    MIN:=MIN-E*(UDDI*sqr((COMPR-UDDI)/COMPR)-H*(3*(COMPR-UDDI)-
       COMPR)/3);
    MFIM:=MFIM+E*((COMPR-UDDI)*sqr(M)-H*(3*UDDI-COMPR)/3);
  end;
end;

PROCEDURE ENGASTAXUV(AXVAR,AXVARNI,COMPR:double;
                Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
Begin
                (*NAO HA OPCAO PARA O SISTEMA GLOBAL*)
  IF AXVARNI=0 THEN
```

```
  BEGIN
    AIN:=AIN+AXVAR*(COMPR/3)/COMPR;
    AFIM:=AFIM+AXVAR*(COMPR*2/3)/COMPR;
    END
    ELSE
    BEGIN
      AIN:=AIN+AXVAR*(COMPR*2/3)/COMPR;
      AFIM:=AFIM+AXVAR*(COMPR/3)/COMPR;
    END;
  END;

PROCEDURE ENGASTAXC(OPTION:longint; AXIC,AXICDI,COMPR,
              COSENO,SENO:double;
            Var E,F,NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
BEGIN
  IF OPTION=0 THEN
  BEGIN       (* OSISTEMA ESTA NAS COORDENADAS GLOBAIS*)
          (*AXIC=CARGA CONCENTRADA NA COORDENADA GLOBAL 1*)
    F:=AXIC*COSENO;  (*COMPONENTE NA COORDENADA LOCAL 1*)
    E:=AXIC*SENO;    (*COMPONENTE NA COORDENADA LOCAL 2*)
    NIN:=NIN-
     E*(SQR((COMPR-AXICDI)/COMPR)*(3-2*((COMPR-AXICDI)/COMPR)));
    NFIM:=NFIM-E*(SQR(AXICDI/COMPR))*(3-2*(AXICDI/COMPR));
    MIN:=MIN-E*(AXICDI*SQR((COMPR-AXICDI)/COMPR));
    MFIM:=MFIM+E*((COMPR-AXICDI)*SQR(AXICDI/COMPR));
    AIN:=AIN-F*(COMPR-AXICDI)/COMPR;
    AFIM:=AFIM-F*AXICDI/COMPR;
  END
  ELSE
  BEGIN
    AIN:=AIN-AXIC*(COMPR-AXICDI)/COMPR;
    AFIM:=AFIM-AXIC*AXICDI/COMPR;
  END;
END;

Procedure ENGASTMB(MOMB,MOMBDI,COMPR:double;
            Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);
Begin
  nin:=nin-momb*6*mombdi*(compr-mombdi)/(compr*sqr(compr));
  nfim:=nfim+momb*6*mombdi*(compr-mombdi)/(compr*sqr(compr));
  min:=min-momb*(compr-mombdi)*(2-3*(compr-mombdi)/compr)/compr;
  mfim:=mfim-momb*mombdi*(2-3*mombdi/compr)/compr;
end;

Procedure Cabenos;
Var
R:real;
y:integer;
begin
  writeln(arqs,
'FOI ANALISADA A ESTRUTURA CUJOS DADOS FIGURAM NO ARQUIVO ',
      NOMEARQE,' ');
  writeln(arqs);
  writeln(arqs,'A matriz U armazenou',limite,' posicoes');
  writeln(arqs);
  Y:=sqr(3*N);
  R:=(limite/Y)*100;
  writeln(arqs,'A matriz de rigidez do caso analisado tem ',Y,' posicoes');
```

```
  writeln(arqs.' e portanto o metodo sparse armazenou '.R:5:2.' % da referida matriz de rigidez.');
  writeln(arqs.'OS RESULTADOS ENCONTRADOS FORAM:');
  writeln(arqs);
  writeln(arqs);
  writeln(arqs.'                    DESLOCAMENTOS');
  writeln(arqs);
  write(arqs.' NO    COORDENADA 1        COORDENADA 2');
  writeln(arqs.'          COORDENADA 3');
end;

Procedure Mostradesloc(C,D,E:double; g:longint; Var X:longint);
Var
A,y:integer;
begin
  write(arqs.' ',g:3,'   ',C:9:6,'          ',D:9:6);
  writeln(arqs.'          ',E:9:6);
end;

Procedure Roda  (B:longint; Topbar,barra:barptr; mbeta:matriz);
Var
T,U:double;
G:longint;
CA:Array[1..6] of double;

begin
   FOR G:=1 TO B DO
   BEGIN
     Tripbar(G,barra);
     T:=BARRA^.COSX;
     U:=BARRA^.COSY;
     matrizbeta(T,U,mbeta);
     CA[1]:=BARRA^.AXIN;
     CA[2]:=BARRA^.NORIN;
     CA[3]:=BARRA^.MOMIN;
     CA[4]:=BARRA^.AXFIM;
     CA[5]:=BARRA^.NORFIM;
     CA[6]:=BARRA^.MOFIM;
     barra^.axin:=mbeta[1,1]*CA[1]+mbeta[1,2]*CA[2]+
              mbeta[1,3]*CA[3]+mbeta[1,4]*CA[4];
     barra^.axin:=barra^.axin+mbeta[1,5]*CA[5]+mbeta[1,6]*CA[6];
     barra^.norin:=mbeta[2,2]*CA[2]+mbeta[2,1]*CA[1]+
                mbeta[2,3]*CA[3]+
                mbeta[2,4]*CA[4];
     barra^.norin:=barra^.norin+mbeta[2,5]*CA[5]+mbeta[2,6]*CA[6];
     barra^.momin:=mbeta[3,3]*CA[3]+mbeta[3,1]*CA[1]+
                mbeta[3,2]*CA[2]+
                mbeta[3,4]*CA[4];
     barra^.momin:=barra^.momin+mbeta[3,5]*CA[5]+
                mbeta[3,6]*CA[6];
     barra^.axfim:=mbeta[4,4]*CA[4]+mbeta[4,1]*CA[1]+
                mbeta[4,2]*CA[2]+
                mbeta[4,3]*CA[3];
     barra^.axfim:=barra^.axfim+mbeta[4,5]*CA[5]+
                mbeta[4,6]*CA[6];
     barra^.norfim:=mbeta[5,1]*CA[1]+mbeta[5,2]*CA[2]+
                 mbeta[5,3]*CA[3]+
                mbeta[5,4]*CA[4];
     barra^.norfim:=barra^.norfim+mbeta[5,5]*CA[5]+
```

```
                mbeta[5,6]*CA[6];
    barra^.mofim:=mbeta[6.1]*CA[1]+mbeta[6.2]*CA[2]+
                mbeta[6.3]*CA[3]+
                mbeta[6.4]*CA[4];
    barra^.mofim:=barra^.mofim+mbeta[6.6]*CA[6]+
                mbeta[6.5]*CA[5];
  end;
end;

Procedure Tripae (busk:longint; Var AE:AEptr);
begin
  while busk<AE^.pos do
  AE:=AE^.next;
  while busk>AE^.pos do
  Ae:=AE^.ant;
end;


Procedure Carga  (N,B:longint; Topbar,barra:barptr; mbeta:matriz;
            no,topno:noptr;    lastae,AE:aeptr;  Var topAE:aeptr);
Var
X,Y,I,busk,W:longint;
T,U:double;
G:longint;
AA:matriz;
CA:Array[1..6] of double;

begin
  barra:=topbar;
  FOR G:=1 TO B DO
  BEGIN
    T:=BARRA^.COSX;
    U:=BARRA^.COSY;
    matrizbeta(T,U,mbeta);
    For x:=1 to 6 do
    For y:=1 to 6 do
    AA[x,y]:=mbeta[y,x];
    CA[1]:=BARRA^.AXIN;
    CA[2]:=BARRA^.NORIN;
    CA[3]:=BARRA^.MOMIN;
    CA[4]:=BARRA^.AXFIM;
    CA[5]:=BARRA^.NORFIM;
    CA[6]:=BARRA^.MOFIM;
    barra^.axin:=AA[1,1]*CA[1]+AA[1,2]*CA[2]+AA[1,3]*CA[3]+
            AA[1,4]*CA[4];
    barra^.axin:=barra^.axin+AA[1,5]*CA[5]+AA[1,6]*CA[6];
    barra^.norin:=AA[2,2]*CA[2]+AA[2,1]*CA[1]+AA[2,3]*CA[3]+
            AA[2,4]*CA[4];
    barra^.norin:=barra^.norin+AA[2,5]*CA[5]+AA[2,6]*CA[6];
    barra^.momin:=AA[3,3]*CA[3]+AA[3,1]*CA[1]+AA[3,2]*CA[2]+
            AA[3,4]*CA[4];
    barra^.momin:=barra^.momin+AA[3,5]*CA[5]+AA[3,6]*CA[6];
    barra^.axfim:=AA[4,4]*CA[4]+AA[4,1]*CA[1]+AA[4,2]*CA[2]+
            AA[4,3]*CA[3];
    barra^.axfim:=barra^.axfim+AA[4,5]*CA[5]+AA[4,6]*CA[6];
    barra^.norfim:=AA[5,1]*CA[1]+AA[5,2]*CA[2]+AA[5,3]*CA[3]+
            AA[5,4]*CA[4];
    barra^.norfim:=barra^.norfim+AA[5,5]*CA[5]+AA[5,6]*CA[6];
```

```
  barra^.mofim:=AA[6,1]*CA[1]+AA[6.2]*CA[2]+AA[6.3]*CA[3]+
          AA[6,4]*CA[4];
  barra^.mofim:=barra^.mofim+AA[6.6]*CA[6]+AA[6.5]*CA[5];
  If g<>b then
  barra:=barra^.next;
END;
new(AE);
AE^.eVAL:=0;
AE^.next:=nil;
AE^.pos:=1;
FOR Y:=2 TO 3*N DO
begin
  lastAE:=AE;
  new(AE);
  lastAE^.ant:=AE;
  AE^.Eval:=0;
  AE^.pos:=Y;
  AE^.next:=lastAE;
end;
topAE:=AE;
I:=0;
FOR G:=1 TO B DO
BEGIN
  Tripbar(G,barra);
  busk:=BARRA^.noinicio*3-2;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.AXIN;
  busk:=BARRA^.noinicio*3-1;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.NORIN;
  busk:=BARRA^.noinicio*3;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.MOMIN;
  busk:=BARRA^.nofim*3-2;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.AXFIM;
  busk:=BARRA^.nofim*3-1;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.NORFIM;
  busk:=BARRA^.nofim*3;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.MOFIM;
END;
FOR X:=1 TO N DO
BEGIN
  i:=x;
  tripno(I,no);
  busk:=3*X-2;
  tripAE(busk,AE);
  AE^.Eval:=AE^.Eval+NO^.AXNO;
  busk:=3*X-1;
  tripAE(busk,AE);
  AE^.Eval:=AE^.Eval+NO^.norno;
  busk:=3*X;
  tripAE(busk,AE);
  AE^.Eval:=AE^.Eval+NO^.mono;
END;
FOR W:=1 TO N DO
```

```
  BEGIN
    I:=w;
    tripno(I,no);
    CASE NO^.RESTRI OF
    3:BEGIN
        busk:=3*W-2;
        tripAE(busk,AE);
        AE^.Eval:=0;
        busk:=3*W-1;
        tripAE(busk,AE);
        AE^.Eval:=0;
        busk:=3*W;
        tripAE(busk,AE);
        AE^.Eval:=0;
      END;
    2:BEGIN
        busk:=3*W-2;
        tripAE(busk,AE);
        AE^.Eval:=0;
        busk:=3*W-1;
        tripAE(busk,AE);
        AE^.Eval:=0;
      END;
    1:BEGIN
        busk:=3*W-1;
        tripAE(busk,AE);
        AE^.Eval:=0;
      END;
    4:BEGIN
        busk:=3*W-2;
        tripAE(busk,AE);
        AE^.Eval:=0;
      END;
    0:begin
        no^.restri:=0;
      end;
    END;
  END;
end;


BEGIN
  WRITELN('PROGRAMA PORTICO PLANO');
  WRITELN('versao sparse');
  WRITE ('ARQUIVO DE DADOS= ');
  writeln;
  READLN(NOMEARQE);
   ASSIGN(ARQE,NOMEARQE);RESET(ARQE);
  WRITELN(NOMEARQE);
  writeln('ARQUIVO DE SAIDA= ');
  readln(nomearqs);
  GetTime(hour,minute,second,sec100);
  tempo:=60*minute+second+sec100/100;
  ASSIGN(ARQS,NOMEARQS);REwrite(ARQS);
  WRITELN(NOMEARQS);
  READLN(ARQE.B);
  READLN(ARQE.N);
  new(no);
```

```
lastno:=no;
X:=0;
FOR I:=1 TO N DO
BEGIN
  READLN(ARQE.no^.nu.no^.ABSNO.no^.ORDNO.no^.RESTRI);
  If no^.restri<>0 then
  begin
    x:=x+1;
    gno[X]:=no;
  end;
  if I=1 then
  begin
    no^.next:=nil;
    new(no);
  end
  else
  begin
    if I<N then
    begin
      no^.next:=lastno;
      lastno^.ant:=no;
      lastno:=no;
      new(no);
    end
    else
    begin
      no^.next:=lastno;
      lastno^.ant:=no;
      no^.ant:=nil;
    end;
  end;
END;
maxres:=X;
topno:=no;
new(barra);
lastbarra:=barra;
primbar:=barra;
FOR G:=1 TO B DO
BEGIN
  READLN(ARQE.barra^.num.BARRA^.NOINICIO.BARRA^.NOFIM.
       BARRA^.MINERCIA.
       BARRA^.MODELAST.BARRA^.AREA);
  I:=barra^.noinicio;
  tripno(I.no);
  R:=-no^.ordno;
  D:=-no^.absno;
  I:=barra^.nofim;
  tripno(I.no);
  R:=R+no^.ordno;
  D:=D+no^.absno;
  C:=sqrt(sqr(R)+sqr(D));
  barra^.comp:=C;
  barra^.cosx:=D/C;
  barra^.cosy:=R/C;
  if g=1 then
  begin
    barra^.next:=nil;
    new(barra);
```

```
    end
    else
    begin
      if g<b then
      begin
        barra^.next:=lastbarra;
        lastbarra^.ant:=barra;
        lastbarra:=barra;
        new(barra);
      end
      else
      begin
        barra^.next:=lastbarra;
        lastbarra^.ant:=barra;
      end;
    end;
END;
topbar:=barra;
limax:=0;
linha:=0;
Z:=0;
max:=3*n;
while MAX>=1 do
begin
  linha:=linha+1;
  numbar:=0;
  barra:=primbar;
  For g:=1 to B do
  begin
    If (barra^.noinicio)*3-2=linha then
    begin
      numbar:=numbar+1;
      gba[numbar]:=barra;
    end;
    If (barra^.nofim)*3-2=linha then
    begin
      numbar:=numbar+1;
      gba[numbar]:=barra;
    end;
    barra:=barra^.ant;
  end;
  limax:=limax+max;
  limin:=limax-max;
  For contador:=1 to 3 do
  begin
    for bbb:=3*N-max+1 to 3*N do
    auxivet[bbb]:=0;
    g:=1;
    while G<=numbar do
    begin
      barra:=gba[G];
      assemblagem(N,B,limax,limin,barra,no,topno,max,maxres,
                  linha, numbar,gno,auxivet,G);
      g:=g+1;
    end;
    for y:=linha to 3*n do
    begin
      Z:=Z+1;
```

```
        if auxivet[y]<>0 then
        begin
          new(kglobal);
          If Y<>1  then
          begin
            kglobal^.next:=lastk;
            lastk^.ant:=kglobal;
          end
          else
          begin
            auxk:=kglobal;
            If linha=1 then
            kglobal^.next:=nil;
          end;
          kglobal^.kval:=auxivet[y];
          kglobal^.ordem:=Z;
          lastk:=kglobal;
          posicao:=posicao+1;
        end;
      end;
      If contador<3 then
      begin
        linha:=linha+1;
        max:=max-1;
        limax:=limax+max;
        limin:=limax-max;
      end;
    end;
    max:=max-1;
end;
topk:=kglobal;
limite:=posicao;
writeln('Foi construida a matriz de rigidez e armazenadas ', limite,' posicoes');
barra:=topbar;
FOR G:=1 TO B DO
BEGIN
  BARRA^.NORIN:=0;
  BARRA^.NORFIM:=0;
  BARRA^.AXIN:=0;
  BARRA^.AXFIM:=0;
  BARRA^.MOMIN:=0;
  BARRA^.MOFIM:=0;
  BARRA^.UV:=0;
  BARRA^.UVNI:=0;
  BARRA^.AXUV:=0;
  BARRA^.AXUVNI:=0;
  FOR X:=1 TO 4 DO
  BEGIN
    barra^.cc[x]:=0;
    barra^.di[x]:=0;
    barra^.axc[x]:=0;
    barra^.axcdi[x]:=0;
    barra^.ud[x]:=0;
    barra^.tam[x]:=0;
    barra^.uddi[x]:=0;
    barra^.mab[x]:=0;
    barra^.mabdi[x]:=0;
  end;
```

```
  If g<>b then
  barra:=barra^.next;
END;
READLN(ARQE.OPTION);  (*ESTA VARIAVEL. DA AO USUARIO A OPCAO DE
                  COLOCAR SEU CARREGAMENTO SEGUNDO O
                   SISTEMA
                  GLOBAL(0) DE COORDENADAS. OU LOCAL(1)*)
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.G);
    tripbar(G,barra);
    for x:=1 to 4 do
    BEGIN
      READ(ARQE.BARRA^.CC[X],BARRA^.DI[X]);
        (*CC:=CARGA CONCENTRADA NA COORDENADA GLOBAL 2*)
      E:=BARRA^.CC[X]*BARRA^.COSX; (*agora na coord.
                          local 2*)
      F:=BARRA^.CC[X]*BARRA^.COSY; (*coord. local 1*)
      CONC:=BARRA^.CC[X];
      DIST:=BARRA^.DI[X];
      remete;
      ENGASTCC(OPTION,E,F,COMPR,CONC,DIST,NIN,NFIM,MIN,
        MFIM, AIN,AFIM);
      recebe;
    END;
    READLN(ARQE);
  end;
end;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.G);
    tripBAR(G,BARRA);
    for x:=1 to 4 do
    begin
      READ(ARQE.BARRA^.AXC[X],BARRA^.AXCDI[X]);
      AXIC:=BARRA^.AXC[X];
      AXICDI:=BARRA^.AXCDI[X];
      COSENO:=BARRA^.COSX;
      SENO:=BARRA^.COSY;
      REMETE;
      ENGASTAXC(OPTION,AXIC,AXICDI,COMPR,COSENO,SENO,E,F,
          NIN,NFIM,MIN,MFIM,
        AIN,AFIM);
      RECEBE;
    end;
    READLN(ARQE);
  end;
END;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
```

```
  begin
    READ(ARQE,G);
    TripBAR(G,BARRA);
    read(arqe,total);(*total=1 significa que a carga se extende por toda a barra*)
    If total=1 then
    begin
      read(arqe,barra^.ud[1]);
      ud:=barra^.ud[1];
      tam:=barra^.comp;
      uddi:=tam/2;
      coseno:=barra^.cosx;
      seno:=barra^.cosy;
      remete;
      ENGASTud(OPTION,COMPR,ud,tam,uddi,COSENO,SENO,e,f,m,
            h,NIN,NFIM,MIN,
        MFIM,AIN,AFIM);
      recebe;
    end
    else
    begin
      for x:=1 to 4 do
      begin
        READ(ARQE,BARRA^.UD[X],BARRA^.TAM[X],
            BARRA^.UDDI[X]);
            (*UD:=CARGA UN. DISTRIB. COORDENADA GLOBAL
                  2*)
        ud:=BARRA^.UD[X];
        tam:=BARRA^.TAM[X];
        uddi:=BARRA^.UDDI[X];
        coseno:=BARRA^.COSX;
        seno:=BARRA^.COSY;
        remete;
        ENGASTud(OPTION,COMPR,ud,tam,uddi,COSENO,SENO,e,f,
            m,h,NIN,NFIM,MIN,
            MFIM,AIN,AFIM);
        recebe;
      end;
    end;
    READLN(ARQE);
  end;
END;
READLN(ARQE,W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE,G);
    TripBAR(G,BARRA);
    read(arqe,BARRA^.UV,BARRA^.UVNI);
    E:=BARRA^.UV*BARRA^.COSX;
    F:=BARRA^.UV*BARRA^.COSY;
    COSENO:=BARRA^.COSX;
    SENO:=BARRA^.COSY;
    UVAR:=BARRA^.UV;
    remete;
    UNI:=BARRA^.UVNI;
    ENGASTUV(OPTION,E,F,COMPR,COSENO,SENO,UVAR,UNI,NIN,NFIM,
        MIN,
```

```
          MFIM,AIN,AFIM);
      recebe;
      readln(arqe);
    end;
  end;
  READLN(ARQE,W);
  if w<>0 then
  begin
    for y:=1 to w do
    begin
      READ(ARQE,G);
      TripBAR(G.BARRA);
      for x:=1 to 4 do
      begin
        READ(ARQE,BARRA^.MAB[X],BARRA^.MABDI[X]);
        momb:=BARRA^.MAB[X];
        mombdi:=BARRA^.MABDI[X];
        REMETE;
        ENGASTMB(MOMB,MOMBDI,COMPR,NIN,NFIM,MIN,MFIM,AIN,
                 AFIM);
        RECEBE;
      end;
      READLN(ARQE);
    end;
  end;
  READLN(ARQE,W);
  if w<>0 then
  begin
    for y:=1 to w do
    READ(ARQE,G);
    TripBAR(G,BARRA);
    READLN(ARQE,BARRA^.AXUV,BARRA^.AXUVNI);
       (*NAO HA OPCAO PARA O SISTEMA GLOBAL*)
    AXVAR:=BARRA^.AXUV;
    AXVARNI:=BARRA^.AXUVNI;
    REMETE;
    ENGASTAXUV(AXVAR,AXVARNI,COMPR,
               NIN,NFIM,MIN,MFIM,AIN,AFIM);
    RECEBE;
  end;
  FOR I:=1 TO N DO
  BEGIN
     tripno(I,no);
    NO^.AXNO:=0;
    NO^.NORNO:=0;
    NO^.MONO:=0;
  END;
  READLN(ARQE,W);
  if w<>0 then
  begin
    for y:=1 to w do
    begin
      READ(ARQE,I);
       tripno(I,no);
      READLN(arqe,NO^.AXNO,NO^.NORNO,NO^.MONO);
    end;
  end;
  Carga (N,B,topbar,barra,mbeta,no,topno,lastAE,AE,topAE);
```

```
writeln('Foi construido o vetor de carga');
ELE[1]:=1;
FOR X:=2 TO 3*N DO
ELE[X]:=0;
X:=1;
G:=1;
sinal:=0;
ae:=topae;
FOR Y:=1 TO 3*N-1 DO
BEGIN
  kglobal:=auxk;
  Z:=((X-1)*3*N-X*(X-1) DIV 2)+X;      (*W*)
  while Z<kglobal^.ordem do
  kglobal:=kglobal^.next;
  while Z>kglobal^.ordem do
  kglobal:=kglobal^.ant;
  val1:=kglobal^.kval;
  Z:=((Y-1)*3*N-Y*(Y-1) div 2)+3*N;
  While Kglobal^.ordem<Z do
  kglobal:=kglobal^.ant;
  If kglobal^.ordem>auxk^.ordem then
  auxk:=kglobal;
  If kglobal^.ordem=Z then
  begin
    auxivet[3*N]:=kglobal^.kval;
    lastk:=kglobal;
    kglobal:=kglobal^.next;
    zero:=lastk^.ordem-kglobal^.ordem-1;
    sinal:=1;
  end
  else
  begin
    auxivet[3*N]:=0;
    proxk:=kglobal^.next;
    zero:=Z-proxk^.ordem-1;
  end;
  For bbb:=3*N-1 downto Y do
  begin
    If Zero=0 then
    begin
      If sinal=1 then
      begin
        auxivet[bbb]:=kglobal^.kval;
        proxk:=kglobal^.next;
        zero:=kglobal^.ordem-proxk^.ordem-1;
        sinal:=0;
      end
      else
      begin
        proxk:=kglobal^.next;
        auxivet[bbb]:=proxk^.kval;
        If bbb<>Y then
        begin
          kglobal:=kglobal^.next;
          proxk:=kglobal^.next;
          zero:=kglobal^.ordem-proxk^.ordem-1;
        end;
      end;
```

```
  end
  else
  begin
    auxivet[bbb]:=0;
    zero:=zero-1;
  end;
end;
WHILE G<3*N DO
BEGIN
  ELE[G+1]:=-auxivet[G+1]/val1;
  G:=G+1;
END;
busk:=X;
tripAE(busk,AE);
D:=AE^.Eval;
FOR A:=X+1  TO 3*N DO
begin
  busk:=A;
  tripAE(busk,AE);
  AE^.Eval:=AE^.Eval+D*ELE[A];
end;
sinal:=0;
bip:=0;
FOR A:=Y+1  TO 3*N DO
begin
  If ELE[A]<>0 then
  begin
    contador:=0;
    FOR K:=A TO 3*N DO
    BEGIN
      alarme:=0;
      If auxivet[3*N-contador]<>0 then
      begin
        Z:=((A-1)*3*N-A*(A-1) DIV 2)-K+3*N+A;
        If Z<kglobal^.ordem then
        begin
          bip:=1;
          while Z<kglobal^.ordem do
          kglobal:=kglobal^.next;
        end
        else
        begin
          while Z>kglobal^.ordem do
          kglobal:=kglobal^.ant;
        end;
        If kglobal^.ordem=Z then
        begin
          val1 :=kglobal^.kval;
          bip:=0;
        end
        else
        begin
          val1 :=0;
          alarme:=1;
        end;
        valor:=val1+ELE[A]*auxivet[3*N-contador];
        If alarme=0 then
        begin
```

```
            kglobal^.kval:=valor;
          end
          else
          begin
            lastk:=kglobal;
            New(Kglobal);
            kglobal^.kval:=valor;
            kglobal^.ordem:=Z;
            If bip=1 then
            begin
              kglobal^.next:=lastk;
              kglobal^.ant:=lastk^.ant;
              proxk:=lastk^.ant;
              proxk^.next:=kglobal;
              lastk^.ant:=kglobal;
              bip:=0;
            end
            else
            begin
              kglobal^.ant:=lastk;
              kglobal^.next:=lastk^.next;
              proxk:=lastk^.next;
              proxk^.ant:=kglobal;
              lastk^.next:=kglobal;
            end;
            limite:=limite+1;
            alarme:=0;
          end;
        end;
        contador:=contador+1;
      end;
    end;
  end;
  X:=X+1;
  G:=X;
end;
writeln('Foi construida a matriz U que armazenou '.limite. 'posicoes');
AE:=topAE;
kglobal:=topk;
R:=0;
ELE[3*N]:=TOPAE^.EVAL/TOPK^.KVAL;
Z:=((3*N-2)*3*N-(3*N-1)*(3*N-2) DIV 2)+3*N;
AE:=AE^.next;
kglobal:=kglobal^.next;
If Z=kglobal^.ordem then
begin
  valor:=kglobal^.kval;
end
else
begin
  valor:=0;
end;
AE^.EVAL:=AE^.EVAL-valor*ELE[3*N];
Z:=((3*N-2)*3*N-(3*N-1)*(3*N-2) DIV 2)+3*N-1;
while Z<kglobal^.ordem do
kglobal:=kglobal^.next;
If Z=kglobal^.ordem then
begin
```

```
  valor:=kglobal^.kval;
end
else
begin
  valor:=0;
end;
ELE[3*n-1]:=AE^.EVAL/valor;
bip:=0;
Z:=((3*N-3)*3*N-(3*N-3)*(3*N-2) div 2)+3*N;
while Z<kglobal^.ordem do
kglobal:=kglobal^.next;
while Z>kglobal^.ordem do
kglobal:=kglobal^.ant;
For K:=3*N-1 downto 2 do
begin
  AE:=AE^.next;
  If z=kglobal^.ordem then
  begin
    valor:=kglobal^.kval;
    AE^.eval:=Ae^.eval-valor*ELE[3*N];
    proxk:=kglobal^.next;
    zero:=kglobal^.ordem-proxk^.ordem-1;
  end
  else
  begin
    zero:=Z-kglobal^.ordem-1;
    bip:=1;
  end;
  Z:=Z-1;
  Y:=1;
  A:=3*N-K+1;
  while Y<A do
  begin
    If zero=0 then
    begin
      If bip=0 then
      begin
        kglobal:=kglobal^.next;
        valor:=kglobal^.kval;
        R:=R+valor*ELE[3*N-Y];
        proxk:=kglobal^.next;
        zero:=kglobal^.ordem-proxk^.ordem-1;
        Z:=Z-1;
      end
      else
      begin
        proxk:=kglobal^.next;
        If Z=proxk^.ORDEM THEN
        begin
          kglobal:=kglobal^.next;
          valor:=kglobal^.kval;
          R:=R+valor*ELE[3*N-Y];
          proxk:=kglobal^.next;
          zero:=kglobal^.ordem-proxk^.ordem-1;
          Z:=Z-1;
          bip:=0;
        end
        else
```

```
          begin
            valor:=kglobal^.kval;
            proxk:=kglobal^.next;
            zero:=kglobal^.ordem-proxk^.ordem-1;
            bip:=0;
            R:=R+valor*ELE[3*N-Y];
            Z:=Z-1;
          end;
        end;
      end
      else
      begin
        zero:=zero-1;
        Z:=Z-1;
      end;
      Y:=Y+1;
    end;
    If Z=kglobal^.ordem then
    begin
      valor:=kglobal^.kval;
    end
    else
    begin
      kglobal:=kglobal^.next;
      valor:=kglobal^.kval;
    end;
    ELE[K-1]:=(AE^.eval-R)/valor;
    If K>2 then
    begin
      R:=0;
      proxk:=kglobal^.next;
      zero:=kglobal^.ordem-proxk^.ordem-1;
      Z:=Z-1;
      kglobal:=kglobal^.next;
    end;
  end;
  FOR G:=1 TO N DO
  BEGIN
    I:=G;
    tripno(I.no);
    NO^.DAX:=ELE[3*G-2];
    NO^.DNOR:=ELE[3*G-1];
    NO^.ROT:=ELE[3*G];
  end;
  cabenos;
  x:=0;
  for g:=1 to N do
  begin
    I:=G;
    tripno(I.no);
    C:=no^.dax;
    D:=no^.dnor;
    E:=no^.rot;
    mostradesloc(C,D,E,g,X);
  end;
  Roda (B.topbar,barra.mbeta);
  FOR G:=1 TO B DO
  BEGIN
```

```
  TripBAR(G.BARRA);
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T.U.MBETA);
  FOR X:=1 TO 6 DO
  BEGIN
    I:=BARRA^.NOINICIO;
     tripno(I.no);
    R:=MBETA[X.1]*NO^.DAX+MBETA[X.2]*NO^.DNOR+
       MBETA[X.3]*NO^.ROT;
    I:=BARRA^.NOFIM;
     tripno(I.no);
    CA[X]:=R+MBETA[X.4]*NO^.DAX+MBETA[X.5]*NO^.DNOR+
        MBETA[X.6]*NO^.ROT;
  END;
  P:=BARRA^.MODELAST;
  J:=BARRA^.MINERCIA;
  S:=BARRA^.AREA;
  C:=BARRA^.COMP;
  MATRIZRIG(C.J.S.P.MRIGB);
  FOR X:=1 TO 6 DO
  BARRA^.AXIN:=BARRA^.AXIN+MRIGB[1.X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.NORIN:=BARRA^.NORIN+MRIGB[2.X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.MOMIN:=BARRA^.MOMIN+MRIGB[3.X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.AXFIM:=BARRA^.AXFIM+MRIGB[4.X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.NORFIM:=BARRA^.NORFIM+MRIGB[5.X]*CA[X];
  FOR X:=1 TO 6 DO
  BARRA^.MOFIM:=BARRA^.MOFIM+MRIGB[6.X]*CA[X];
END;
  WRITELN(ARQS);
writeln(arqs.'                    ESFORCOS NAS BARRAS');
WRITELN(ARQS);
writeln(arqs.'  BARRA  NO INICIAL                NO FINAL');
writeln(arqs.'        AXIAL   NORMAL   MOMENTO  AXIAL   NORMAL   MOMENTO');
G:=1;
FOR G:=1 TO B DO
BEGIN
  tripbar(G.barra);
  WRITE(ARQS.'  '.g:3.'  '.BARRA^.AXIN:8:2.'  '.BARRA^.NORIN:8:2);
  WRITE(ARQS.'  '.BARRA^.MOMIN:8:2.'  '.BARRA^.AXFIM:8:2);
  WRITELN(ARQS.'  '.BARRA^.NORFIM:8:2.'  '.BARRA^.MOFIM:8:2);
END;
FOR I:=1 TO N DO
BEGIN
   tripno(I.no);
  NO^.REAX:=0;
  NO^.RENOR:=0;
  NO^.REMON:=0;
END;
FOR G:=1 TO B DO
BEGIN
  TripBAR(G.BARRA);
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
```

```
      MATRIZBETA(T,U,MBETA);
      FOR X:=1 TO 6 DO
      FOR K:=1 TO 6 DO
      AA[X,K]:=MBETA[K,X];
      FOR X:=1 TO 6 DO
      BEGIN
        R:=AA[X,1]*BARRA^.AXIN+AA[X,2]*BARRA^.NORIN+
            AA[X,3]*BARRA^.MOMIN;
        CA[X]:=R+AA[X,4]*BARRA^.AXFIM+AA[X,5]*BARRA^.NORFIM+
             AA[X,6]*BARRA^.MOFIM;
      END;
      I:=BARRA^.NOINICIO;
       tripno(I,no);
      NO^.REAX:=NO^.REAX+CA[1];
      NO^.RENOR:=NO^.RENOR+CA[2];
      NO^.REMON:=NO^.REMON+CA[3];
      I:=BARRA^.NOFIM;
       tripno(I,no);
      NO^.REAX:=NO^.REAX+CA[4];
      NO^.RENOR:=NO^.RENOR+CA[5];
      NO^.REMON:=NO^.REMON+CA[6];
    END;
    x:=0;
    writeln(arqs);
    writeln(arqs,'                    REACOES NOS APOIOS');
   writeln(arqs);
   write(arqs,'  NO   COORDENADA 1       COORDENADA 2');
   writeln(arqs,'          COORDENADA 3');
    FOR G:=1 TO N DO
    BEGIN
      I:=G;
      tripno(I,no);
      IF NO^.RESTRI>0 THEN
      BEGIN
        write(arqs,'  ',G:3,'      ',NO^.REAX:9:2);
        WRITE(arqs,'          ',no^.renor:9:2,'          ');
        WRITELN(arqs,no^.remon:9:2);
      END;
    END;
      t1:=tempo;
    GetTime(hour,minute,second,sec100);
    writeln('hora= ',hour,'  ',minute,'  ',second,'  ',sec100);
    tempo:=60*minute+second+sec100/100;
    t2:=tempo-t1;
    writeln(arqs,'O tempo total de processamento utilizado pelo metodo');
    writeln(arqs,'sparse foi de ',t2:7:2,' segundos.');
    close(arqs);
    LIMPA;
    for g:=1 to 6 do
    writeln;
    writeln('Foi analisada a estrutura constante do arquivo. ',nomearqe,'.');
    writeln('Os resultados encontrados estao no arquivo. ',nomearqs);
     writeln;
    writeln;
    writeln('Para sair digite qualquer tecla');
    repeat until keypressed;
end.
```

ANEXO 4
Listagem do Programa Versão "Skyline"

```
PROGRAM PORTICO_PLANO;     (*armazena metade da matriz global dinamicamente
                           atraves do processo skyline*)
uses crt,dos;
type
   matriz=array[1..6,1..6] of double;
   tarray=array[1..4] of double;
   vetors=array[1..6] of double;
   vetor =array[1..2000] of double;
   vetint=array[1..1200] of longint;
   arquivo=text;
   AEptr=^AErec;
   AErec=record
         Eval:double;
         pos:longint;
         next:AEptr;
         ant:AEptr;
   end;
   kbalptr=^krec;
   krec=record
         kval:double;
         ordem:longint;
         ant:kbalptr;
         next:kbalptr;
   end;
   barptr=^barrec;
   barrec=record
          noinicio:longint;
          nofim:longint;
          minercia:double;
          modelast:double;
          area:double;
          comp:double;
          cosx:double;
          cosy:double;
          cc:tarray;   (*carga concentrada*)
          di:tarray;   (*distancia do no inicio ao ponto de
                       aplicacao*)
          df:tarray;   (*dist. do no fim ao ponto de
                       aplicacao*)
          ud:tarray;   (*carga uniformemnte distribuida*)
          uddf:tarray; (*dist. do no fim ao ponto medio
                       da carga*)
          uddi:tarray; (*dist. do no inicio ao ponto
                       medio da carga*)
          tam:tarray;  (*comprimento da carga uniformemente
                       dist.*)
          uv:double;     (*valor maior da carga
                       uniformemente variada*)
          uvni:double;   (*valor da uv no no inicio*)
          axc:tarray;  (*carga axial aplicada na barra*)
          axcdi:tarray;(*distancia do no inicio ao ponto de
                       aplicacao*)
```

```
        axcdf: tarray;(*distancia do no fim ao ponto de
                        aplicacao
                    da carga*)
        axud:tarray; (**)
        atam:tarray;
        axudi:tarray;
        axudf:tarray;
        axuv:double;
        axuvni:double;
        mab:tarray;
        mabdi:tarray;
        mabdf:tarray;
        norin:double;
        norfim:double;
        axin:double;
        axfim:double;
        momin:double;
        mofim:double;
        ant:barptr;
        next:barptr;
        num:longint;
    end;
    noptr=^norec;
    norec=record
        nu:longint;
        absno:double;
        ordno:double;
        norno:double;
        axno:double;
        mono:double;
        restri:integer;
        dax:double;
        dnor:double;
        rot:double;
        reax:double;
        renor:double;
        remon:double;
        ant:noptr;
        next:noptr;
    end;

     guarda=array[1..20] of noptr;
     guarba=array[1..20] of barptr;


var
lastAE,AE,topAE:AEptr;
primbar,lastbarra,barra,topbar:barptr;
lastno,noi,nof,no,topno:noptr;
proxk,auxk,KGLOBAL,lastk,topk:kbalptr;
gno:guarda;
gba:guarba;
A,B,G,I,K,L,N,X,Y,W,Z,NI,NF,AAA,GG,OPTION,posicao,busk,numno,limax,limin,
  total,bip,nulo,zero,bbb,contador,limite,sinal,max,maxres,
        iaux,linha,numbar:longINT;
C,D,E,F,H,J,M,O,P,Q,R,S,T,U,CONC,COMPR,COSENO,SENO,UVAR,NIN,NFIM,MIN,MFIM,
    AXIC,AXICDI,AXVAR,AXVARNI,UNI,AIN,AFIM,DIST,ud,tam,uddi,
    momb,mombdi,valor,val1,val2:double;
```

```
t1,t2,sec, hund :real;
tempo:real;
Hour, Minute, Second, Sec100: word;
CC,AA,BB,MBETA,MRIGB: MATRIZ;
auxivet,ELE:VETOR;
veth:vetint;
CA:VETORS;
ARQE,ARQS:ARQUIVO;
NOMEARQE,NOMEARQS:STRING;


procedure remete;
begin
  nin:=barra^.norin;
  nfim:=barra^.norfim;
  ain:=barra^.axin;
  afim:=barra^.axfim;
  min:=barra^.momin;
  mfim:=barra^.mofim;
  compr:=barra^.comp;
end;
procedure recebe;
begin
  barra^.norin:=nin;
  barra^.norfim:=nfim;
  barra^.axin:=ain;
  barra^.axfim:=afim;
  barra^.momin:=min;
  barra^.mofim:=mfim;
end;

procedure Limpa;
begin
   lastk:=topk^.next;
   kglobal:=topk;
   while kglobal^.next<>nil do
   begin
     dispose(kglobal);
     kglobal:=lastk;
     lastk:=kglobal^.next;
   end;
   barra:=topbar;
   lastbarra:=topbar^.next;
   while barra^.next<>nil do
   begin
     dispose(barra);
     barra:=lastbarra;
     lastbarra:=barra^.next;
   end;
   no:=topno;
   lastno:=no^.next;
   while no^.next<>nil do
   begin
     dispose(no);
     no:=lastno;
     lastno:=no^.next;
   end;
end;
```

```
procedure matrizbeta(t,u:double; Var mbeta:matriz);
begin
  MBETA[1,1]:=T;
  MBETA[1,2]:=U;
  MBETA[1,3]:=0;
  MBETA[1,4]:=0;
  MBETA[1,5]:=0;
  MBETA[1,6]:=0;
  MBETA[2,1]:=-U;
  MBETA[2,2]:=T;
  MBETA[2,3]:=0;
  MBETA[2,4]:=0;
  MBETA[2,5]:=0;
  MBETA[2,6]:=0;
  MBETA[3,1]:=0;
  MBETA[3,2]:=0;
  MBETA[3,3]:=1;
  MBETA[3,4]:=0;
  MBETA[3,5]:=0;
  MBETA[3,6]:=0;
  MBETA[4,1]:=0;
  MBETA[4,2]:=0;
  MBETA[4,3]:=0;
  MBETA[4,4]:=T;
  MBETA[4,5]:=U;
  MBETA[4,6]:=0;
  MBETA[5,1]:=0;
  MBETA[5,2]:=0;
  MBETA[5,3]:=0;
  MBETA[5,4]:=-U;
  MBETA[5,5]:=T;
  MBETA[5,6]:=0;
  MBETA[6,1]:=0;
  MBETA[6,2]:=0;
  MBETA[6,3]:=0;
  MBETA[6,4]:=0;
  MBETA[6,5]:=0;
  MBETA[6,6]:=1;
END;

procedure matrizrig(C,J,S,P:double;Var mrigb:matriz);
     (*C=COMP, J=MOM. DE INERCIA, S=AREA, P=MODULO DE EL.*)
begin
  MRIGB[1,1]:=P*S/C;
  MRIGB[1,2]:=0;
  MRIGB[1,3]:=0;
  MRIGB[1,4]:=-P*S/C;
  MRIGB[1,5]:=0;
  MRIGB[1,6]:=0;
  MRIGB[2,1]:=0;
  MRIGB[2,2]:=12*P*J/(SQR(C)*C);
  MRIGB[2,3]:=6*P*J/(SQR(C));
  MRIGB[2,4]:=0;
  MRIGB[2,5]:=-12*P*J/(SQR(C)*C);
  MRIGB[2,6]:=6*P*J/(SQR(C));
  MRIGB[3,1]:=0;
  MRIGB[3,2]:=6*P*J/(SQR(C));
```

```
  MRIGB[3,3]:=4*P*J/C;
  MRIGB[3,4]:=0;
  MRIGB[3,5]:=-6*P*J/(SQR(C));
  MRIGB[3,6]:=2*P*J/C;;
  MRIGB[4,1]:=-P*S/C;
  MRIGB[4,2]:=0;
  MRIGB[4,3]:=0;
  MRIGB[4,4]:=P*S/C;
  MRIGB[4,5]:=0;
  MRIGB[4,6]:=0;
  MRIGB[5,1]:=0;
  MRIGB[5,2]:=-12*P*J/(SQR(C)*C);
  MRIGB[5,3]:=-6*P*J/(SQR(C));
  MRIGB[5,4]:=0;
  MRIGB[5,5]:=12*P*J/(SQR(C)*C);
  MRIGB[5,6]:=-6*P*J/(SQR(C));
  MRIGB[6,1]:=0;
  MRIGB[6,2]:=6*P*J/(SQR(C)) ;
  MRIGB[6,3]:=2*P*J/C;
  MRIGB[6,4]:=0;
  MRIGB[6,5]:=-6*P*J/(SQR(C)) ;
  MRIGB[6,6]:=4*P*J/C;
end;

procedure produmax(aa,bb:matriz; Var cc:matriz) ;
Var
x,y,k:integer;

begin
  for x:=1 to 6 do
  for y:=1 to 6 do
  begin
    cc[x,y]:=0;
     for k:=1 to 6 do
    begin
      cc[x,y]:=cc[x,y]+aa[x,k]*bb[k,y];
    end;
  end;
end;

procedure Tripno(I:longint; Var no:noptr);
begin
  while I<no^.nu do
  no:=no^.next;
  while I>no^.nu do
  no:=no^.ant;
end;

procedure assemblagem(n,B,limax,limin:longint; barra:barptr;
     no,topno:noptr;  max,maxres,linha,numbar:longint;
     gno:guarda;  Var auxivet:vetor; Var G:longint);

Var
x,k,y,ni,nf,ww,yy,v,I,sinal,numno,bip,lin,col:longint;
cont,Ind,xx:longint;
cc,bb,aa,mbeta,mrigb:matriz;
t,u,p,s,j,C,r,d:double;
inno,fino:noptr;
```

```
Procedure vecalc(lin.col:longint; Var xx:longint);
begin
  XX:=((lin-1)*3*N-lin*(lin-1) DIV 2) +col;
end;
Procedure ConstCc;
Var
x.y.k:longint;
BEGIN
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T.U.MBETA);
  P:=BARRA^.MODELAST;
  J:=BARRA^.MINERCIA;
  S:=BARRA^.AREA;
  C:=BARRA^.COMP;
  MATRIZRIG(C.J.S.P.MRIGB);
  FOR X:=1 TO 6 DO
  FOR K:=1 TO 6 DO
  AA[X.K]:=MBETA[K.X];
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BB[X.Y]:=MRIGB[X.Y];
  PRODUMAX(AA.BB.CC);
  FOR X:=1 TO 6 DO
  FOR Y:=1 TO 6 DO
  BEGIN
    AA[X.Y]:=CC[X.Y];
    BB[X.Y]:=MBETA[X.Y];
  END;
  PRODUMAX(AA.BB.CC);

end;
Procedure Dirichilet;
Var
Y.X.bbb.sinal:longint;
begin
  Y:=2;
  sinal:=0;
  Case no^.restri of
  3:begin
      while Y>=0 do
      begin
        X:=2;
        While X>=0 do
        begin
          lin:=numno*3-X;
          col:=numno*3-y;
          If col=lin then
          begin
            vecalc(lin.col.xx);
            If (XX<=limax) and (xx>limin) then
            begin
              auxivet[xx-limin+3*N-max]:=1;
              For bbb:=xx-limin+3*N-max+1 to 3*N do
              auxivet[bbb]:=0;
              G:=numbar+1;
              sinal:=1;
```

```
            X:=-1;
          end
          else
            x:=x-1;
        end
        else
          X:=X-1;
      end;
      If sinal=1 then
      begin
        Y:=-1;
      end
      else
      begin
        Y:=Y-1;
      end;
    end;
end;
2:begin
    while Y>=1 do
    begin
      X:=2;
      while X>=1 do
      begin
        lin:=numno*3-X;
        col:=numno*3-y;
        If col=lin then
        begin
          vecalc(lin.col.xx);
          If (XX<=limax) and (xx>limin) then
          begin
            auxivet[xx-limin+3*N-max]:=1;
            For bbb:=xx-limin+3*N-max+1 to 3*N do
            auxivet[bbb]:=0;
            bip:=1;
            G:=numbar+1;
            X:=0;
            sinal:=1;
          end
          else
            x:=x-1;
        end
        else
          X:=X-1;
      end;
      If sinal=1 then
      begin
        Y:=-1;
      end
      else
      begin
        Y:=Y-1;
      end;
    end;
end;
1:begin
    lin:=numno*3-1;
    col:=numno*3-1;
```

```
      If col=lin then
      begin
        vecalc(lin,col,xx);
        If XX<=limax then
        If xx>limin then
        begin
          auxivet[xx-limin+3*N-max]:=1;
          For bbb:=xx-limin+3*N-max+1 to 3*N do
          auxivet[bbb]:=0;
          G:=numbar+1;
        end;
      end;
    end;
  end;
end;
Procedure Assem;
Var
X,Y:integer;
begin
  For X:=2 downto 0 do
  For Y:=X downto 0 do
  begin
    lin:=ni*3-X;          {4}
    col:=ni*3-y;
    vecalc(lin,col,xx);
    If XX<=limax then
    If xx>limin then
    begin
      If sinal=0 then
      begin
        constCC;
        sinal:=1;
      end;
      auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                        cc[3-x,3-y];
    end;
    lin:=nf*3-X;        {3}
    col:=nf*3-y;
    vecalc(lin,col,xx);
    If XX<=limax then
    If xx>limin then
    begin
      If sinal=0 then
      begin
        constCC;
        sinal:=1;
      end;
      auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                          cc[6-x,6-y];
    end;
  end;
  For X:=2 downto 0 do
  For Y:=2 downto 0 do
  begin
    lin:=ni*3-X;   {1}
    col:=nf*3-y;
    If col>=lin then
    begin
```

```
      vecalc(lin,col,xx);
      If XX<=limax then
      If xx>limin then
      begin
        If sinal=0 then
        begin
          constCC;
          sinal:= 1;
        end;
        auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                        cc[3-x,6-y] ;
      end;
    end;
    lin:=nf*3-X;       {2}
    col:=ni*3-y;
    If col>=lin then
    begin
      vecalc(lin,col,xx);
      If XX<=limax then
      If xx>limin then
      begin
        If sinal=0 then
        begin
          constCC;
          sinal:= 1;
        end;
        auxivet[xx-limin+3*N-max]:=auxivet[xx-limin+3*N-max]+
                        cc[6-x,3-y];
      end;
    end;
  end;
end;
BEGIN
  sinal:=0;
  NI:=BARRA^.NOINICIO;
  NF:=BARRA^.NOFIM;
  I:=Barra^.noinicio;
  tripno(I.no);
  inno:=no;
  I:=Barra^.nofim;
  tripno(I.no);
  fino:=no;
  If ni<nf then
  begin
    lin:=ni*3-2;          {4}
    col:=ni*3-2;
    vecalc(lin,col,xx);
    If XX<=limax then
    begin
      lin:=nf*3;
      col:=nf*3;
      vecalc(lin,col,xx);
      If XX>=limin then
      begin
        assem;
      end;
    end;
  end
```

```
else
begin
  lin:=nf*3-2;          {4}
  col:=nf*3-2;
  vecalc(lin,col,xx);
  If XX<=limax then
  begin
    lin:=ni*3;
    col:=ni*3;
    vecalc(lin,col,xx);
    If XX>=limin then
    begin
      assem;
    end;
  end;
end;
if fino^.restri<>0 then
begin
   no:=fino;
   numno:=barra^.nofim;
   richilet;
end;
If inno^.restri<>0 then
begin
  no:=inno;
  numno:=barra^.noinicio;
  richilet;
end;
If G=numbar then
begin
  For K:=1 to maxres do
  begin
    iaux:=gno[k]^.restri;
    Case iaux of
    3:begin
        For Y:=2 downto 0 do
        begin
          cont:=gno[k]^.nu;
          If (cont*3-Y)>(3*N-max+1) then
          begin
            col:=cont*3-Y;
            lin:=linha;
            vecalc(lin,col,xx);
            auxivet[xx-limin+3*N-max]:=0;
          end;
        end;
      end;
    2:begin
        For Y:=2 downto 1 do
        begin
          cont:=gno[k]^.nu;
          If (cont*3-Y)>(3*N-max+1) then
          begin
            col:=cont*3-Y;
            lin:=linha ;
            vecalc(lin,col,xx);
            auxivet[xx-limin+3*N-max]:=0;
          end;
```

```
          end;
        end;
       1:begin
          cont:=gno[k]^.nu;
          If (cont*3-1)>(3*N-max+1) then
          begin
            col:=cont*3-1;
            lin:=linha;
            vecalc(lin,col,xx);
            auxivet[xx-limin+3*N-max]:=0;
          end;
        end;
      end;
    end;
  end;
end;



procedure ENGASTUV(OPTION,E,F,COMPR,COSENO,SENO,UVAR,UNI:double;
            Var NIN,NFIM,MIN,MFIM,AIN,AFIM:double);

begin

  IF OPTION=0 THEN
  BEGIN              (*SITEMA GLOBAL DE COORD.*)
    IF UNI=0 THEN
    BEGIN
      NIN:=NIN-(3/20)*E*COMPR*COSENO;
      NFIM:=NFIM-(7/20)*E*COMPR*COSENO;
      AIN:=AIN+(F*COMPR*SENO/2)*(COMPR*SENO/3)/COMPR;
      AFIM:=AFIM+(F*COMPR*SENO)*(COMPR*SENO/3)/COMPR;
      MIN:=MIN-E*SQR(COMPR*COSENO)/30;
      MFIM:=MFIM+E*SQR(COMPR*COSENO)/20;
    END
    ELSE
    BEGIN
      NIN:=NIN-(7/20)*E*COMPR*COSENO;
      NFIM:=NFIM-(3/20)*E*COMPR*COSENO;
      AIN:=AIN+(F*COMPR*SENO)*(COMPR*SENO/3)/COMPR;
      AFIM:=AFIM+(F*COMPR*SENO/2)*(COMPR*SENO/3)/COMPR;
      MIN:=MIN-E*SQR(COMPR*COSENO)/20;
      MFIM:=MFIM+E*SQR(COMPR*COSENO)/30;
    END;
  END
  ELSE
  BEGIN          (*SISTEMA LOCAL*)
    IF UNI=0 THEN
    BEGIN
      NIN:=NIN-(3/20)*UVAR*COMPR;
      NFIM:=NFIM-(7/20)*UVAR*COMPR;
      MIN:=MIN-UVAR*SQR(COMPR)/30;
      MFIM:=MFIM+UVAR*SQR(COMPR)/20;
    END
    ELSE
    BEGIN
      NIN:=NIN-(7/20)*UVAR*COMPR;
      NFIM:=NFIM-(3/20)*UVAR*COMPR;
```

```
        MIN:=MIN-UVAR*SQR(COMPR)/20;
        MFIM:=MFIM+UVAR*SQR(COMPR)/30;
      END;
   END;
END;

procedure ENGASTCC(OPTION.E.F.COMPR.CONC.DIST:double;
              Var NIN.NFIM.MIN.MFIM.AIN.AFIM:double);
var
cont:longint;
begin
  if option=0 then
  begin
   nin:=nin-e*(sqr((compr-dist)/compr)*(3-2*(compr-dist)/compr));
    nfim:=nfim-e*(sqr(dist/compr)*(3-2*dist/compr));
    min:=min-e*(dist*sqr((compr-dist)/compr));
    mfim:=mfim+e*(compr-dist)*sqr(dist/compr);
    ain:=ain-f*(compr-dist)/compr;
    afim:=afim-f*dist/compr;
  end
  else
  begin
 nin:=nin-conc*(sqr((compr-dist)/compr)*(3-2*(compr-dist)/compr));
    nfim:=nfim-conc*(sqr(dist/compr)*(3-2*dist/compr));
    min:=min-conc*(dist*sqr((compr-dist)/compr));
    mfim:=mfim+conc*(compr-dist)*sqr(dist/compr);
  end;

end;

procedure ENGASTUD(OPTION.COMPR.UD.TAM.UDDI.COSENO.SENO:double;
              Var E.F.M.H.NIN.NFIM.MIN.MFIM.AIN.AFIM:double);
begin
  if option=0 then
  begin
    E:=ud*tam*sqr(coseno);
    F:=ud*tam*coseno*seno;
    M:=uddi/compr;
    H:=sqr(tam/(2*compr));
    NIN:=NIN-E*(1-3*sqr(M)-H+2*M*(sqr(M)+H));
    NFIM:=NFIM-E*(3*sqr(M)+H-2*M*(sqr(M)+H));
    MIN:=MIN-E*(UDDI*sqr((COMPR-UDDI)/COMPR)-H*(3*(COMPR-UDDI)-
       COMPR)/3);
    MFIM:=MFIM+E*((COMPR-UDDI)*sqr(M)-H*(3*UDDI-COMPR)/3);
    AIN:=AIN-F*(COMPR-UDDI)/COMPR;
    AFIM:=AFIM-F*UDDI/COMPR;
  end
  else
  begin
    E:=ud*tam;
    M:=uddi/compr;
    H:=sqr(tam/(2*compr));
    NIN:=NIN-E*(1-3*sqr(M)-H+2*M*(sqr(M)+H));
    NFIM:=NFIM-E*(3*sqr(M)+H-2*M*(sqr(M)+H));
    MIN:=MIN-E*(UDDI*sqr((COMPR-UDDI)/COMPR)-H*(3*(COMPR-UDDI)-
       COMPR)/3);
    MFIM:=MFIM+E*((COMPR-UDDI)*sqr(M)-H*(3*UDDI-COMPR)/3);
  end;
```

```
end;

PROCEDURE ENGASTAXUV(AXVAR.AXVARNI.COMPR:double;
            Var NIN.NFIM.MIN.MFIM.AIN.AFIM:double);
Begin
                (*NAO HA OPCAO PARA O SISTEMA GLOBAL*)
  IF AXVARNI=0 THEN
  BEGIN
    AIN:=AIN+AXVAR*(COMPR/3)/COMPR;
    AFIM:=AFIM+AXVAR*(COMPR*2/3)/COMPR;
    END
    ELSE
    BEGIN
      AIN:=AIN+AXVAR*(COMPR*2/3)/COMPR;
      AFIM:=AFIM+AXVAR*(COMPR/3)/COMPR;
    END;
  END;

PROCEDURE ENGASTAXC(OPTION:longint; AXIC.AXICDI.COMPR.
            COSENO.SENO:double;
          Var E.F.NIN.NFIM.MIN.MFIM.AIN.AFIM:double);
BEGIN
  IF OPTION=0 THEN
  BEGIN       (* OSISTEMA ESTA NAS COORDENADAS GLOBAIS*)
        (*AXIC=CARGA CONCENTRADA NA COORDENADA GLOBAL 1*)
    F:=AXIC*COSENO;  (*COMPONENTE NA COORDENADA LOCAL 1*)
    E:=AXIC*SENO;    (*COMPONENTE NA COORDENADA LOCAL 2*)
    NIN:=NIN-
     E*(SQR((COMPR-AXICDI)/COMPR)*(3-2*((COMPR-AXICDI)/COMPR)));
    NFIM:=NFIM-E*(SQR(AXICDI/COMPR))*(3-2*(AXICDI/COMPR));
    MIN:=MIN-E*(AXICDI*SQR((COMPR-AXICDI)/COMPR));
    MFIM:=MFIM+E*((COMPR-AXICDI)*SQR(AXICDI/COMPR));
    AIN:=AIN-F*(COMPR-AXICDI)/COMPR;
    AFIM:=AFIM-F*AXICDI/COMPR;
  END
  ELSE
  BEGIN
    AIN:=AIN-AXIC*(COMPR-AXICDI)/COMPR;
    AFIM:=AFIM-AXIC*AXICDI/COMPR;
  END;
END;

Procedure ENGASTMB(MOMB.MOMBDI.COMPR:double;
            Var NIN.NFIM.MIN.MFIM.AIN.AFIM:double);
Begin
  nin:=nin-momb*6*mombdi*(compr-mombdi)/(compr*sqr(compr));
  nfim:=nfim+momb*6*mombdi*(compr-mombdi)/(compr*sqr(compr));
  min:=min-momb*(compr-mombdi)*(2-3*(compr-mombdi)/compr)/compr;
  mfim:=mfim-momb*mombdi*(2-3*mombdi/compr)/compr;
end;



Procedure Cabenos;
Var
R:real;
y:integer;
```

```
begin
  writeln(arqs.
'FOI ANALISADA A ESTRUTURA CUJOS DADOS FIGURAM NO ARQUIVO '.
        NOMEARQE.' ');
  writeln(arqs);
  writeln(arqs.'A matriz U armazenou'.limite.' posicoes');
  writeln(arqs);
  Y:=sqr(3*N);
  R:=(limite/Y)*100;
  writeln(arqs.'A matriz de rigidez do caso analisado tem '.Y.' posicoes');
  writeln(arqs.'  e portanto o metodo skyline armazenou '.R:5:2.' % da referida matriz de rigidez.');
  writeln(arqs.'OS RESULTADOS ENCONTRADOS FORAM:');
  writeln(arqs);
  writeln(arqs);
  writeln(arqs.'                          DESLOCAMENTOS');
  writeln(arqs);
  write(arqs.'  NO    COORDENADA 1          COORDENADA 2');
  writeln(arqs.'          COORDENADA 3');
end;

Procedure Mostradesloc(C.D.E:double; g:longint; Var X:longint);
Var
A,y:integer;
begin
  write(arqs,'  '.g:3.'    '.C:9:6.'              '.D:9:6);
  writeln(arqs.'          '.E:9:6);
end;



Procedure Roda  (B:longint; Topbar.barra:barptr; mbeta:matriz);
Var
T.U:double;
G:longint;
CA:Array[1..6] of double;

begin
   FOR G:=1 TO B DO
   BEGIN
     Tripbar(G.barra);
     T:=BARRA^.COSX;
     U:=BARRA^.COSY;
     matrizbeta(T.U.mbeta);
     CA[1]:=BARRA^.AXIN;
     CA[2]:=BARRA^.NORIN;
     CA[3]:=BARRA^.MOMIN;
     CA[4]:=BARRA^.AXFIM;
     CA[5]:=BARRA^.NORFIM;
     CA[6]:=BARRA^.MOFIM;
     barra^.axin:=mbeta[1.1]*CA[1]+mbeta[1.2]*CA[2]+
              mbeta[1.3]*CA[3]+mbeta[1.4]*CA[4];
     barra^.axin:=barra^.axin+mbeta[1.5]*CA[5]+mbeta[1.6]*CA[6];
     barra^.norin:=mbeta[2.2]*CA[2]+mbeta[2.1]*CA[1]+
              mbeta[2.3]*CA[3]+
              mbeta[2.4]*CA[4];
     barra^.norin:=barra^.norin+mbeta[2.5]*CA[5]+mbeta[2.6]*CA[6];
     barra^.momin:=mbeta[3.3]*CA[3]+mbeta[3.1]*CA[1]+
              mbeta[3.2]*CA[2]+
```

```
                mbeta[3,4]*CA[4];
    barra^.momin:=barra^.momin+mbeta[3,5]*CA[5]+
                mbeta[3,6]*CA[6];
    barra^.axfim:=mbeta[4,4]*CA[4]+mbeta[4,1]*CA[1]+
                mbeta[4,2]*CA[2]+
                mbeta[4,3]*CA[3];
    barra^.axfim:=barra^.axfim+mbeta[4,5]*CA[5]+
                mbeta[4,6]*CA[6];
    barra^.norfim:=mbeta[5,1]*CA[1]+mbeta[5,2]*CA[2]+
                 mbeta[5,3]*CA[3]+
                mbeta[5,4]*CA[4];
    barra^.norfim:=barra^.norfim+mbeta[5,5]*CA[5]+
                 mbeta[5,6]*CA[6];
    barra^.mofim:=mbeta[6,1]*CA[1]+mbeta[6,2]*CA[2]+
                 mbeta[6,3]*CA[3]+
                mbeta[6,4]*CA[4];
    barra^.mofim:=barra^.mofim+mbeta[6,6]*CA[6]+
                mbeta[6,5]*CA[5];
  end;
end;



Procedure Tripae (busk:longint; Var AE:AEptr);
begin
  while busk<AE^.pos do
  AE:=AE^.next;
  while busk>AE^.pos do
  Ae:=AE^.ant;
end;


Procedure Carga  (N,B:longint; Topbar,barra:barptr; mbeta:matriz;
              no,topno:noptr;     lastae,AE:aeptr;  Var topAE:aeptr);
Var
X,Y,I,busk,W:longint;
T,U:double;
G:longint;
AA:matriz;
CA:Array[1..6] of double;

begin
  barra:=topbar;
  FOR G:=1 TO B DO
  BEGIN
    T:=BARRA^.COSX;
    U:=BARRA^.COSY;
    matrizbeta(T,U,mbeta);
    For x:=1 to 6 do
    For y:=1 to 6 do
    AA[x,y]:=mbeta[y,x];
    CA[1]:=BARRA^.AXIN;
    CA[2]:=BARRA^.NORIN;
    CA[3]:=BARRA^.MOMIN;
    CA[4]:=BARRA^.AXFIM;
    CA[5]:=BARRA^.NORFIM;
    CA[6]:=BARRA^.MOFIM;
    barra^.axin:=AA[1,1]*CA[1]+AA[1,2]*CA[2]+AA[1,3]*CA[3]+
```

```
            AA[1,4]*CA[4];
  barra^.axin:=barra^.axin+AA[1,5]*CA[5]+AA[1,6]*CA[6];
  barra^.norin:=AA[2,2]*CA[2]+AA[2,1]*CA[1]+AA[2,3]*CA[3]+
            AA[2,4]*CA[4];
  barra^.norin:=barra^.norin+AA[2,5]*CA[5]+AA[2,6]*CA[6];
  barra^.momin:=AA[3,3]*CA[3]+AA[3,1]*CA[1]+AA[3,2]*CA[2]+
            AA[3,4]*CA[4];
  barra^.momin:=barra^.momin+AA[3,5]*CA[5]+AA[3,6]*CA[6];
  barra^.axfim:=AA[4,4]*CA[4]+AA[4,1]*CA[1]+AA[4,2]*CA[2]+
            AA[4,3]*CA[3];
  barra^.axfim:=barra^.axfim+AA[4,5]*CA[5]+AA[4,6]*CA[6];
  barra^.norfim:=AA[5,1]*CA[1]+AA[5,2]*CA[2]+AA[5,3]*CA[3]+
            AA[5,4]*CA[4];
  barra^.norfim:=barra^.norfim+AA[5,5]*CA[5]+AA[5,6]*CA[6];
  barra^.mofim:=AA[6,1]*CA[1]+AA[6,2]*CA[2]+AA[6,3]*CA[3]+
            AA[6,4]*CA[4];
  barra^.mofim:=barra^.mofim+AA[6,6]*CA[6]+AA[6,5]*CA[5];
  If g<>b then
  barra:=barra^.next;
END;
new(AE);
AE^.eVAL:=0;
AE^.next:=nil;
AE^.pos:=1;
FOR Y:=2 TO 3*N DO
begin
  lastAE:=AE;
  new(AE);
  lastAE^.ant:=AE;
  AE^.Eval:=0;
  AE^.pos:=Y;
  AE^.next:=lastAE;
end;
topAE:=AE;
I:=0;
FOR G:=1 TO B DO
BEGIN
  Tripbar(G,barra);
  busk:=BARRA^.noinicio*3-2;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.AXIN;
  busk:=BARRA^.noinicio*3-1;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.NORIN;
  busk:=BARRA^.noinicio*3;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.MOMIN;
  busk:=BARRA^.nofim*3-2;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.AXFIM;
  busk:=BARRA^.nofim*3-1;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.NORFIM;
  busk:=BARRA^.nofim*3;
  tripae(busk,ae);
  AE^.eval:=AE^.eval-BARRA^.MOFIM;
END;
FOR X:=1 TO N DO
```

```
  BEGIN
    i:=x;
    tripno(I.no);
    busk:=3*X-2;
    tripAE(busk.AE);
    AE^.Eval:=AE^.Eval+NO^.AXNO;
    busk:=3*X-1;
    tripAE(busk.AE);
    AE^.Eval:=AE^.Eval+NO^.norno;
    busk:=3*X;
    tripAE(busk.AE);
    AE^.Eval:=AE^.Eval+NO^.mono;
  END;
  FOR W:=1  TO N DO
  BEGIN
    I:=w;
     tripno(I.no);
    CASE NO^.RESTRI OF
    3:BEGIN
        busk:=3*W-2;
        tripAE(busk.AE);
        AE^.Eval:=0;
        busk:=3*W-1;
        tripAE(busk.AE);
        AE^.Eval:=0;
        busk:=3*W;
        tripAE(busk.AE);
        AE^.Eval:=0;
      END;
    2:BEGIN
        busk:=3*W-2;
        tripAE(busk.AE);
        AE^.Eval:=0;
        busk:=3*W-1;
        tripAE(busk.AE);
        AE^.Eval:=0;
      END;
    1:BEGIN
        busk:=3*W-1;
        tripAE(busk.AE);
        AE^.Eval:=0;
      END;
    4:BEGIN
        busk:=3*W-2;
        tripAE(busk.AE);
        AE^.Eval:=0;
      END;
    0:begin
        no^.restri:=0;
      end;
    END;
  END;
end;


BEGIN
   WRITELN('PROGRAMA   PORTICO   PLANO');
   WRITELN('versao skyline');
```

```
WRITE ('ARQUIVO DE DADOS= ');
writeln;
READLN(NOMEARQE);
ASSIGN(ARQE,NOMEARQE);RESET(ARQE);
WRITELN(NOMEARQE);
writeln('ARQUIVO DE SAIDA= ');
readln(nomearqs);
GetTime(hour,minute,second,sec100);
tempo:=60*minute+second+sec100/100;
ASSIGN(ARQS,NOMEARQS);REwrite(ARQS);
WRITELN(NOMEARQS);
READLN(ARQE,B);
READLN(ARQE,N);
new(no);
lastno:=no;
X:=0;
FOR I:=1 TO N DO
BEGIN
  READLN(ARQE,no^.nu,no^.ABSNO,no^.ORDNO,no^.RESTRI);
  If no^.restri<>0 then
  begin
    x:=x+1;
    gno[X]:=no;
  end;
  if I=1 then
  begin
    no^.next:=nil;
    new(no);
  end
  else
  begin
    if I<N then
    begin
      no^.next:=lastno;
      lastno^.ant:=no;
      lastno:=no;
      new(no);
    end
    else
    begin
      no^.next:=lastno;
      lastno^.ant:=no;
      no^.ant:=nil;
    end;
  end;
END;
maxres:=X;
topno:=no;
new(barra);
lastbarra:=barra;
primbar:=barra;
FOR G:=1 TO B DO
BEGIN
  READLN(ARQE,barra^.num,BARRA^.NOINICIO,BARRA^.NOFIM,
       BARRA^.MINERCIA,
       BARRA^.MODELAST,BARRA^.AREA);
  I:=barra^.noinicio;
  tripno(I,no);
```

```
    R:=-no^.ordno;
    D:=-no^.absno;
    I:=barra^.nofim;
    tripno(I.no);
    R:=R+no^.ordno;
    D:=D+no^.absno;
    C:=sqrt(sqr(R)+sqr(D));
    barra^.comp:=C;
    barra^.cosx:=D/C;
    barra^.cosy:=R/C;
    if g=1 then
    begin
      barra^.next:=nil;
      new(barra);
    end
    else
    begin
      if g<b then
      begin
        barra^.next:=lastbarra;
        lastbarra^.ant:=barra;
        lastbarra:=barra;
        new(barra);
      end
      else
      begin
        barra^.next:=lastbarra;
        lastbarra^.ant:=barra;
      end;
    end;
END;
topbar:=barra;
limax:=0;
linha:=0;
Z:=0;
max:=3*n;
For x:=1 to 3*n do
VETH[x]:=x;
while MAX>=1 do
begin
  linha:=linha+1;
  numbar:=0;
  barra:=primbar;
  For g:=1 to B do
  begin
    If (barra^.noinicio)*3-2=linha then
    begin
      numbar:=numbar+1;
      gba[numbar]:=barra;
    end;
    If (barra^.nofim)*3-2=linha then
    begin
      numbar:=numbar+1;
      gba[numbar]:=barra;
    end;
    barra:=barra^.ant;
  end;
  limax:=limax+max;
```

```
limin:=limax-max;
For contador:=1 to 3 do
begin
  for bbb:=3*N-max+1 to 3*N do
  auxivet[bbb]:=0;
  g:=1;
  while G<=numbar do
  begin
    barra:=gba[G];
    assemblagem(N,B,limax,limin,barra,no,topno,max,maxres,linha,numbar,gno,auxivet,G);
    g:=g+1;
  end;
  Y:=linha;
  for X:=linha to 3*n do
  begin
    Z:=Z+1;
    if auxivet[X]=0 then
    begin
      If VETH[X]=X-Y+1 then
      begin
        VETH[X]:=VETH[X]-1;
      end
      else
      begin
        new(kglobal);
        kglobal^.kval:=0;
        kglobal^.ordem:=Z;
        kglobal^.next:=lastk;
        lastk^.ant:=kglobal;
        lastk:=kglobal;
        posicao:=posicao+1;
      end;
    end
    else
    begin
      If Y=1 then
      begin
        If x=1 then
        begin
          new(kglobal);
          auxk:=kglobal;
          kglobal^.kval:=auxivet[x];
          kglobal^.ordem:=1;
          lastk:=kglobal;
          kglobal^.next:=nil;
          posicao:=1;
        end
        else
        begin
          new(kglobal);
          kglobal^.next:=lastk;
          lastk^.ant:=kglobal;
          lastk:=kglobal;
          kglobal^.kval:=auxivet[X];
          kglobal^.ordem:=Z;
          posicao:=posicao+1;
        end;
      end
```

```
          else
          begin
            new(kglobal);
            kglobal^.next:=lastk;
            lastk^.ant:=kglobal;
            lastk:=kglobal;
            kglobal^.kval:=auxivet[X];
            kglobal^.ordem:=Z;
            posicao:=posicao+1;
          end;
        end;
      end;
      If contador<3 then
      begin
        linha:=linha+1;
        max:=max-1;
        limax:=limax+max;
        limin:=limax-max;
      end;
    end;
    max:=max-1;
end;
topk:=kglobal;
kglobal^.ant:=nil;
limite:=posicao;
writeln('Foi construida a matriz de rigidez e armazenadas '.limite.' posicoes');
barra:=topbar;
FOR G:=1 TO B DO
BEGIN
  BARRA^.NORIN:=0;
  BARRA^.NORFIM:=0;
  BARRA^.AXIN:=0;
  BARRA^.AXFIM:=0;
  BARRA^.MOMIN:=0;
  BARRA^.MOFIM:=0;
  BARRA^.UV:=0;
  BARRA^.UVNI:=0;
  BARRA^.AXUV:=0;
  BARRA^.AXUVNI:=0;
  FOR X:=1 TO 4 DO
  BEGIN
    barra^.cc[x]:=0;
    barra^.di[x]:=0;
    barra^.axc[x]:=0;
    barra^.axcdi[x]:=0;
    barra^.ud[x]:=0;
    barra^.tam[x]:=0;
    barra^.uddi[x]:=0;
    barra^.mab[x]:=0;
    barra^.mabdi[x]:=0;
  end;
  If g<>b then
  barra:=barra^.next;
END;
READLN(ARQE.OPTION);  (*ESTA VARIAVEL. DA AO USUARIO A OPCAO DE
                    COLOCAR SEU CARREGAMENTO SEGUNDO O
                     SISTEMA
                    GLOBAL(0) DE COORDENADAS. OU LOCAL(1)*)
```

```
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.G);
    tripbar(G.barra);
    for x:=1 to 4 do
    BEGIN
      READ(ARQE.BARRA^.CC[X].BARRA^.DI[X]);
        (*CC:=CARGA CONCENTRADA NA COORDENADA GLOBAL 2*)
      E:=BARRA^.CC[X]*BARRA^.COSX; (*agora na coord.
                          local 2*)
      F:=BARRA^.CC[X]*BARRA^.COSY; (*coord. local 1*)
      CONC:=BARRA^.CC[X];
      DIST:=BARRA^.DI[X];
      remete;
      ENGASTCC(OPTION.E.F.COMPR.CONC.DIST.NIN.NFIM.MIN.
        MFIM.AIN.AFIM);
      recebe;
    END;
    READLN(ARQE);
  end;
end;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.G);
    tripBAR(G.BARRA);
    for x:=1 to 4 do
    begin
      READ(ARQE.BARRA^.AXC[X].BARRA^.AXCDI[X]);
      AXIC:=BARRA^.AXC[X];
      AXICDI:=BARRA^.AXCDI[X];
      COSENO:=BARRA^.COSX;
      SENO:=BARRA^.COSY;
      REMETE;
      ENGASTAXC(OPTION.AXIC.AXICDI.COMPR.COSENO.SENO.E.F.
            NIN.NFIM.MIN.MFIM.
        AIN.AFIM);
      RECEBE;
    end;
    READLN(ARQE);
  end;
END;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.G);
    TripBAR(G.BARRA);
    read(arqe.total);(*total=1 significa
                que a carga se extende por toda a
      barra*)
    If total=1 then
```

```
    begin
      read(arqe,barra^.ud[1]);
      ud:=barra^.ud[1];
      tam:=barra^.comp;
      uddi:=tam/2;
      coseno:=barra^.cosx;
      seno:=barra^.cosy;
      remete;
      ENGASTud(OPTION,COMPR,ud,tam,uddi,COSENO,SENO,e,f,m,
            h,NIN,NFIM,MIN,
        MFIM,AIN,AFIM);
      recebe;
    end
    else
    begin
      for x:=1 to 4 do
      begin
        READ(ARQE,BARRA^.UD[X],BARRA^.TAM[X],
            BARRA^.UDDI[X]);
            (*UD:=CARGA UN. DISTRIB. COORDENADA GLOBAL
                  2*)
        ud:=BARRA^.UD[X];
        tam:=BARRA^.TAM[X];
        uddi:=BARRA^.UDDI[X];
        coseno:=BARRA^.COSX;
        seno:=BARRA^.COSY;
        remete;
        ENGASTud(OPTION,COMPR,ud,tam,uddi,COSENO,SENO,e,f,
            m,h,NIN,NFIM,MIN,
            MFIM,AIN,AFIM);
        recebe;
      end;
    end;
    READLN(ARQE);
  end;
END;
READLN(ARQE,W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE,G);
    TripBAR(G,BARRA);
    read(arqe,BARRA^.UV,BARRA^.UVNI);
    E:=BARRA^.UV*BARRA^.COSX;
    F:=BARRA^.UV*BARRA^.COSY;
    COSENO:=BARRA^.COSX;
    SENO:=BARRA^.COSY;
    UVAR:=BARRA^.UV;
    remete;
    UNI:=BARRA^.UVNI;
    ENGASTUV(OPTION,E,F,COMPR,COSENO,SENO,UVAR,UNI,NIN,NFIM,
        MIN,
        MFIM,AIN,AFIM);
    recebe;
    readln(arqe);
  end;
end;
```

```
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.G);
    TripBAR(G.BARRA);
    for x:=1 to 4 do
    begin
      READ(ARQE.BARRA^.MAB[X].BARRA^.MABDI[X]);
      momb:=BARRA^.MAB[X];
      mombdi:=BARRA^.MABDI[X];
      REMETE;
      ENGASTMB(MOMB.MOMBDI.COMPR.NIN.NFIM.MIN.MFIM.AIN.
               AFIM);
      RECEBE;
    end;
    READLN(ARQE);
  end;
end;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  READ(ARQE.G);
  TripBAR(G.BARRA);
  READLN(ARQE.BARRA^.AXUV.BARRA^.AXUVNI);
     (*NAO HA OPCAO PARA O SISTEMA GLOBAL*)
  AXVAR:=BARRA^.AXUV;
  AXVARNI:=BARRA^.AXUVNI;
  REMETE;
  ENGASTAXUV(AXVAR.AXVARNI.COMPR.
             NIN.NFIM.MIN.MFIM.AIN.AFIM);
  RECEBE;
end;
FOR I:=1 TO N DO
BEGIN
   tripno(I.no);
  NO^.AXNO:=0;
  NO^.NORNO:=0;
  NO^.MONO:=0;
END;
READLN(ARQE.W);
if w<>0 then
begin
  for y:=1 to w do
  begin
    READ(ARQE.I);
     tripno(I.no);
    READLN(arqe.NO^.AXNO.NO^.NORNO.NO^.MONO);
  end;
end;
Carga (N.B.topbar.barra.mbeta.no.topno.lastAE.AE.topAE);
writeln('Foi construido o vetor de carga');
 ELE[1]:=1;
FOR X:=2 TO 3*N DO
ELE[X]:=0;
X:=1;
```

```
G:=1;
sinal:=0;
ae:=topae;
FOR Y:=1 TO 3*N-1 DO
BEGIN
  kglobal:=auxk;
  Z:=((X-1)*3*N-X*(X-1) DIV 2)+X;   (*W*)
  while Z<kglobal^.ordem do
   kglobal:=kglobal^.next;
  while Z>kglobal^.ordem do
  kglobal:=kglobal^.ant;
  val1:=kglobal^.kval;
  Z:=((Y-1)*3*N-Y*(Y-1) div 2)+3*N;
  While Kglobal^.ordem<Z do
  kglobal:=kglobal^.ant;
  If kglobal^.ordem>auxk^.ordem then
  auxk:=kglobal;
  If kglobal^.ordem=Z then
   begin
    auxivet[3*N]:=kglobal^.kval;
    lastk:=kglobal;
    kglobal:=kglobal^.next;
    zero:=lastk^.ordem-kglobal^.ordem-1;
    sinal:=1;
  end
  else
  begin
    auxivet[3*N]:=0;
    proxk:=kglobal^.next;
    zero:=Z-proxk^.ordem-1;
  end;
  For bbb:=3*N-1 downto Y do
  begin
    If Zero=0 then
    begin
      If sinal=1 then
      begin
        auxivet[bbb]:=kglobal^.kval;
        proxk:=kglobal^.next;
        zero:=kglobal^.ordem-proxk^.ordem-1;
        sinal:=0;
      end
      else
      begin
        proxk:=kglobal^.next;
        auxivet[bbb]:=proxk^.kval;
        If bbb<>Y then
        begin
          kglobal:=kglobal^.next;
          proxk:=kglobal^.next;
          zero:=kglobal^.ordem-proxk^.ordem-1;
        end;
      end;
    end
    else
    begin
      auxivet[bbb]:=0;
      zero:=zero-1;
```

```
    end;
  end;

WHILE G<3*N DO
  BEGIN
  ELE[G+1]:=-auxivet[g+1]/val1;(* ELE[G+1]:=-KGLOBAL[Z]/KGLOBAL[W];*)
  G:=G+1;
  END;
     busk:=X;
  tripAE(busk,AE);
  D:=AE^.Eval;
  FOR A:=X+1  TO 3*N DO
  begin
    busk:=A;
    tripAE(busk,AE);
    AE^.Eval:=AE^.Eval+D*ELE[A];
  end;
  sinal:=0;
  bip:=0;
  FOR A:=Y+1  TO 3*N DO
  begin
    If ELE[A]<>0 then
    begin
      contador:=0;
      FOR K:=A TO 3*N DO
      BEGIN
        If auxivet[3*N-contador]<>0 then
        begin
          Z:=((A-1)*3*N-A*(A-1) DIV 2)-K+3*N+A;
          If Z<kglobal^.ordem then
          begin
            bip:=1;
            while Z<kglobal^.ordem do
            kglobal:=kglobal^.next;
          end
          else
          begin
            while Z>kglobal^.ordem do
            kglobal:=kglobal^.ant;
          end;
          If kglobal^.ordem=Z then
          begin
            val1:=kglobal^.kval;
            bip:=0;
          end
          else
          begin
            val1:=0;
          end;
          valor:=val1+ELE[A]*auxivet[3*N-contador];
          kglobal^.kval:=valor;
        end;
        contador:=contador+1;
      end;
    end;
  end;
  X:=X+1;
  G:=X;
```

```
end;

writeln('Foi construida a matriz U que armazenou '.limite.
        'posicoes');
AE:=topAE;
kglobal:=topk;
R:=0;
ELE[3*N]:=TOPAE^.EVAL/TOPK^.KVAL;
Z:=((3*N-2)*3*N-(3*N-1)*(3*N-2) DIV 2)+3*N;
AE:=AE^.next;
kglobal:=kglobal^.next;
If Z=kglobal^.ordem then
begin
  valor:=kglobal^.kval;
end
else
begin
  valor:=0;
end;
AE^.EVAL:=AE^.EVAL-valor*ELE[3*N];
Z:=((3*N-2)*3*N-(3*N-1)*(3*N-2) DIV 2)+3*N-1;
while Z<kglobal^.ordem do
kglobal:=kglobal^.next;
If Z=kglobal^.ordem then
begin
  valor:=kglobal^.kval;
end
else
begin
  valor:=0;
end;
ELE[3*n-1]:=AE^.EVAL/valor;
bip:=0;
Z:=((3*N-3)*3*N-(3*N-3)*(3*N-2) div 2)+3*N;
while Z<kglobal^.ordem do
kglobal:=kglobal^.next;
while Z>kglobal^.ordem do
kglobal:=kglobal^.ant;
For K:=3*N-1 downto 2 do
begin
  AE:=AE^.next;
  If z=kglobal^.ordem then
  begin
    valor:=kglobal^.kval;
    AE^.eval:=Ae^.eval-valor*ELE[3*N];
    proxk:=kglobal^.next;
    zero:=kglobal^.ordem-proxk^.ordem-1;
  end
  else
  begin
    zero:=Z-kglobal^.ordem-1;
    bip:=1;
  end;
  Z:=Z-1;
  Y:=1;
  A:=3*N-K+1;
  while Y<A do
  begin
```

```
  If zero=0 then
  begin
    If bip=0 then
    begin
      kglobal:=kglobal^.next;
      valor:=kglobal^.kval;
      R:=R+valor*ELE[3*N-Y];
      proxk:=kglobal^.next;
      zero:=kglobal^.ordem-proxk^.ordem-1;
      Z:=Z-1;
    end
    else
    begin
      proxk:=kglobal^.next;
      If Z=proxk^.ORDEM THEN
      begin
        kglobal:=kglobal^.next;
        valor:=kglobal^.kval;
        R:=R+valor*ELE[3*N-Y];
        proxk:=kglobal^.next;
        zero:=kglobal^.ordem-proxk^.ordem-1;
        Z:=Z-1;
        bip:=0;
      end
      else
      begin
        valor:=kglobal^.kval;
        proxk:=kglobal^.next;
        zero:=kglobal^.ordem-proxk^.ordem-1;
        bip:=0;
        R:=R+valor*ELE[3*N-Y];
        Z:=Z-1;
      end;
    end;
  end
  else
  begin
    zero:=zero-1;
    Z:=Z-1;
  end;
  Y:=Y+1;
end;
If Z=kglobal^.ordem then
begin
  valor:=kglobal^.kval;
end
else
begin
  kglobal:=kglobal^.next;
  valor:=kglobal^.kval;
end;
ELE[K-1]:=(AE^.eval-R)/valor;
If K>2 then
begin
  R:=0;
  proxk:=kglobal^.next;
  zero:=kglobal^.ordem-proxk^.ordem-1;
  Z:=Z-1;
```

```
      kglobal:=kglobal^.next;
    end;
  end;
  FOR G:=1 TO N DO
  BEGIN
    I:=G;
    tripno(I,no);
    NO^.DAX:=ELE[3*G-2];
    NO^.DNOR:=ELE[3*G-1];
    NO^.ROT:=ELE[3*G];
  end;
  cabenos;
  x:=0;
  for g:=1 to N do
  begin
    I:=G;
    tripno(I,no);
    C:=no^.dax;
    D:=no^.dnor;
    E:=no^.rot;
    mostradesloc(C,D,E,g,X);
  end;
  Roda (B,topbar,barra,mbeta);
  FOR G:=1 TO B DO
  BEGIN
    TripBAR(G,BARRA);
    T:=BARRA^.COSX;
    U:=BARRA^.COSY;
    MATRIZBETA(T,U,MBETA);
    FOR X:=1 TO 6 DO
    BEGIN
      I:=BARRA^.NOINICIO;
       tripno(I,no);
      R:=MBETA[X,1]*NO^.DAX+MBETA[X,2]*NO^.DNOR+
         MBETA[X,3]*NO^.ROT;
      I:=BARRA^.NOFIM;
       tripno(I,no);
      CA[X]:=R+MBETA[X,4]*NO^.DAX+MBETA[X,5]*NO^.DNOR+
          MBETA[X,6]*NO^.ROT;
    END;
    P:=BARRA^.MODELAST;
    J:=BARRA^.MINERCIA;
    S:=BARRA^.AREA;
    C:=BARRA^.COMP;
    MATRIZRIG(C,J,S,P,MRIGB);
    FOR X:=1 TO 6 DO
    BARRA^.AXIN:=BARRA^.AXIN+MRIGB[1,X]*CA[X];
    FOR X:=1 TO 6 DO
    BARRA^.NORIN:=BARRA^.NORIN+MRIGB[2,X]*CA[X];
    FOR X:=1 TO 6 DO
    BARRA^.MOMIN:=BARRA^.MOMIN+MRIGB[3,X]*CA[X];
    FOR X:=1 TO 6 DO
    BARRA^.AXFIM:=BARRA^.AXFIM+MRIGB[4,X]*CA[X];
    FOR X:=1 TO 6 DO
    BARRA^.NORFIM:=BARRA^.NORFIM+MRIGB[5,X]*CA[X];
    FOR X:=1 TO 6 DO
    BARRA^.MOFIM:=BARRA^.MOFIM+MRIGB[6,X]*CA[X];
  END;
```

```
WRITELN(ARQS);
writeln(arqs.'                    ESFORCOS NAS BARRAS');
WRITELN(ARQS);
writeln(arqs.'  BARRA  NO INICIAL                NO FINAL');
writeln(arqs.'        AXIAL   NORMAL   MOMENTO  AXIAL   NORMAL   MOMENTO');
FOR G:=1 TO B DO
BEGIN
  tripbar(G.barra);
  WRITE(ARQS.'  ',g:3.'  ',BARRA^.AXIN:8:2,'  ',BARRA^.NORIN:8:2);
  WRITE(ARQS.'  ',BARRA^.MOMIN:8:2,'  ',BARRA^.AXFIM:8:2);
  WRITELN(ARQS.'  ',BARRA^.NORFIM:8:2,'  ',BARRA^.MOFIM:8:2);
END;
For I:=1 to N do
BEGIN
  tripno(I.no);
  NO^.REAX:=0;
  NO^.RENOR:=0;
  NO^.REMON:=0;
END;
FOR G:=1 TO B DO
BEGIN
  TripBAR(G.BARRA);
  T:=BARRA^.COSX;
  U:=BARRA^.COSY;
  MATRIZBETA(T.U.MBETA);
  FOR X:=1 TO 6 DO
  FOR K:=1 TO 6 DO
  AA[X.K]:=MBETA[K.X];
  FOR X:=1 TO 6 DO
  BEGIN
    R:=AA[X.1]*BARRA^.AXIN+AA[X.2]*BARRA^.NORIN+
       AA[X.3]*BARRA^.MOMIN;
    CA[X]:=R+AA[X.4]*BARRA^.AXFIM+AA[X.5]*BARRA^.NORFIM+
       AA[X.6]*BARRA^.MOFIM;
  END;
  I:=BARRA^.NOINICIO;
   tripno(I.no);
  NO^.REAX:=NO^.REAX+CA[1];
  NO^.RENOR:=NO^.RENOR+CA[2];
  NO^.REMON:=NO^.REMON+CA[3];
  I:=BARRA^.NOFIM;
   tripno(I.no);
  NO^.REAX:=NO^.REAX+CA[4];
  NO^.RENOR:=NO^.RENOR+CA[5];
  NO^.REMON:=NO^.REMON+CA[6];
END;
x:=0;
writeln(arqs);
writeln(arqs.'                    REACOES NOS APOIOS');
writeln(arqs);
write(arqs.'  NO   COORDENADA 1       COORDENADA 2');
writeln(arqs.'       COORDENADA 3');
FOR G:=1 TO N DO
BEGIN
  I:=G;
  tripno(I.no);
  IF NO^.RESTRI>0 THEN
  BEGIN
```

```
        write(arqs.' '.G:3.'      '.NO^.REAX:9:2);
        WRITE(arqs.'          '.no^.renor:9:2.'          ');
        WRITELN(arqs.no^.remon:9:2);
      END;
    END;
    t1:=tempo;
    GetTime(hour,minute,second,sec100);
    writeln('hora= '.hour.' '.minute.' '.second.' '.sec100);
    tempo:=60*minute+second+sec100/100;
    t2:=tempo-t1;
    writeln(arqs.'O tempo total de processamento utilizado pelo metodo');
    writeln(arqs.'skyline foi de '.t2:7:2,' segundos.');
    LIMPA;
    close(arqs);
    for g:=1 to 6 do
    writeln;
    writeln('Foi analisada a estrutura constante do arquivo. '.nomearqe.'.');
    writeln('Os resultados encontrados estao no arquivo. '.nomearqs);
    writeln;
    writeln;
    writeln('Para sair digite qualquer tecla');
    repeat until keypressed;
end.
```

ANEXO 5
Relatórios de Arquivos do SAP90

1- RESULTADOS OBTIDOS PARA A ESTRUTURA DO PRIMEIRO CASO

UNICAMP**********
PAGE 1

PROGRAM:SAP90/FILE:portico.SOL

PORTICO

J O I N T D I S P L A C E M E N T S

LOAD CONDITION 1 - DISPLACEMENTS "U" AND ROTATIONS "R"

JOINT	U(X)	U(Y)	R(Z)
1	.000000	.000000	.000000
2	.5050E-03	-.8187E-05	-.4437E-03
3	.001457	-.000016	-.000501
4	.001457	-.000843	.000151
5	.001457	.000000	.000565

UNICAMP**********
PAGE 2

PROGRAM:SAP90/FILE:portico.SOL

PORTICO

R E A C T I O N S A N D A P P L I E D F O R C E S

LOAD CONDITION 1 - FORCES "F" AND MOMENTS "M"

JOINT	F(X)	F(Y)	M(Z)
1	-1.6000	4.0937	3.3518
2	.0000	.0000	-1.0000
3	.0000	-2.0000	.0000
4	.0000	-2.0000	.0000
5	.0000	2.4063	.0000

TOTAL -.1600E+01 .2500E+01 .2352E+01

2- RESULTADOS OBTIDOS PARA A ESTRUTURA DO SEGUNDO CASO

UNICAMP ********** PAGE
1

PROGRAM:SAP90/FILE:trelica.SOL

TRELICA HIPERESTATICA

J O I N T D I S P L A C E M E N T S

LOAD CONDITION 1 - DISPLACEMENTS "U" AND ROTATIONS "R"

JOINT	U(X)	U(Y)	R(Z)
1	.000000	.000000	.000000
2	.043759	-.000065	-.007032
3	.043759	-.001119	-.007032
4	.045370	-.002199	-.001037
5	.043793	-.004250	-.001327
6	.045670	-.004344	-.001894
7	.043919	-.005993	-.001468
8	.045798	-.006073	-.001233
9	.044191	-.007804	-.000873
10	.045722	-.007790	-.000807
11	.044450	-.008861	-.000352
12	.044935	-.008862	-.000427
13	.045604	-.008933	-.000349
14	.044736	-.008691	.000025
15	.045084	-.008845	.000095
16	.044648	-.008609	.000315
17	.045209	-.008616	.000105
18	.044553	-.008596	.000796
19	.044542	-.006541	.000468
20	.043473	-.006517	.000440
21	.044435	-.006971	.000276
22	.043225	-.006977	.000413
23	.042895	-.007018	-.000031
24	.044346	-.006114	.000577
25	.042725	-.006212	.000528
26	.044288	-.005145	.000664
27	.043820	-.005148	.000694
28	.042595	-.005165	.000929
29	.044230	-.003139	.001069
30	.042759	-.003194	.001076
31	.044116	-.001430	.000918
32	.042880	-.001510	.001011
33	.043992	-.000230	.001758
34	.043025	-.000337	.000717
35	.043234	.000826	.002326
36	.043686	.001138	-.007816
37	.043686	-.000034	-.007816
38	.000000	.000000	.000000

PROGRAM:SAP90/FILE:trelica.SOL

TRELICA HIPERESTATICA

R E A C T I O N S A N D A P P L I E D F O R C E S

LOAD CONDITION 1 - FORCES "F" AND MOMENTS "M"

JOINT	F(X)	F(Y)	M(Z)
1	-62.2689	28.7670	220.5544
2	9.6700	.0000	-5.3000
3	-.1615E-06	.4305E-05	-.9456E-07
4	1.0000	-3.9000	.0000
5	-.1475E-11	.0000E+00	.0000E+00
6	1.4000	-3.4000	.0000
7	-.3257E-11	.0000E+00	.0000E+00
8	2.6000	-9.6000	.0000
9	-.1029E-11	.0000E+00	.0000E+00
10	2.3000	-8.4000	.0000
11	-.1422E-11	.0000E+00	.0000E+00
12	.0000	.0000	.0000
13	2.3000	-8.4000	.0000
14	.2122E-11	.0000E+00	.0000E+00
15	9.9000	-3.5000	.0000
16	.1874E-11	.0000E+00	.0000E+00
17	.0000	.0000	.0000
18	.0000	1.1000	.0000
19	.0000	.0000	.0000
20	2.1000	6.5000	.0000
21	.1726E-11	.0000E+00	.0000E+00
22	.0000	.0000	.0000
23	.0000	-5.3000	.0000
24	.0000	.0000	.0000
25	-2.0000	-2.3000	.0000
26	-.2108E-11	.0000E+00	.0000E+00
27	-.1452E-11	.0000E+00	.0000E+00
28	-.6000	-2.3000	.0000
29	.1060E-11	.0000E+00	.0000E+00
30	-.6000	-2.3000	.0000
31	.0000	.0000	.0000
32	-.7000	-2.6000	.0000
33	-.3407E-11	.0000E+00	.0000E+00
34	-.4000	-1.4000	.0000
35	-3.0000	-1.1000	.0000
36	.1830E-06	-.8394E-05	-.1125E-05

37	-2.0000	.0000	1.1200
38	-44.7411	18.1330	284.5301
TOTAL	-.8504E+02	-.7958E-12	.5009E+03

3- RESULTADOS OBTIDOS PARA A ESTRUTURA DO TERCEIRO CASO

UNICAMP**********

PROGRAM:SAP90/FILE:AVE.SOL

ESTRUTURA CIRCULAR

J O I N T D I S P L A C E M E N T S

LOAD CONDITION 1 - DISPLACEMENTS "U" AND ROTATIONS "R"

JOINT	U(X)	U(Y)	R(Z)
1	.000000	.000000	-.010247
2	-.002516	-.014381	-.007840
3	.006171	-.007103	-.007290
4	.010952	-.002578	-.007479
5	-.002183	-.019336	-.006744
6	.003021	-.019255	-.005842
7	.004684	-.016755	-.006042
8	.010900	-.013680	-.008180
9	.016856	-.013868	-.005699
10	.014282	-.015043	.006674
11	.017257	-.014486	.003278
12	.016828	-.014857	-.003036
13	.014999	-.014462	-.008220
14	.010351	-.016542	-.015193
15	.007411	-.016560	-.009139
16	.004748	-.019088	-.004069
17	.003887	-.019194	-.003313
18	.003362	-.021098	-.003442
19	.002575	-.021640	-.003822
20	.001239	-.022368	-.007820
21	-.002093	-.024454	-.009920
22	.008312	-.013788	.008689
23	.011983	-.013660	.008195
24	.014692	-.013501	.006658
25	.009832	-.024272	-.015185
26	.006655	-.026744	-.023134

JOINT	U(X)	U(Y)	R(Z)
27	.003956	-.024046	-.017375
28	.002283	-.023610	-.003137
29	.001667	-.024250	-.003760
30	-.000255	-.025572	-.011104
31	.001739	-.027129	-.006797
32	-.000022	-.028957	-.010082
33	-.003884	-.033674	-.013309
34	-.004054	-.043609	-.010025
35	-.001262	-.047683	-.001026
36	.001657	-.045162	.007875
37	.001903	-.036907	.010331
38	.001261	-.029405	.012680
39	.003122	-.025452	.006872
40	.005001	-.020331	.009303
41	.002593	.000000	.013824
42	-.008806	-.005063	.009003
43	-.004742	-.010632	.008478
44	-.000157	-.035758	-.026343
45	-.002643	-.039881	-.012295
46	-.001569	-.044990	-.007790
47	-.002053	-.047338	-.005188
48	-.000945	-.048495	-.001062
49	-.000075	-.048066	.003105
50	-.000159	-.046409	.005539
51	.000798	-.042115	.009993
52	-.001044	-.038674	.023684
53	-.001011	-.033033	.007253
54	-.000635	-.030548	.009303
55	-.000856	-.028569	.003296
56	-.001570	-.026063	.004088
57	-.002827	-.023529	.006470
58	-.008929	-.020104	.009987
59	-.001986	-.031701	.004003
60	-.001465	-.029468	.001575
61	-.001404	-.028042	.002580
62	-.001743	-.027601	.002498
63	-.002066	-.025932	.002875
64	-.002866	-.025737	.003690
65	-.005295	-.023271	.008797
66	-.008375	-.023250	.015589
67	-.002082	-.030260	.016338
68	-.007615	-.030487	.014015
69	-.004611	-.032818	.021798
70	-.007616	-.035733	.029547

JOINT	U(X)	U(Y)	R(Z)
71	-.000578	-.036024	.027451
72	-.003010	-.051766	.033012
73	-.005547	-.067798	.027562
74	.001673	-.068006	.027445
75	-.000861	-.081454	.018508
76	-.000436	-.085408	.000663
77	.000874	-.085631	.004303
78	.000589	-.086010	.001883
79	.001250	-.085699	-.001526
80	.000795	-.086003	-.001526
81	.001457	-.085645	-.004910
82	.002934	-.084681	-.003726
83	.000719	-.084904	-.007366
84	.002906	-.079138	-.021500
85	.007976	-.064107	-.030228
86	-.000012	-.064315	-.030109
87	.005056	-.046719	-.035244
88	.002400	-.030130	-.028989
89	.009885	-.029840	-.031074
90	-.012557	-.020961	.010222
91	-.015632	-.020143	.005287
92	-.015589	-.021486	.004661
93	-.016327	-.020965	-.002843
94	-.014181	-.021476	-.005633
95	-.014568	-.020158	-.005647
96	-.012118	-.020280	-.007993
97	-.008374	-.020414	-.008889
98	-.009352	-.018197	-.009069
99	-.005824	-.017681	-.008784
100	-.002614	-.017346	-.007665
101	-.003816	-.015558	-.007849
102	-.001042	-.014726	-.006080
103	.000496	-.014244	-.003315
104	-.000179	-.013458	-.002905
105	.000657	-.013233	-.001124
106	.000281	-.013076	-.000843
107	.000596	-.013187	.002099
108	.000581	-.013093	-.000878
109	.000994	-.013275	-.001184
110	.000465	-.012997	-.000543
111	.000335	-.012852	-.000756
112	.000627	-.012869	-.001787
113	.000500	-.012553	-.001066
114	.000347	-.012182	-.001290

JOINT	U(X)	U(Y)	R(Z)
115	.000715	-.012167	-.002180
116	.000563	-.011772	-.001304
117	.000773	-.011381	-.001112
118	.000438	-.011430	-.001118
119	.000648	-.011156	-.000930
120	.000830	-.010936	-.000065
121	.000529	-.010950	-.000947
122	.000711	-.010699	-.001188
123	.000920	-.010391	-.000397
124	.000546	-.010384	-.001503
125	.000695	-.009965	-.001857
126	.000602	-.009267	-.001682
127	.001167	-.011318	-.001100
128	.000966	-.008945	-.004091
129	.001100	-.007891	.001047
130	.001077	-.012778	-.001589
131	.001497	-.012401	-.001140
132	.001168	-.012299	-.001200
133	.001467	-.012023	-.000856
134	.001723	-.011786	-.000806
135	.001448	-.011760	-.000744
136	.001651	-.011523	-.000769
137	.001907	-.011185	-.001122
138	.001570	-.011242	-.001118
139	.001739	-.010711	-.001474
140	.001928	-.010121	-.001457
141	.001540	-.010227	-.001519
142	.001605	-.009594	-.001420
143	.001659	-.009109	-.001149
144	.001358	-.009245	-.001111
145	.001349	-.008787	-.000672
146	.001227	-.008681	-.001177
147	.001349	-.008505	-.001035
148	.000890	-.008020	-.001012
149	.000809	-.008310	.001498
150	.001653	-.008578	.004656
151	.004043	-.009284	.006807
152	.003019	-.010821	.006625
153	.005821	-.011121	.008143
154	-.000358	-.045410	-.007683
155	-.000638	-.049004	-.001069
156	-.000883	-.046829	.005316
157	.009260	-.011616	.008869
158	-.001582	-.029722	.001538

159	-.003135	-.025633	.014251
160	.005082	-.019093	-.015232

UNICAMP**********

PROGRAM:SAP90/FILE:AVE.SOL

ESTRUTURA CIRCULAR

R E A C T I O N S A N D A P P L I E D F O R C E S

LOAD CONDITION 1 - FORCES "F" AND MOMENTS "M"

JOINT	F(X)	F(Y)	M(Z)
1	.0000	1888.3540	.0000
2	-.1644E-11	-.2643E-11	.0000E+00
3	.0000	.0000	.0000
4	-.2444E-11	.0000E+00	.0000E+00
5	-.1592E-11	.0000E+00	.0000E+00
6	.0000E+00	.2285E-10	.0000E+00
7	.0000E+00	-.2785E-11	.0000E+00
8	.0000E+00	.1478E-11	.0000E+00
9	-.2359E-11	-.4434E-11	.0000E+00
10	.1805E-11	.0000E+00	.0000E+00
11	.0000E+00	.5627E-11	.0000E+00
12	.0000	.0000	.0000
13	.1862E-11	-.2245E-11	.0000E+00
14	.4775E-11	.1364E-11	.0000E+00
15	-.2075E-11	.2615E-11	.0000E+00
16	.1108E-11	.8242E-11	.0000E+00
17	-.3922E-11	-.7844E-11	.0000E+00
18	.0000E+00	-.1012E-10	.0000E+00
19	.1577E-11	.3212E-11	.0000E+00
20	.0000E+00	.1725E-10	.0000E+00
21	.2046E-11	.0000E+00	.0000E+00
22	.0000E+00	-.3162E-11	.0000E+00
23	.0000	.0000	.0000
24	.1027E-11	-.8640E-11	.0000E+00
25	-.1080E-11	.1215E-10	.0000E+00
26	.9493E-11	.1569E-10	.0000E+00
27	.0000E+00	.1046E-10	.0000E+00
28	-.1549E-11	.0000E+00	.0000E+00
29	.3865E-11	-.5002E-11	.0000E+00
30	.0000E+00	-.3254E-11	.0000E+00
31	.0000E+00	-.5059E-11	.0000E+00
32	.3013E-11	.2146E-11	.0000E+00
33	-.2615E-11	.2899E-11	.0000E+00
34	.3865E-11	-.9180E-11	.0000E+00

JOINT	F(X)	F(Y)	M(Z)
35	.0000E+00	-.2615E-11	.0000E+00
36	.1705E-11	-.1216E-10	.0000E+00
37	.2444E-11	.1399E-10	.0000E+00
38	.0000E+00	.3212E-11	.0000E+00
39	.0000E+00	.2558E-11	.0000E+00
40	.0000E+00	-.3794E-11	.0000E+00
41	.0000	1888.3540	.0000
42	.0000	.0000	.0000
43	.0000	.0000	.0000
44	-.4334E-11	.7674E-11	.0000E+00
45	-.4661E-11	-.1517E-11	.0000E+00
46	.0000E+00	.2621E-10	.0000E+00
47	.0000E+00	-.1876E-11	.0000E+00
48	.0000E+00	-.2444E-11	.0000E+00
49	.0000E+00	.4036E-11	.0000E+00
50	.0000E+00	.1637E-10	.0000E+00
51	.1478E-11	.0000E+00	.0000E+00
52	-.1023E-11	-.6366E-11	.0000E+00
53	-.1592E-11	-.1634E-11	.0000E+00
54	.2061E-11	-.1086E-10	.0000E+00
55	-.2501E-11	.3212E-11	.0000E+00
56	.0000	.0000	.0000
57	.0000E+00	-.2160E-11	.0000E+00
58	-.3887E-11	.4860E-11	.0000E+00
59	-.2046E-11	.2103E-11	.0000E+00
60	.2103E-11	-.6338E-11	.0000E+00
61	-.2728E-11	-.3666E-11	.0000E+00
62	.0000E+00	.5457E-11	.0000E+00
63	.1510E-11	.9052E-11	.0000E+00
64	-.1478E-11	.0000E+00	.0000E+00
65	-.1265E-11	-.1137E-11	.0000E+00
66	-.3382E-11	.8981E-11	.0000E+00
67	.2370E-11	-.5144E-11	.0000E+00
68	-.8029E-11	-.1326E-10	.0000E+00
69	.7482E-11	.2156E-10	.0000E+00
70	-.3794E-11	.7997E-11	.0000E+00
71	-.8640E-11	-.1566E-11	.0000E+00
72	.0000E+00	-.2487E-11	.0000E+00
73	.0000E+00	.1521E-11	.0000E+00
74	.0000E+00	.2480E-11	.0000E+00
75	.0000E+00	-.4292E-11	.0000E+00
76	.2251E-10	.5998E-10	.0000E+00
77	-.4862E-10	.6761E-10	.0000E+00
78	.4661E-11	-.9973E-10	.0000E+00
79	-.1342E-10	.2469E-10	.0000E+00

JOINT	F(X)	F(Y)	M(Z)
80	.1971E-10	.5586E-11	.0000E+00
81	-.2061E-11	-.7887E-11	.0000E+00
82	.3979E-11	.3342E-10	.0000E+00
83	-.1023E-11	-.2801E-10	.0000E+00
84	-.4320E-11	-.1478E-11	.0000E+00
85	.1364E-11	.9493E-11	.0000E+00
86	.0000E+00	-.1874E-10	.0000E+00
87	.0000E+00	-.1052E-11	.0000E+00
88	.4547E-11	-.8645E-11	.0000E+00
89	.5663E-11	-.1080E-10	.0000E+00
90	.1307E-11	-.7134E-11	.0000E+00
91	-.3681E-11	-.8697E-11	.0000E+00
92	-.2281E-11	-.1052E-10	.0000E+00
93	.0000E+00	.1080E-11	.0000E+00
94	-.1862E-11	.1791E-11	.0000E+00
95	.1599E-11	.0000E+00	.0000E+00
96	.0000E+00	.1904E-11	.0000E+00
97	.1734E-11	.5443E-11	.0000E+00
98	-.2494E-11	-.2771E-11	.0000E+00
99	-.2530E-11	.3823E-11	.0000E+00
100	.1378E-11	.0000E+00	.0000E+00
101	-.1393E-11	.1023E-11	.0000E+00
102	-.1265E-11	.0000E+00	.0000E+00
103	.2807E-11	.0000E+00	.0000E+00
104	.0000	.0000	.0000
105	-.1819E-11	.0000E+00	.0000E+00
106	.0000E+00	-.1229E-11	.0000E+00
107	-.2220E-11	.4711E-11	.0000E+00
108	-.1712E-11	.3173E-11	.0000E+00
109	-.2775E-11	.3099E-11	.0000E+00
110	.2707E-11	.0000E+00	.0000E+00
111	.0000E+00	-.1695E-11	.0000E+00
112	.0000E+00	.2402E-11	.0000E+00
113	-.2415E-11	-.2146E-11	.0000E+00
114	.1840E-11	.0000E+00	.0000E+00
115	.0000E+00	-.1034E-11	.0000E+00
116	.0000E+00	.2885E-11	.0000E+00
117	-.1229E-11	.0000E+00	.0000E+00
118	-.1066E-11	.5814E-11	.0000E+00
119	.0000	.0000	.0000
120	.0000E+00	.2412E-11	.0000E+00
121	.1617E-11	-.2021E-11	.0000E+00
122	-.1054E-11	.1619E-11	.0000E+00
123	.0000E+00	.1978E-11	.0000E+00
124	.0000	.0000	.0000

JOINT	F(X)	F(Y)	M(Z)
125	-.1087E-11	-.1318E-11	.0000E+00
126	.0000	.0000	.0000
127	.0000E+00	.2537E-11	.0000E+00
128	.2942E-11	-.2103E-11	.0000E+00
129	.0000	.0000	.0000
130	.2217E-11	.0000E+00	.0000E+00
131	-.1579E-11	.1002E-11	.0000E+00
132	.0000E+00	-.2412E-11	.0000E+00
133	.2083E-11	-.1537E-11	.0000E+00
134	.1158E-11	.1394E-11	.0000E+00
135	-.1005E-11	.0000E+00	.0000E+00
136	.1727E-11	-.1158E-11	.0000E+00
137	-.2373E-11	.0000E+00	.0000E+00
138	.0000	.0000	.0000
139	.1080E-11	.0000E+00	.0000E+00
140	.0000	.0000	.0000
141	.0000	.0000	.0000
142	.2547E-11	-.1545E-11	.0000E+00
143	.0000E+00	-.2177E-11	.0000E+00
144	.0000	.0000	.0000
145	.0000	.0000	.0000
146	.0000E+00	-.3340E-11	.0000E+00
147	.0000E+00	.5194E-11	.0000E+00
148	.0000E+00	.2750E-11	.0000E+00
149	.0000E+00	-.4064E-11	.0000E+00
150	-.1466E-11	.0000E+00	.0000E+00
151	-.1819E-11	.0000E+00	.0000E+00
152	.0000E+00	-.2736E-11	.0000E+00
153	.3211E-11	.1037E-11	.0000E+00
154	-.1222E-11	-.1933E-11	.0000E+00
155	.1350E-11	-.1074E-10	.0000E+00
156	.0000E+00	.4150E-11	.0000E+00
157	.0000	.0000	.0000
158	.3794E-11	-.2103E-11	.0000E+00
159	.0000	.0000	.0000
160	.0000E+00	-.1186E-11	.0000E+00
TOTAL	-.6745E-11	.3777E+04	-.3214E-12

ANEXO 6
Relatório de Saída do Sparse para o Terceiro Caso

FOI ANALISADA A ESTRUTURA CUJOS DADOS FIGURAM NO ARQUIVO
plan.pas

A matriz U armazenou20879 posicoes

A matriz de rigidez do caso analisado tem 230400 posicoes
e portanto o metodo sparse armazenou 9.06 % da referida matriz de rigidez.
OS RESULTADOS ENCONTRADOS FORAM:

DESLOCAMENTOS

NO	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	0.000000	0.000000	-0.010616
2	-0.002657	-0.014843	-0.008225
3	0.006459	-0.007314	-0.007669
4	0.011596	-0.002580	-0.007871
5	-0.002412	-0.020052	-0.007146
6	0.003188	-0.019943	-0.006263
7	0.005094	-0.017312	-0.006419
8	0.011556	-0.014160	-0.008561
9	0.017781	-0.014334	-0.006017
10	0.015334	-0.015545	0.006572
11	0.018268	-0.015000	0.003099
12	0.017752	-0.015405	-0.003313
13	0.015820	-0.014979	-0.008576
14	0.011029	-0.017131	-0.015580
15	0.007959	-0.017130	-0.009474
16	0.005198	-0.019751	-0.004452
17	0.004206	-0.019871	-0.003671
18	0.003597	-0.021885	-0.003834
19	0.002673	-0.022472	-0.004202
20	0.001190	-0.023250	-0.008281
21	-0.002417	-0.025369	-0.010221
22	0.009389	-0.014317	0.008703
23	0.013066	-0.014190	0.008154
24	0.015740	-0.014037	0.006555
25	0.010478	-0.025207	-0.015353
26	0.007276	-0.027699	-0.023234
27	0.004556	-0.024987	-0.017563
28	0.002315	-0.024580	-0.003576
29	0.001585	-0.025312	-0.003845
30	-0.000293	-0.026651	-0.012195
31	0.001668	-0.028196	-0.006481
32	0.000018	-0.029962	-0.009223
33	-0.003486	-0.034363	-0.012431

34	-0.003591	-0.043860	-0.009568
35	-0.000807	-0.047645	-0.000678
36	0.002062	-0.044887	0.008147
37	0.002233	-0.036511	0.010379
38	0.001584	-0.029051	0.012615
39	0.003475	-0.025211	0.006743
40	0.005380	-0.020179	0.009179
41	0.003018	0.000000	0.013703
42	-0.008173	-0.005066	0.008879
43	-0.004226	-0.010563	0.008356
44	-0.000002	-0.036356	-0.025475
45	-0.002293	-0.040272	-0.011698
46	-0.001254	-0.045205	-0.007300
47	-0.001657	-0.047413	-0.004810
48	-0.000597	-0.048454	-0.000702
49	0.000306	-0.047894	0.003433
50	0.000168	-0.046143	0.005812
51	0.001131	-0.041750	0.010173
52	-0.000737	-0.038281	0.023763
53	-0.000693	-0.032634	0.007182
54	-0.000306	-0.030190	0.009223
55	-0.000466	-0.028287	0.003167
56	-0.001103	-0.025843	0.003964
57	-0.002279	-0.023350	0.006354
58	-0.008300	-0.019950	0.009862
59	-0.001637	-0.031328	0.003891
60	-0.001050	-0.029161	0.001437
61	-0.000964	-0.027776	0.002447
62	-0.001254	-0.027352	0.002367
63	-0.001550	-0.025714	0.002757
64	-0.002303	-0.025524	0.003553
65	-0.004703	-0.023086	0.008691
66	-0.007740	-0.023058	0.015457
67	-0.001458	-0.029962	0.016358
68	-0.007006	-0.030193	0.014055
69	-0.003995	-0.032529	0.021887
70	-0.007017	-0.035458	0.029665
71	0.000055	-0.035750	0.027585
72	-0.002393	-0.051571	0.033270
73	-0.004976	-0.067776	0.027896
74	0.002337	-0.067984	0.027777
75	-0.000242	-0.081607	0.018893
76	0.000127	-0.085763	0.001043
77	0.001549	-0.085987	0.004682
78	0.001208	-0.086411	0.002251
79	0.001815	-0.086144	-0.001158

80	0.001469	-0.086448	-0.001158
81	0.002076	-0.086134	-0.004553
82	0.003501	-0.085212	-0.003365
83	0.001391	-0.085436	-0.007004
84	0.003526	-0.079862	-0.021175
85	0.008566	-0.064958	-0.029994
86	0.000641	-0.065166	-0.029877
87	0.005677	-0.047691	-0.035126
88	0.003013	-0.031092	-0.029034
89	0.010516	-0.030802	-0.031138
90	-0.011884	-0.020794	0.010085
91	-0.014902	-0.020000	0.005143
92	-0.014859	-0.021301	0.004514
93	-0.015537	-0.020797	-0.002993
94	-0.013340	-0.021322	-0.005774
95	-0.013735	-0.019964	-0.005788
96	-0.011231	-0.020092	-0.008118
97	-0.007445	-0.020228	-0.008991
98	-0.008434	-0.017984	-0.009171
99	-0.004868	-0.017463	-0.008854
100	-0.001644	-0.017126	-0.007698
101	-0.002849	-0.015331	-0.007884
102	-0.000064	-0.014495	-0.006073
103	0.001451	-0.014019	-0.003251
104	0.000791	-0.013245	-0.002834
105	0.001601	-0.013036	-0.001016
106	0.001259	-0.012958	-0.000716
107	0.001538	-0.013045	0.002209
108	0.001509	-0.012941	-0.000762
109	0.001893	-0.013104	-0.001062
110	0.001424	-0.012917	-0.000411
111	0.001317	-0.012806	-0.000621
112	0.001568	-0.012823	-0.001651
113	0.001462	-0.012544	-0.000929
114	0.001329	-0.012212	-0.001154
115	0.001658	-0.012196	-0.002044
116	0.001526	-0.011839	-0.001170
117	0.001716	-0.011485	-0.000978
118	0.001422	-0.011534	-0.000983
119	0.001612	-0.011297	-0.000795
120	0.001774	-0.011114	0.000071
121	0.001514	-0.011129	-0.000810
122	0.001675	-0.010915	-0.001049
123	0.001865	-0.010646	-0.000255
124	0.001532	-0.010639	-0.001363
125	0.001667	-0.010255	-0.001715

126	0.001589	-0.009600	-0.001538
127	0.002064	-0.011422	-0.000971
128	0.001911	-0.009306	-0.003974
129	0.002089	-0.008335	0.001177
130	0.001971	-0.012648	-0.001461
131	0.002348	-0.012314	-0.001005
132	0.002055	-0.012223	-0.001064
133	0.002322	-0.011986	-0.000721
134	0.002544	-0.011795	-0.000674
135	0.002307	-0.011776	-0.000613
136	0.002486	-0.011581	-0.000640
137	0.002718	-0.011289	-0.000991
138	0.002420	-0.011346	-0.000988
139	0.002574	-0.010863	-0.001342
140	0.002748	-0.010324	-0.001322
141	0.002399	-0.010422	-0.001383
142	0.002459	-0.009843	-0.001280
143	0.002508	-0.009412	-0.001009
144	0.002245	-0.009531	-0.000970
145	0.002242	-0.009132	-0.000532
146	0.002158	-0.009052	-0.001054
147	0.002247	-0.008899	-0.000896
148	0.001837	-0.008441	-0.000875
149	0.001772	-0.008780	0.001626
150	0.002667	-0.009065	0.004772
151	0.005091	-0.009783	0.006901
152	0.004052	-0.011341	0.006716
153	0.006891	-0.011646	0.008203
154	-0.000158	-0.045625	-0.007141
155	-0.000394	-0.048966	-0.000707
156	-0.000607	-0.046563	0.005577
157	0.010338	-0.012144	0.008884
158	-0.001215	-0.029387	0.001402
159	-0.002512	-0.025333	0.014227
160	0.005695	-0.019973	-0.015489

ESFORCOS NAS BARRAS

BARRA	NO INICIAL			NO FINAL		
	AXIAL	NORMAL	MOMENTO	AXIAL	NORMAL	MOMENTO
1	-692.97	6.14	3.23	692.97	-6.14	4.79
2	1116.48	5.13	1.11	-1116.48	-5.13	3.76
3	778.64	-4.29	-4.33	-778.64	4.29	-2.95
4	-674.43	2.33	1.38	674.43	-2.33	1.74
5	-1246.62	-8.39	-3.65	1246.62	8.39	-2.37
6	188.96	7.95	2.69	-188.96	-7.95	4.54

7	454.54	-4.05	-3.36	-454.54	4.05	-2.24
8	-168.47	-1.85	-1.85	168.47	1.85	-1.99
9	-660.40	-6.43	-3.42	660.40	6.43	-3.58
10	1102.29	-4.13	-2.07	-1102.29	4.13	-2.73
11	26.69	3.44	2.38	-26.69	-3.44	1.91
12	387.56	6.78	4.15	-387.56	-6.78	5.29
13	340.97	-4.15	-2.33	-340.97	4.15	-1.52
14	-368.76	43.47	11.68	368.76	-43.47	9.70
15	-893.53	-19.19	-6.98	893.53	19.19	-9.95
16	-489.11	-2.79	-1.03	489.11	2.79	-1.05
17	1.77	-9.80	-2.28	-1.77	9.80	-1.58
18	631.25	7.75	0.73	-631.25	-7.75	1.59
19	-464.08	3.54	0.44	464.08	-3.54	0.99
20	-443.14	-2.75	-0.88	443.14	2.75	-1.17
21	-392.51	2.35	1.03	392.51	-2.35	0.72
22	-259.16	-3.14	-0.21	259.16	3.14	-1.03
23	714.66	15.68	2.00	-714.66	-15.68	2.70
24	-572.68	3.95	0.47	572.68	-3.95	1.18
25	269.34	1.16	0.25	-269.34	-1.16	0.61
26	-173.50	6.18	1.30	173.50	-6.18	1.30
27	936.25	-19.23	-1.64	-936.25	19.23	-4.13
28	-91.90	10.17	2.17	91.90	-10.17	1.92
29	497.02	-3.27	-2.12	-497.02	3.27	-0.32
30	240.78	2.26	-0.86	-240.78	-2.26	1.71
31	24.56	-7.79	-1.65	-24.56	7.79	-0.69
32	243.90	-8.63	-1.27	-243.90	8.63	-2.00
33	189.49	-0.55	-0.36	-189.49	0.55	-0.05
34	202.73	3.34	0.46	-202.73	-3.34	0.88
35	-88.07	9.74	1.46	88.07	-9.74	1.46
36	-130.37	-8.03	-1.25	130.37	8.03	-2.13
37	-148.29	8.39	2.45	148.29	-8.39	0.52
38	225.57	-7.91	-2.03	-225.57	7.91	-1.10
39	-184.43	-2.12	0.42	184.43	2.12	-1.17
40	-395.19	0.75	1.21	395.19	-0.75	-0.70
41	-434.90	-2.03	-1.46	434.90	2.03	0.07
42	242.08	4.63	1.86	-242.08	-4.63	-0.02
43	-260.65	-20.02	-4.66	260.65	20.02	-2.94
44	-304.30	-1.05	-1.23	304.30	1.05	0.50
45	-95.35	14.83	2.05	95.35	-14.83	3.49
46	-1.25	-6.34	-1.32	1.25	6.34	-1.10
47	-421.10	-0.84	-0.34	421.10	0.84	-0.24
48	-572.08	8.25	1.51	572.08	-8.25	1.46
49	858.50	-5.18	-1.34	-858.50	5.18	-3.01
50	-204.18	-5.48	-0.97	204.18	5.48	-1.07
51	590.08	-4.98	-1.93	-590.08	4.98	-3.32
52	-474.21	0.31	0.09	474.21	-0.31	0.13

53	-463.47	7.28	1.26	463.47	-7.28	1.44
54	-201.96	-8.52	-1.18	201.96	8.52	-2.26
55	123.83	-1.02	-0.99	-123.83	1.02	0.68
56	-617.18	-5.71	-1.74	617.18	5.71	-0.57
57	-348.04	-6.29	-2.06	348.04	6.29	-2.62
58	215.07	-4.18	-1.49	-215.07	4.18	-1.75
59	-191.85	142.05	11.48	191.85	-142.05	10.09
60	-525.41	0.20	0.23	525.41	-0.20	-0.08
61	-99.74	-3.50	-0.62	99.74	3.50	-0.77
62	151.05	-1.17	-0.18	-151.05	1.17	-0.31
63	-37.68	-0.57	-0.07	37.68	0.57	-0.35
64	-39.25	6.34	0.92	39.25	-6.34	0.98
65	-124.90	1.41	0.49	124.90	-1.41	0.04
66	185.57	-2.69	-0.61	-185.57	2.69	-0.40
67	-90.93	-0.84	-0.47	90.93	0.84	-0.11
68	147.63	90.87	12.93	-147.63	100.70	-17.63
69	43.68	33.49	5.40	-43.68	-33.49	1.00
70	187.87	23.34	3.89	-187.87	-23.34	3.11
71	-42.02	27.36	4.92	42.02	20.12	-4.06
72	587.45	-7.89	-2.34	-587.45	7.89	0.83
73	-19.05	41.13	5.55	19.05	-41.13	2.32
74	565.61	-67.18	-4.22	-565.61	67.18	-8.63
75	977.52	-4.56	-2.18	-977.52	4.56	-2.39
76	1666.05	27.41	5.83	-1666.05	-27.41	0.68
77	-669.96	-0.49	0.06	669.96	0.49	-0.39
78	-209.65	-4.88	-0.87	209.65	4.88	-0.95
79	-191.63	-9.26	-0.76	191.63	9.26	-2.97
80	-626.73	10.20	2.28	626.73	-10.20	1.52
81	166.08	-7.43	-2.13	-166.08	7.43	-0.10
82	-446.21	-8.02	-1.88	446.21	8.02	-1.00
83	-281.95	-1.34	-0.48	281.95	1.34	-0.52
84	-408.38	-5.29	-0.48	408.38	5.29	-1.33
85	-886.62	-5.91	-0.55	886.62	5.91	-3.50
86	-466.16	6.57	1.82	466.16	-6.57	1.04
87	-362.90	-7.58	0.51	362.90	7.58	-3.66
88	-913.69	1.43	0.33	913.69	-1.43	0.74
89	433.70	-40.45	-3.75	-433.70	40.45	-8.39
90	-397.19	16.14	2.97	397.19	-16.14	3.18
91	-1080.47	-0.50	-0.82	1080.47	0.50	0.45
92	-700.61	-7.13	-1.22	700.61	7.13	-1.77
93	737.13	0.55	-0.32	-737.13	-0.55	0.49
94	-522.69	11.44	1.63	522.69	-11.44	2.91
95	-540.26	-12.28	-1.97	540.26	12.28	-3.05
96	690.93	0.07	0.01	-690.93	-0.07	0.01
97	-546.48	12.18	1.95	546.48	-12.18	3.03
98	-1074.91	0.49	-0.45	1074.91	-0.49	0.82

99	-892.66	-1.65	-0.78	892.66	1.65	-0.46
100	-515.99	-11.25	-1.60	515.99	11.25	-2.87
101	-710.88	6.81	1.17	710.88	-6.81	1.69
102	-510.95	2.18	0.69	510.95	-2.18	1.00
103	-254.80	0.82	0.39	254.80	-0.82	0.22
104	-475.35	-7.44	-1.23	475.35	7.44	-2.01
105	425.86	40.32	3.67	-425.86	-40.32	8.43
106	-382.52	-16.22	-3.06	382.52	16.22	-3.12
107	-174.44	-99.85	-6.39	174.44	99.85	-8.77
108	234.80	5.57	2.81	-234.80	-5.57	1.51
109	-849.27	5.05	2.58	849.27	-5.05	1.87
110	-1171.68	2.52	0.72	1171.68	-2.52	1.08
111	-355.78	-13.54	-2.94	355.78	13.54	-3.72
112	337.03	0.40	0.35	-337.03	-0.40	0.03
113	-705.68	-1.45	-1.13	705.68	1.45	-0.76
114	763.21	1.04	1.04	-763.21	-1.04	0.74
115	1139.48	-1.21	-0.28	-1139.48	1.21	-0.88
116	-629.05	1.57	0.88	629.05	-1.57	0.83
117	-26.00	-1.04	-0.69	26.00	1.04	-0.61
118	417.43	-1.54	-0.93	-417.43	1.54	-1.21
119	-631.95	-0.40	-0.23	631.95	0.40	-0.30
120	1136.63	0.71	0.34	-1136.63	-0.71	0.48
121	-28.16	0.44	0.44	28.16	-0.44	0.47
122	358.34	0.65	0.56	-358.34	-0.65	0.34
123	106.63	-2.24	-0.71	-106.63	2.24	-1.32
124	-663.37	-30.88	-7.57	663.37	30.88	-3.44
125	-511.76	-4.78	-3.06	511.76	4.78	-0.23
126	60.91	259.37	26.18	-60.91	265.88	-28.32
127	-349.22	13.31	2.03	349.22	-13.31	3.22
128	540.34	-27.71	-2.80	-540.34	27.71	-2.72
129	-363.29	0.84	-0.23	363.29	-0.84	0.80
130	-503.61	-1.24	-0.59	503.61	1.24	0.13
131	-479.27	3.08	0.01	479.27	-3.08	1.16
132	491.74	0.51	0.07	-491.74	-0.51	0.07
133	-368.88	-0.94	-0.83	368.88	0.94	0.19
134	-472.92	-3.32	-1.21	472.92	3.32	-0.05
135	-510.38	1.27	-0.11	510.38	-1.27	0.58
136	540.30	29.11	2.96	-540.30	-29.11	2.84
137	-528.80	4.68	0.21	528.80	-4.68	3.00
138	-334.97	-13.82	-3.31	334.97	13.82	-2.13
139	-674.24	31.43	3.57	674.24	-31.43	7.64
140	50.85	265.68	28.26	-50.85	259.57	-26.26
141	-358.35	7.30	3.65	358.35	-7.30	-0.61
142	-927.74	5.81	3.54	927.74	-5.81	0.45
143	-417.55	6.10	1.56	417.55	-6.10	0.53
144	-441.57	7.61	1.06	441.57	-7.61	1.67

145	-372.59	4.71	2.19	372.59	-4.71	1.32
146	175.68	5.41	1.76	-175.68	-5.41	-0.14
147	-179.15	9.40	2.93	179.15	-9.40	0.86
148	-510.92	0.95	0.30	510.92	-0.95	0.41
149	174.92	-8.06	-0.90	-174.92	8.06	-1.52
150	-219.78	4.96	1.10	219.78	-4.96	0.91
151	-656.56	0.82	0.40	656.56	-0.82	-0.07
152	-565.72	2.53	0.77	565.72	-2.53	1.11
153	609.07	-9.20	-1.10	-609.07	9.20	-1.67
154	-62.08	9.07	1.95	62.08	-9.07	1.63
155	-517.81	-1.10	-0.66	517.81	1.10	-0.16
156	-572.84	-3.53	-0.27	572.84	3.53	-1.20
157	691.51	-12.60	-1.39	-691.51	12.60	-2.39
158	-289.73	6.00	0.87	289.73	-6.00	1.50
159	239.56	-1.51	-0.23	-239.56	1.51	-0.90
160	-175.22	-10.15	-1.89	175.22	10.15	-2.20
161	987.35	20.41	2.07	-987.35	-20.41	4.06
162	-170.97	-0.60	-0.14	170.97	0.60	-0.08
163	-663.38	-9.39	-1.39	663.38	9.39	-2.10
164	-682.85	1.04	0.55	682.85	-1.04	0.17
165	-474.07	0.00	-0.07	474.07	-0.00	0.07
166	-191.20	7.07	1.32	191.20	-7.07	1.31
167	-222.48	6.17	1.16	222.48	-6.17	1.14
168	-575.13	-7.40	-1.39	575.13	7.40	-1.27
169	-417.21	0.48	0.07	417.21	-0.48	0.26
170	-66.78	5.00	0.84	66.78	-5.00	1.06
171	-292.20	1.20	-0.52	292.20	-1.20	1.34
172	-165.09	-13.87	-3.32	165.09	13.87	-1.85
173	-277.49	18.44	2.55	277.49	-18.44	4.45
174	-392.26	-0.80	0.58	392.26	0.80	-1.13
175	1003.53	4.75	2.34	-1003.53	-4.75	2.43
176	196.33	100.89	17.66	-196.33	90.68	-12.77
177	265.59	-4.75	0.01	-265.59	4.75	-1.65
178	810.86	5.10	1.32	-810.86	-5.10	2.95
179	556.91	4.70	1.77	-556.91	-4.70	3.19
180	196.81	-23.39	-3.10	-196.81	23.39	-3.92
181	586.40	6.92	-0.88	-586.40	-6.92	2.21
182	1633.28	-27.29	-5.76	-1633.28	27.29	-0.72
183	-11.59	20.15	4.02	11.59	27.34	-4.88
184	48.97	-34.10	-5.45	-48.97	34.10	-1.08
185	565.34	68.07	4.34	-565.34	-68.07	8.68
186	-15.37	-40.55	-5.47	15.37	40.55	-2.29
187	-248.74	35.96	5.76	248.74	-35.96	5.02
188	523.00	97.73	15.44	-523.00	102.27	-17.71
189	-99.01	-1.51	-0.76	99.01	1.51	-0.03
190	1080.38	0.12	-0.55	-1080.38	-0.12	0.61

191	584.72	-2.92	-1.47	-584.72	2.92	-1.45
192	1071.54	-1.34	0.20	-1071.54	1.34	-0.90
193	-90.83	-5.16	-0.79	90.83	5.16	-1.91
194	-177.75	24.63	3.72	177.75	-24.63	3.67
195	274.50	-10.79	-1.90	-274.50	10.79	-3.74
196	1307.57	97.75	14.11	-1307.57	102.25	-16.36
197	708.65	-7.24	-0.98	-708.65	7.24	-2.80
198	-202.11	-1.14	0.66	202.11	1.14	-1.80
199	760.11	5.98	3.39	-760.11	-5.98	-0.26
200	224.25	6.50	3.15	-224.25	-6.50	0.24
201	-190.75	-13.35	-2.65	190.75	13.35	-1.35
202	161.63	12.15	1.84	-161.63	-12.15	0.48
203	-85.67	-2.03	1.07	85.67	2.03	-1.55
204	224.55	68.28	7.48	-224.55	-68.28	5.58
205	218.62	-85.45	-8.51	-218.62	85.45	-7.84
206	178.43	-11.31	-0.13	-178.43	11.31	-2.03
207	1821.32	4.90	-1.79	-1821.32	42.59	-2.68
208	-259.64	-0.04	-0.01	259.64	0.04	-0.01
209	227.12	-68.22	-5.58	-227.12	68.22	-7.47
210	1819.73	42.61	2.68	-1819.73	4.88	1.80
211	175.86	11.37	2.03	-175.86	-11.37	0.14
212	215.88	85.29	7.83	-215.88	-85.29	8.49
213	164.37	-12.31	-0.49	-164.37	12.31	-1.86
214	-190.75	13.15	2.62	190.75	-13.15	1.33
215	1297.79	102.29	16.39	-1297.79	97.71	-14.10
216	766.61	-5.80	0.31	-766.61	5.80	-3.34
217	-84.08	2.05	1.56	84.08	-2.05	-1.07
218	217.76	-6.32	-0.20	-217.76	6.32	-3.10
219	-192.33	1.	2.75	-716.16	-7.12	0.97
222	-177.75	-25.05	-3.78	177.75	25.05	-3.74
223	1077.56	1.67	0.96	-1077.56	-1.67	-0.09
224	499.90	102.30	17.76	-499.90	97.70	-15.46
225	1088.41	-0.37	-0.72	-1088.41	0.37	0.53
226	-106.97	1.22	-0.09	106.97	-1.22	0.73
227	-247.48	-36.11	-5.04	247.48	36.11	-5.79
228	314.45	1.72	0.98	-314.45	-1.72	-0.24
229	-240.45	0.10	0.74	240.45	-0.10	-0.70
230	37.07	3.51	0.64	-37.07	-3.51	0.41
231	-152.48	-18.47	-3.41	152.48	18.47	-3.36
232	49.28	-11.35	-1.21	-49.28	11.35	-2.01
233	-113.78	14.08	3.59	113.78	-14.08	5.38
234	-133.35	18.50	3.42	133.35	-18.50	3.38
235	-92.70	-14.55	-3.70	92.70	14.55	-5.56
236	50.85	12.61	1.40	-50.85	-12.61	2.18
237	81.78	1.79	0.14	-81.78	-1.79	1.13
238	90.29	-1.87	-0.18	-90.29	1.87	-1.15

239	137.66	2.48	1.32	-137.66	-2.48	2.39
240	273.23	-3.50	0.25	-273.23	3.50	-1.76
241	504.92	1.84	1.47	-504.92	-1.84	-0.10
242	-176.59	-4.44	0.06	176.59	4.44	-1.89
243	-455.79	1.61	1.35	455.79	-1.61	-0.25
244	-99.27	8.77	2.00	99.27	-8.77	1.19
245	209.94	7.04	1.65	-209.94	-7.04	0.78
246	-103.93	-7.73	-1.16	103.93	7.73	-1.16
247	204.02	-3.68	-0.43	-204.02	3.68	-1.05
248	228.64	0.71	0.50	-228.64	-0.71	0.03
249	-126.84	-1.64	0.02	126.84	1.64	-0.64
250	-133.01	1.02	0.61	133.01	-1.02	0.09
251	-99.57	4.10	0.93	99.57	-4.10	0.70
252	184.83	3.26	0.76	-184.83	-3.26	0.46
253	-46.56	-6.48	-0.94	46.56	6.48	-1.00
254	153.10	0.53	0.09	-153.10	-0.53	0.13
255	-0.35	0.56	0.12	0.35	-0.56	0.30
256	-63.70	2.01	0.34	63.70	-2.01	0.43
257	109.37	0.65	0.12	-109.37	-0.65	0.32
258	-65.36	-0.30	-0.22	65.36	0.30	0.10
259	153.78	-1.08	-0.34	-153.78	1.08	-0.05
260	-3.94	-4.16	-0.59	3.94	4.16	-0.66
261	102.21	2.78	0.35	-102.21	-2.78	0.78
262	-150.83	0.44	-0.15	150.83	-0.44	0.48
263	0.51	3.91	0.47	-0.51	-3.91	0.99
264	260.12	0.91	-0.08	-260.12	-0.91	0.71
265	-12.72	-2.86	-0.95	12.72	2.86	-0.20
266	116.33	-1.87	-0.81	-116.33	1.87	0.12
267	68.63	2.40	0.29	-68.63	-2.40	0.44
268	-88.54	1.21	-0.05	88.54	-1.21	0.54
269	-82.16	-0.96	-0.51	82.16	0.96	-0.20
270	134.74	1.22	-0.03	-134.74	-1.22	0.49
271	135.35	-1.08	-0.53	-135.35	1.08	-0.21
272	146.20	-1.35	-0.69	-146.20	1.35	-0.42
273	125.31	-1.88	-0.37	-125.31	1.88	-0.39
274	-79.70	-3.32	-0.65	79.70	3.32	-0.58
275	69.92	-2.61	-2.45	-69.92	2.61	-1.45
276	-28.15	-0.26	-0.12	28.15	0.26	-0.07
277	-31.74	0.27	0.13	31.74	-0.27	0.07
278	-110.43	5.34	1.41	110.43	-5.34	-0.54
279	-58.89	7.48	0.92	58.89	-7.48	1.78
280	26.58	-0.83	-0.24	-26.58	0.83	-0.22
281	92.50	-1.24	-0.26	-92.50	1.24	-0.17
282	-37.82	2.81	1.00	37.82	-2.81	0.46
283	10.38	38.24	4.68	-10.38	36.94	-4.19
284	-66.61	3.23	0.54	66.61	-3.23	0.43

285	11.38	2.67	0.56	-11.38	-2.67	0.35
286	50.21	0.60	0.23	-50.21	-0.60	0.18
287	24.10	0.10	0.08	-24.10	-0.10	-0.04
288	9.73	0.22	0.08	-9.73	-0.22	0.09
289	-24.07	-0.05	0.03	24.07	0.05	-0.05
290	74.37	-3.97	-0.33	-74.37	3.97	-0.80
291	14.46	-2.12	-0.13	-14.46	2.12	-0.50
292	2.42	1.17	0.24	-2.42	-1.17	0.13
293	5.99	0.40	0.16	-5.99	-0.40	0.06
294	33.00	-2.27	-0.48	-33.00	2.27	-0.23
295	41.61	26.06	2.08	-41.61	29.05	-2.91
296	34.22	0.59	0.13	-34.22	-0.59	0.05
297	4.16	-1.40	-0.03	-4.16	1.40	-0.41
298	-13.67	-3.60	-0.38	13.67	3.60	-0.70
299	-4.64	1.20	0.19	4.64	-1.20	0.19
300	42.86	0.34	0.08	- 42.86	-0.34	0.11
301	45.45	-2.64	-0.56	-45.45	2.64	-0.27
302	27.21	-0.47	-0.03	-27.21	0.47	-0.23
303	-1.90	1.34	0.18	1.90	-1.34	0.24
304	43.85	-0.42	-0.10	-43.85	0.42	-0.03
305	-41.78	0.01	0.00	41.78	-0.01	0.00
306	-1.57	-1.30	-0.24	1.57	1.30	-0.17
307	26.99	0.46	0.23	-26.99	-0.46	0.03
308	43.65	0.42	0.03	-43.65	-0.42	0.10
309	43.03	-0.34	-0.11	-43.03	0.34	-0.08
310	45.19	2.60	0.26	-45.19	-2.60	0.55
311	-4.23	-1.19	-0.18	4.23	1.19	-0.19
312	-12.85	3.54	0.69	12.85	-3.54	0.37
313	6.81	1.42	0.41	-6.81	-1.42	0.03
314	42.02	29.15	2.94	-42.02	25.96	-2.06
315	32.12	-0.55	-0.05	-32.12	0.55	-0.13
316	8.39	-0.42	-0.06	-8.39	0.42	-0.17
317	30.75	2.53	0.26	-30.75	-2.53	0.53
318	4.97	-1.40	-0.17	-4.97	1.40	-0.27
319	15.82	37.21	4.30	-15.82	37.97	-4.59
320	74.55	2.23	0.60	-74.55	-2.23	0.11
321	6.00	2.13	0.52	-6.00	-2.13	0.12
322	-16.04	-0.15	0.05	16.04	0.15	-0.09
323	26.25	0.92	0.27	-26.25	-0.92	0.24
324	103.47	2.55	0.38	-103.47	-2.55	0.44
325	-59.33	-2.29	-0.40	59.33	2.29	-0.84
326	-44.83	-0.05	-0.01	44.83	0.05	-0.01
327	-28.65	0.27	0.17	28.65	-0.27	0.00
328	-32.53	-0.28	-0.18	32.53	0.28	-0.00
329	-53.40	-0.01	-0.00	53.40	0.01	-0.01
330	-46.30	-7.67	-1.02	46.30	7.67	-1.75

331	-116.05	-9.31	-1.72	116.05	9.31	0.20
332	155.92	1.29	0.35	-155.92	-1.29	0.71
333	149.93	1.27	0.61	-149.93	-1.27	0.26
334	145.94	-1.46	0.02	-145.94	1.46	-0.57
335	77.53	-2.80	-0.50	-77.53	2.80	-0.34
336	115.95	2.37	-0.08	-115.95	-2.37	0.96
337	293.02	-0.93	-0.76	-293.02	0.93	0.13
338	9.06	-0.26	-0.11	-9.06	0.26	-0.00
339	26.73	-0.71	-0.19	-26.73	0.71	-0.09
340	-8.31	0.07	0.02	8.31	-0.07	0.00
341	5.24	-0.01	-0.04	-5.24	0.01	0.04
342	12.85	-0.11	-0.07	-12.85	0.11	-0.02
343	41.95	-0.17	-0.09	-41.95	0.17	-0.02
344	36.08	0.28	0.00	-36.08	-0.28	0.10
345	36.69	-0.05	-0.02	-36.69	0.05	-0.01
346	3.87	-0.54	-0.12	-3.87	0.54	-0.09
347	-23.54	1.02	0.14	23.54	-1.02	0.16
348	4.88	-0.40	-0.08	-4.88	0.40	-0.08
349	42.42	-0.15	-0.03	-42.42	0.15	-0.08
350	43.35	0.36	0.07	-4 3.35	-0.36	0.06
351	26.29	-0.22	-0.05	-26.29	0.22	-0.10
352	3.67	-0.76	-0.09	-3.67	0.76	-0.19
353	43.95	0.30	0.11	-43.95	-0.30	0.02
354	-48.60	0.02	0.00	48.60	-0.02	0.00
355	43.64	-0.31	-0.02	-43.64	0.31	-0.11
356	42.72	0.15	0.08	-42.72	-0.15	0.03
357	5.09	0.40	0.08	-5.09	-0.40	0.08
358	43.07	-0.38	-0.06	-43.07	0.38	-0.08
359	26.02	0.21	0.10	-26.02	-0.21	0.04
360	3.92	0.77	0.20	-3.92	-0.77	0.10
361	-24.21	-1.07	-0.15	24.21	1.07	-0.17
362	37.97	0.10	0.02	-37.97	-0.10	0.03
363	12.15	0.11	0.02	-12.15	-0.11	0.07
364	0.68	0.60	0.10	-0.68	-0.60	0.13
365	41.59	0.18	0.03	-41.59	-0.18	0.09
366	3.67	-0.03	-0.04	-3.67	0.03	0.03
367	36.31	-0.38	-0.11	-36.31	0.38	-0.02
368	-8.43	-0.21	-0.04	8.43	0.21	-0.02
369	12.35	0.39	0.03	-12.35	-0.39	0.14
370	3.99	-0.21	-0.09	-3.99	0.21	-0.07
371	21.31	0.76	0.10	-21.31	-0.76	0.21
372	55.58	-0.46	-0.15	-55.58	0.46	-0.16
373	14.40	-2.52	-0.35	-14.40	2.52	-0.51
374	19.27	-0.35	-0.01	-19.27	0.35	-0.12
375	-66.90	-3.16	-0.50	66.90	3.16	-0.45
376	134.66	2.14	0.43	-134.66	-2.14	0.43

377	-100.38	1.03	0.20	100.38	-1.03	0.56
378	-90.89	3.98	0.71	90.89	-3.98	0.77
379	-101.37	-1.51	-0.64	101.37	1.51	0.03
380	-9.68	3.43	0.28	9.68	-3.43	1.11
381	-183.35	-0.43	-0.53	183.35	0.43	0.20
382	99.20	-3.33	-0.93	-99.20	3.33	-0.42
383	6.15	-4.47	-1.14	-6.15	4.47	-0.52
384	2.09	4.06	0.64	-2.09	-4.06	0.58
385	-62.95	0.96	-0.01	62.95	-0.96	0.39
386	155.19	1.81	0.13	-155.19	-1.81	0.52
387	146.38	-0.65	-0.38	-146.38	0.65	-0.07
388	-57.89	-2.67	-0.60	57.89	2.67	-0.41
389	88.62	274.46	31.03	-88.62	271.80	-30.13
390	79.96	271.64	30.08	-79.96	274.62	-31.10
391	607.77	2.96	1.43	-607.77	-2.96	1.52
392	-96.79	5.49	1.97	96.79	-5.49	0.90
393	736.54	-0.20	0.39	-736.54	0.20	-0.45
394	-500.34	-5.75	-1.21	500.34	5.75	-0.92
395	-497.59	-4.67	-0.74	497.59	4.67	-1.14

REACOES NOS APOIOS

NO	COORDENADA 1	COORDENADA 2	COORDENADA 3
1	0.00	1888.35	-0.00
41	0.00	1888.35	0.00

O tempo total de processamento utilizado pelo metodo sparse foi de 44.1 segundos.