BT/PEF-8825

UM PROGRAMA PARA SOLUÇÃO DE SISTEMAS LINEARES DE GRANDE PORTE - APLICAÇÃO À ENGENHARIA DE ESTRUTURAS

(*) Priscila Goldenberg
Reyolando Brasil
(*) Professora Assistente Doutora
(recebido em 08/12/88)

1. GENERALIDADES

1.1. Motivação na Engenharia de Estruturas

Na notável síntese de Crandall [1], a análise dos problemas de engenharia por via numérica se faz em dois passos:

- Construção de um modelo matemático da situação física;
- Redução do problema matemático a um procedimento numérico.

Esses problemas recaem, no geral, em três grupos:

a) Problemas de equilíbrio (estáticos);

b) Problemas de valores característicos;

c) Problemas de valor inicial (dinâmicos).

Nas três categorias se consideram tanto os sistemas naturalmente discretos, como os problemas contínuos discretizáveis. Contemplam-se ainda, nos vários casos, problemas de natureza linear e não linear.

Reduzindo o foco à Engenharia de Estruturas, as variáveis básicas com que se trabalha são as ações externas (cargas e reações) e internas (tensões), e os deslocamentos absolutos e relativos (deformações) dos pontos dos sólidos deformáveis que constituem as estruturas.

Vai-se agora tratar de um sistema discreto (como as estruturas reticuladas) ou discretizado via, p. ex., o Método dos Elementos Finitos (*).

(*) Como comentário histórico é interessante notar que esse Método, contribuição original da Engenharia de Estruturas à Física Matemática, que revolucionou a análise numérica, foi primeiro apresentado por Turner, Clough, Martin e Topp [2] em 1956, mesmo ano da edição do trabalho clássico de Crandall, já citado, que, como é claro, não o contém explicitamente.

Sendo f o vetor dos carregamentos nodais (i.e., pontos discretos da estrutura), u o vetor dos deslocamentos nodais, e caracterizando por pontos superpostos suas derivadas temporais, o sistema de equações diferenciais do movimento, na formulação de deslocamentos, se escreve

$$M\,\ddot{u} + C\,\dot{u} + K\,u = f(t) \qquad (1.1)$$

em que M, C e K são, respectivamente, as matrizes de massa, amortecimento e rigidez, constantes no caso do comportamento linear, ou, no geral, função de deslocamento, comportamento do material, histórico, etc.

Trata-se de problema encaixado no grupo dos de valor inicial acima citado. Na integração numérica de (1.1) no tempo, passo a passo, utilizam-se algorítimos em que se adota certa variação de u e suas derivadas no intervalo Δt e, em cada passo, resolve-se um sistema do tipo

$$\hat{K}\,u_{t+\Delta t} = \hat{f}_t \quad \text{ou} \quad \hat{K}\,u_{t+\Delta t} = \hat{f}_{t+\Delta t} \qquad (1.2)$$

em que $\hat{K}$ e $\hat{f}$ dependem do específico esquema adotado.

No caso particular de f nulo, tem-se o problema de valores característicos da determinação das frequências e modos naturais de vibração.

Quando a evolução no tempo das variáveis de estado permitir desprezar em (1.1) os esforços de inércia e amortecimento, recai-se nos problemas de equilíbrio estático na forma

$$K\,u = f \qquad (1.3)$$

que, quando não lineares, implicam no uso de algorítimos incrementais e iterativos, resolvendo-se em cada etapa, partindo das condições conhecidas no passo anterior, um sistema do tipo

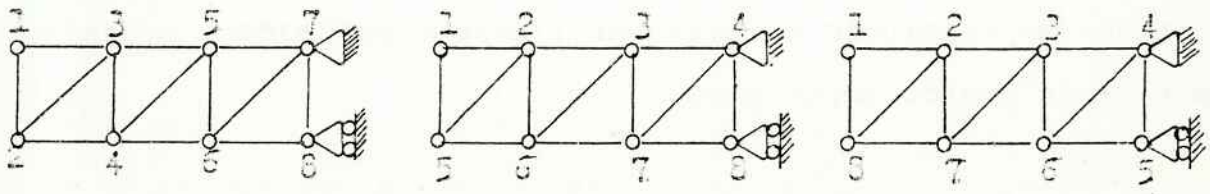
$$K\,\Delta u = \Delta f \qquad (1.4)$$

Estuda-se aqui também o problema da estabilidade do equilíbrio das estruturas, que recai na categoria dos de valores característicos.

Na maioria dos casos das estruturas, os sistemas resultantes são simétricos definidos positivos. Além disso, para estruturas discretas, como as reticuladas, ou discretizadas pelo Método dos Elementos Finitos, os sistemas resultantes são naturalmente esparsos, tanto mais quanto maior for a estrutura. Com uma numeração cuidadosa dos nós pode-se eventualmente concentrar os valores não nulos de K nas proximidades da diagonal principal.

Tais propriedades sugerem esquemas de armazenamento de coeficientes do tipo banda ou envelope, como o "skyline" de Bathe [3], ou de renumeração como o de Cuthill-McKee [4], visando manter e operar o mais possível apenas sobre os elementos não nulos.

Três exemplos de numeração variando de boa a péssima constam da fig. 1.1 com as respectivas "larguras de meia banda" LMB , para uma mesma treliça de 8 nós (2 graus de liberdade por nó).



LMB = 6 LMB = 10 LMB = 16

Fig. 1.1

Com o crescimento cada vez maior do tamanho dos sistemas a resolver e sabendo-se ser essa etapa da análise a que mais tempo consome, cabe um estudo de otimização desses esquemas.

A motivação maior, entretanto, não é nos problemas em que se resolveria um sistema uma única vez, mas sim nos problemas dinâmicos e nós de natureza não linear em que essa operação se repete em grande número, sempre com matrizes que podem variar em cada vez, ou em cada certo número de vezes, no valor numérico de seus coeficientes, mas não em sua estrutura de elementos nulos e não nulos. Esta permanece a mesma já que o sólido analisado é o mesmo, com as mesmas conexões entre nós. Deve ser lembrado aqui que a forma como se comunica aos programas usuais de análise estrutural a ligação entre nós é a clássica tabela de incidência de barras ou de elementos, que nada mais é que um grafo, cuja conceituação se fará com detalhe no ítem 2 deste trabalho.

1.2. Escopo e Estrutura do Trabalho

Conforme ilustrado acima, a utilização de esquemas de armazenamento e rotulação dos coeficientes da matriz K pode influenciar substancialmente no tempo de processamento e utilização de memória de computador na resolução de sistemas lineares definidos positivos esparsos de grande porte tipo

$$K\,u = f \qquad (1.5)$$

Na seção seguinte apresenta-se a descrição de um algorítimo computacional visando a resolução de sistemas dessas características pelo método de Cholesky com armazenamento de coeficientes na forma de envelope e prévia reordenação do mesmo para otimização de memória e número de operações utilizando o esquema de Cuthill-McKee.

A implementação do algorítimo fez-se em microcomputador compatível com o PC da IBM, através de programa computacional em linguagem MS-FORTRAN-77 descrito na seção 3, cuja listagem completa é fornecida no Apêndice.

Na seção 4 apresenta-se um exemplo voltado para a área de Engenharia de Estruturas na forma da análise estática de uma viga balcão modelada como pórtico espacial bi-engastado de três barras. Embora se trate de um sistema de apenas 12 equações simultâneas, o efeito do algorítimo de otimização é extraordinário, como lá se verá.

2. SOLUÇÃO DE SISTEMAS LINEARES ESPARSOS DE GRANDE PORTE SIMÉTRICOS DEFINIDOS POSITIVOS

2.1. Descrição do Problema

Nesta secção será estudada a solução de sistemas lineares esparsos de grande porte simétricos definidos positivos. O método utilizado será o de Cholesky.

Seja dado um sistema de equações lineares

$$K\,u = f \tag{2.1}$$

onde K é uma matriz real NxN, simétrica definida positiva, u e f vetores de R^N. Aplicando o método de Cholesky obtem-se uma fatorização triangular

$$K = LL^T \tag{2.2}$$

onde L é triangular inferior com elementos positivos na diagonal. Uma matriz L é triangular inferior (superior) se $\ell_{ij} = 0$ para $i < j$ $(i > j)$. O superescrito T indica transposta.

Demonstra-se [4] que esta fatorização sempre existe quando K é simétrica definida positiva.

Utilizando (2.1) e (2.2) obtem-se

$$LL^T\,u = f \tag{2.3}$$

e substituindo $v = L^T u$ é claro que obtem-se u resolvendo os siste-

mas triangulares

$$L\,v = f \tag{2.4}$$

e

$$L^T u = v \tag{2.5}$$

Diversos exemplos de aplicações do método de Cholesky a matrizes esparsas K mostram que a matriz sofre um "preenchimento", isto é, L possui elementos não nulos em posições que eram nulas na parte triangular inferior de K. Entretanto observa-se que em diversos problemas com matrizes esparsas uma reordenação de linhas e colunas da matriz K pode reduzir a quantidade de preenchimentos e portanto economizar tempo e memória do computador. O estudo de um algorítimo que automaticamente efetua esta reordenação será apresentado na secção 2.2, juntamente com um esquema de armazenamento para a matriz L do sistema reordenado.

O problema pode ser esquematizado em 3 etapas:

i) Encontrar uma "boa ordenação", ou seja, encontrar uma matriz de permutação P para a dada matriz K, com respeito ao método de armazenamento escolhido.

ii) Determinar informações necessárias sobre o fator de Cholesky L de PKP^T para fixar um esquema apropriado de armazenamento.

iii) Efetuar os cálculos numéricos, que podem ser divididos em 2 partes:

a) Efetuar a fatorização da matriz PKP^T em LL^T

b) Resolver os sistemas triangulares $L\,v = f$ e $L^T z = v$. Dai $u = P^T z$.

Estas 3 etapas podem ser feitas independentemente uma da outra. Esta independência além de possuir a vantagem de modulação nos programas, permite a resolução de diversos problemas com a matriz K possuindo a mesma forma, resolvendo uma só vez as duas primeiras etapas e repetindo a 3ª. etapa para cada problema.

2.2. Algorítmo de ordenação de envelope

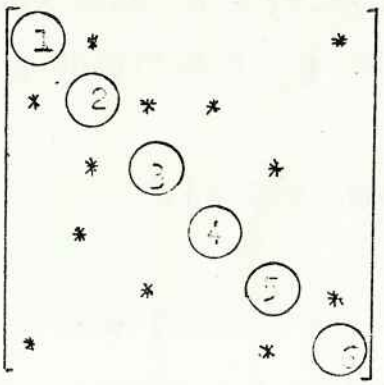
Nesta secção será apresentado um algorítimo de ordenação de envelope [4] . Inicialmente na secção 2.2.1 serão apresentadas algumas definições e convenções relativas a teoria de grafos e sua correspondência com matrizes. Na secção 2.2.2 serão feitas algumas considerações sobre o método de armazenamento denominado envelope e finalmente na secção 2.2.3 será descrito o algorítimo de ordenação de envelope.

2.2.1. Noções sobre teoria de grafos aplicada às matrizes

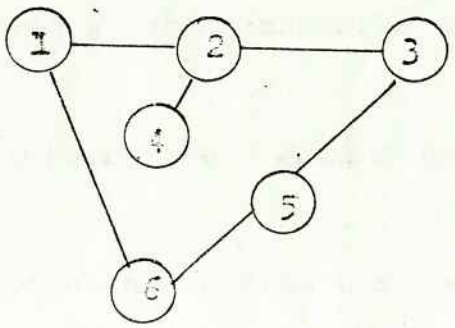
Nesta secção serão apresentadas algumas convenções e definições, bem como algumas noções sobre teoria de grafos e sua correspondência com as matrizes.

Um grafo $G = (U,E)$ constitui-se de um conjunto de nós U juntamente com um conjunto de arestas E que são pares não ordenadas de vértices. Uma ordenação (rotulação) α de $G = (U,E)$ é simplesmente uma aplicação de $\{1, \ldots, N\}$ sobre U onde N denota o número de nós de G. A menos de menção contraria os grafos serão não ordenados; o grafo G ordenado por α será denotado por $G^{\alpha} = (U^{\alpha},E^{\alpha})$. Como o objetivo é introduzir grafos para o estudo de matrizes esparsas, esta relação será estabelecida a seguir.

Seja K uma matriz simétrica NxN. O grafo ordenado de K, denotado por $G^k = (U^k, E^k)$ é um grafo no qual os N vértices de G^k são numerados de 1 a N e $\{ui, uj\} \in E^k$ se e somente se Kij = Kij $\neq$ 0, i $\neq$ j. Aqui u_i está indicando o nó de U^k com rótulo i. Para qualquer matriz NxN de permutação, P $\neq$ I, os grafos não rotulados de K e de PKP^T são os mesmos mas as rotulações associadas são diferentes. O exemplo seguinte facilita a compreensão deste fato. Considere a matriz K abaixo e seu grafo associado



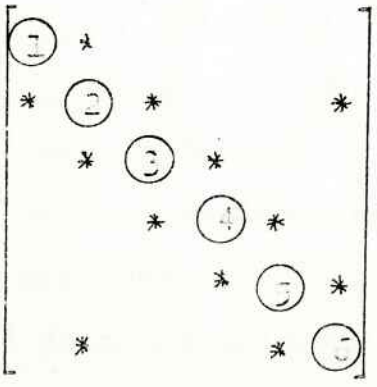
Matriz K



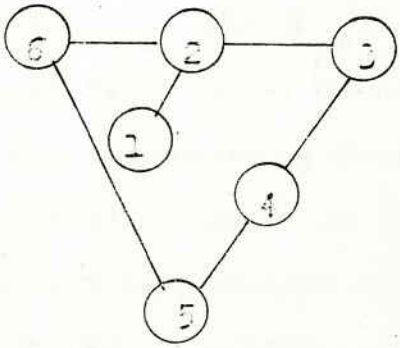
Grafo G^k

O símbolo * indica os elementos não nulos de K.

Considere a seguir uma permutação P de K. Ter-se-á



Matriz PKP^T



GRAFO G^{PKP^T}

Daí, um grafo não rotulado de K representa a estrutura de K sem sugerir qualquer ordenação particular. Ele representa a classe de equivalencia das matrizes PKP^T onde P é qualquer matriz de permutação NxN. Encontrar uma boa permutação de K pode ser vista como uma boa rotulação de seu grafo.

No que segue, referindo-se a uma matriz correspondente ao grafo G, ou especificar-se-á a ordenação α de G ou estará implícito que alguma ordenação foi assumida arbitrariamente.

Dois nós u e v de G são adjacentes se $\{u,v\} \in E$. Para VCU o conjunto de adjacencias de V, denotado por Adj(V) é definido por

$$Adj(V) = \{u \in U-V \mid \{u,v\} \in E \text{ para algum } v \in V\}.$$

Quando V possui um único nó v escrever-se-á simplesmente Adj(v).

Para VCU, o grau de V, denotado por Deg(V) é o número $|Adj(V)|$ onde $|S|$ denota o número de elementos do conjunto S. Quando V possui um único nó v escrever-se-á simplesmente Deg(v).

Um subgrafo G' = (U',E') de G é um grafo no qual U'CU e E'CE.

Sejam u e v nós distintos em G. Um caminho de u a v de comprimento $\ell \geq 1$ é um conjunto ordenado de $\ell+1$ nós distintos $(v_1, \ldots, v_{\ell+1})$ tais que $v_{i+1} \in Adj(v_i)$, $i = 1, \ldots, \ell$ com $v_i = u$ e $v_{\ell+1} = v$. Um grafo é conexo se todo par de nós distintos é unido por pelo menos um caminho. Caso contrário G é desconexo e consiste de duas ou mais componentes conexas. Em termos de matrizes se G é desconexo e constituído de m componentes conexas, e cada componente conexa é rotulada consecutivamente, a correspondente matriz será bloco diagonal, com cada bloco diagonal correspondendo a uma componente conexa.

Seja G = (U,E) um grafo com N nós. Uma lista de adjacencias para $u \in U$ é uma lista contendo todos os nós em Adj(u). Uma estrutura de adjacencias para G é simplesmente um conjunto de listas de adjacencias para todos $u \in U$. Tal estrutura pode ser implementada de modo simples e econômico, armazenando as listas de adjacencias sequencialmente em um vetor unidimensional ADJNCY juntamente com um vetor de índices XADJ de comprimento N+1, contendo os apontadores para o começo de cada lista de adjacencias em ADJNCY.

Observe-se que o armazenamento de um grafo em (XADJ,ADJNCY) implica em uma particular rotulação do grafo. Esta ordenação será referida como ordenação original. Quando uma subrotina encontra uma nova ordenação, a ordenação será armazenada num vetor PERM de comprimento N, onde PERM(i) = m significa que o nó original m é o i-ésimo nó na nova ordenação. Associado ao vetor de permutação define-se o vetor de permutação inversa INVP de comprimento N que satisfaz INVP(PERM(i)) = i.

Finalmente, em algumas subrotinas, somente certos subgrafos de G serão considerados. Para tanto define-se um vetor MASK de comprimento N, onde MASK(i) $\neq$ 0 se o nó i for considerado. Além disso em certos casos, inicialmente um só nó é rotulado. Tal nó será denotado por ROOT, com MASK(ROOT) $\neq$ 0 e as subrotinas considerarão a componente conexa do grafo que contém o nó ROOT.

2.2.2. Método do envelope

Seja $K = [K_{ij}]$ uma matriz simétrica definida positiva NxN. Para a i-ésima linha de K, i = 1, ... , N, define-se

$$f_i(K) = \min \left\{ j \mid K_{ij} \neq 0 \right\}$$

e

$$\beta_i(k) = i - f_i(K).$$

O número $f_i(K)$ indica o índice da coluna onde aparece o primeiro elemento não nulo da i-ésima linha de K. Como todos os elementos da diagonal de K são positivos, tem-se

$$f_i(K) \leqslant i \qquad \text{e} \qquad \beta_i(K) \geqslant 0.$$

Define-se o tamanho da banda de K por

$$\beta(K) = \max \left\{ \beta_i(K) \mid 1 \leqslant i \leqslant N \right\}$$

e o número $\beta_i(K)$ será denominado o tamanho da banda da i-ésima linha de K.

O envelope de K, denotado por Env(K) é definido por

$$\text{Env}(K) = \left\{ \{i,j\} \mid 0 < i - j \leqslant \beta_i(K) \right\}$$

A quantidade $|\text{Env}(K)|$ é denominada tamanho do envelope de K e é dada por

$$|\text{Env}(K)| = \sum_{i=1}^{N} \beta_i(K)$$

Exemplo

$$\begin{bmatrix}
K_{11} & K_{12} & & K_{14} & & & \\
K_{21} & K_{22} & & & & & \\
 & & K_{33} & 0 & K_{35} & K_{36} & \\
K_{41} & 0 & 0 & K_{44} & K_{45} & & \\
 & & K_{53} & K_{54} & K_{55} & & K_{57} \\
 & & K_{63} & 0 & 0 & K_{66} & K_{67} \\
 & & & & K_{75} & K_{76} &
\end{bmatrix}$$

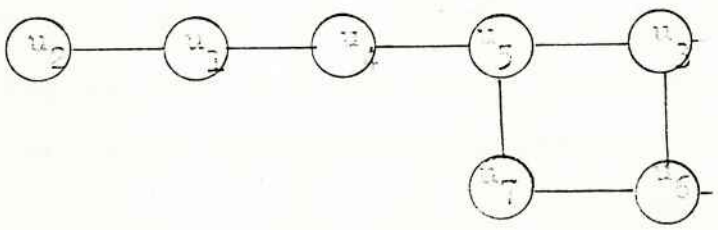
Verifica-se que $Env(K) = En_V(L + L^T)$.

Seja $G^K = (U^K, E^K)$ o grafo associado a K, onde o conjunto de nós é rotulado de acordo com a definição dada pela matriz K:

$$U^K = \{u_1, \ldots, u_N\}$$

Para $i < j$, $\{i,j\} \in Env(K) \Leftrightarrow u_i \in Adj(\{u_1, \ldots, u_i\})$

O grafo associado ao exemplo anterior é dado por



2.2.3. Ordenação de envelope

2.2.3.1. Algorítmo reverso de Cuthill-McKee

O algorítmo reverso de Cuthill-McKee (RCM) é um algorítmo que reduz o tamanho da banda de uma matriz via minimização local dos β_i. Isto sugere que este método pode reduzir o envelope $\sum \beta_i$ de uma matriz.

A descrição do algorítmo RCM será feita para um grafo conexo. No caso de grafos desconexos pode-se aplicar este algorítmo para cada componente conexa do grafo.

Passo 1 - Determine um nó inicial r e atribua $u_1 \leftarrow r$ (*)

Passo 2 - Para i = 1, ... , N encontre todos os nós vizinhos do nó u_i não rotulados e numere-os em ordem crescente dos graus.

Passo 3 - A ordenação reversa de Cuthill-McKee é dada por $v_1, \ldots, v_N$ onde $v_i = u_{N-i+1}$ para i = 1 , ... , N.

A implementação do algorítmo acima será efetuada nas subrotinas RCM, GENRCM, DEGREE descritas na secção 3.

2.2.3.2. Determinação de um nó inicial

Passemos à análise do problema de determinação de um nó inicial para o algorítmo RCM. O objetivo é encontrar um par de nós que possuam máxima ou próxima da máxima distância (definida abaixo). Diversas experiências mostram que tais nós são convenientes como nós iniciais para o algorítmo RCM.

A distância d(u,v) entre os nós u e v de um grafo conexo G é o comprimento do menor caminho ligando u a v. A excentricidade do nó

(*) A tarefa de determinar um nó inicial será considerada logo após a descrição do algorítmo.

u é definida por $\ell(u) = \max \{ d(u,v) \mid v \in U \}$. O algorítmo que será descrito é baseado na construção de uma estrutura de níveis rotulada que passamos a descrever.

Dado um nó $u \in U$ a estrutura de níveis rotulados em u é uma partição $L(u)$ de U satisfazendo

$$L(u) = \{ L_0(u), L_1(u), \ldots, L_{\ell(u)}(u) \}$$

onde

$$L_0(u) = \{u\}$$
$$L_1(u) = Adj(L_0(u))$$

e

$$L_i(u) = Adj(L_{i-1}(u)) - L_{i-2}\{u\}, \quad i = 1, \ldots, \ell(u)$$

Utilizando a notação acima, o algorítmo é o seguinte

Passo 1 - Escolha um nó arbitrário r em U.

Passo 2 - Construa uma estrutura de níveis rotulada em r: $L(r) = \{L_0(r), L_1(r), \ldots, L_{\ell(r)}(r)\}$.

Passo 3 - Escolha um nó u em $L_{\ell(r)}(r)$ de grau mínimo.

Passo 4 - a) Construa uma estrutura de níveis rotulada em u

$$L(u) = \{ L_0(u), L_1(u), \ldots, L_{\ell(u)}(u) \}.$$

b) Se $\ell(u) > \ell(r)$, coloque $r \leftarrow u$ e vá ao passo 3.

Passo 5 - O nó u é o desejado.

A implementação do algorítmo acima será efetuado nas subrotinas FNROOT e ROOTLS descritas na secção 3.

2.2.3.3. Esquema de armazenamento de envelope

O esquema de armazenamento de envelope é feito de modo que para cada linha todas as entradas a partir do primeiro elemento não nulo até a diagonal são armazenados. Estas porções são armazenadas de modo sequencial em um vetor unidimensional ENV. Mais especificadamente o vetor ENV contém o envelope da matriz K. A diagonal será armasenada num vetor separado DIAG. Um vetor de índices auxiliar XENV de comprimento N será utilizado para apontar o início de cada porção da linha. Desta forma o vetor XENV permite localizar qualquer componente não nula convenientemente. A subrotina FNENV descrita na secção 3, determinará os vetores ENV e XENV.

2.3. Comentários finais

Antes de concluir esta secção, será efetuado um esquema das etapas computacionais que serão descritas na secção 3.

Conforme já mencionado, o problema pode ser esquematizado em 3 etapas:

i) Encontrar uma boa ordenação, utilizando as subrotinas RCM, DEGREE, FNROOT, ROOTLS e GENRCM;

ii) Fixar um esquema apropriado de armazenamento utilizando a subrotina FNENV;

iii) Efetuar cálculos numéricos:

a) Fatorização da matriz PKP^T em LL^T utilizando as subrotinas PAPT, ESFCT;

b) Resolução de $Lv = f$ utilizando a subrotina ELSLV;

c) Resolução de $L^t z = v$ utilizando a subrotina EUSLV.

3. DESCRIÇÃO DO PROGRAMA COMPUTACIONAL

Nesta secção descrever-se-á o programa PARENV (escrito em linguagem FORTRAN-77 e implementado num microcomputador compatível com PC-XT(IBM)) que resolve um sistema linear de equações pelo método descrito na secção 2. Algumas subrotinas aqui descritas foram extraídas de [4].

3.1. Utilização do programa PARENV

O programa PARENV para resolução de sistemas de equações lineares simultâneas definidos positivos esparsos de grande porte com aplicação de um esquema de otimização de memória e tempo de processamento, é constituído das etapas:

1) Comandos de leitura e impressão de dados.
2) Resolução do problema pela chamada das subrotinas adequadas.

A seguir se descreve a forma de se utilizar o programa.

3.1.1. Leitura de dados

1) Com um microcomputador compatível com o PC-XT (IBM) acoplado a uma impressora, ligados e sob comando de seu sistema operacional, introduz-se o disquete contendo o programa PARENV.EXE e digita-se o código PARENV.
2) Após a tela de apresentação, a mensagem
   - INDIQUE O NUMERO DE EQUAÇOES
   solicitará a digitação desse parâmetro.

3) O programa colocará a seguir a mensagem

- ENTRADA DO VETOR DE ADJACENCIAS=ADJNCY
  E DE SEUS INDICADORES=XADJ. OPÇOES:
  A) GERA-LOS A PARTIR DO GRAFO ASSOCIADO (DIGITE Ø)
  B) FORNECE-LOS DIRETAMENTE (DIGITE 1)

Se digitado o código Ø, o programa solicita

- ENTRE NUMERO DE ARESTAS DO GRAFO

com o que deve-se entrar esse valor que é o número de segmentos conectando nós do grafo associado à matriz de coeficientes do sistema segundo definição da seção 2 deste trabalho. A seguir a mensagem

- PARA CADA ARESTA ENTRE NO INICIAL, NO FINAL

solicitará a entrada da tabela de incidência de arestas, operação auxiliada pela repetida impressão na tela de

- ARESTA i (i=1,2,...,número de arestas)

Se digitado o código 1, surgirá a mensagem

- ENTRE O VETOR INDICADOR DE ADJACENCIAS=XADJ

solicitando a entrada um a um dos números que apontarão no vetor de adjacências ADJNCY o endereço inicial da lista de nós conectados a cada número de equação, conforme definição dada na seção 2 deste trabalho. A seguir a mensagem

- ENTRE O VETOR DE ADJACENCIAS=ADJNCY

Solicitará a entrada, um a um, dos elementos desse vetor, que nada mais é que uma sequência de listas de nós conectados a cada número de equação, conforme seção 2.

4) A partir dos dados fornecidos em 3) o programa processa o tamanho do envelope da matriz original e de sua largura de banda soltando na impressora acoplada as mensagens

- TAMANHO DO ENVELOPE DA MATRIZ ORIGINAL E -----

- LARGURA DE BANDA ORIGINAL E ------

5) A seguir surge a mensagem

- INDIQUE O VETOR ENVELOPE DA MATRIZ ORIGINAL

solicita a digitação seguida, um a um, da esquerda para a direita em cada linha, dos coeficientes da matriz original a partir do primeiro elemento não nulo até a diagonal, não incluída, em formato livre.

Os elementos da diagonal serão a seguir fornecidos um a um, pela ordem das equações, após a solicitação

- INDIQUE A DIAGONAL DO SISTEMA ORIGINAL

6) Segue-se a mensagem

- INDIQUE O VETOR SEGUNDO MEMBRO DO SISTEMA

após a qual digita-se os elementos desse vetor um a um.

7) Embora a intenção do programa PARENV seja otimizar memória e tempo de processamento, é fornecida a possibilidade de se resolver o problema de forma não automática, fornecendo-se uma permutação diretamente, para, por exemplo, possibilitar comparações de performance. A opção é exercida digitando o código correspondente após a mensagem

- INDIQUE SE A OTIMIZAÇAO DEVERA SER EFETUADA

  CASO AFIRMATIVO DIGITE ZERO;CASO NEGATIVO DIGITE 1

Na hipótese afirmativa os elementos do vetor de permutação escolhido pelo usuário deverão ser digitados após a mensagem

- INDIQUE O VETOR DE PERMUTAÇOES

3.1.2. Saída de resultados

1) Segue-se a saída de resultados através da impressora, começando pelo fornecimento da extensão do vetor envelope e largura de banda do sistema após a otimização na forma

- TAMANHO DO ENVELOPE APOS PERMUTAÇAO E ---
- LARGURA DE BANDA APOS PERMUTAÇAO E ----

seguido da impressão do vetor de resultados do sistema na ordem original, já que o programa se encarrega de fazer a permutação inversa. A impressão, nesta versão, é feita em formato livre, com precisão dupla.

2) Surge a seguir na tela a mensagem

- QUER ENTRAR OUTRO VETOR RHS?(1=NAO,Ø=SIM)

para possibilitar a solução de um novo problema com a mesma matriz K de coeficientes, cuja decomposição já foi feita e está armazenada, e um outro vetor segundo membro. Tal operação poderia ser necessária, por exemplo, para analisar uma mesma estrutura sob efeito de diversos carregamentos.

3.2. Descrição das subrotinas

1) Subrotina ROOTLS

Objetivo: gerar uma estrutura de níveis rotulada no nó inicial chamado ROOT.

Utilização: CALL ROOTLS (ROOT, XADJ, ADJNCY, MASK, NLVL, XLS, LS).

Argumentos: ROOT = variável inteira de entrada contendo o nó a partir do qual a estrutura de níveis será rotulada.

(XADJ, ADJNCY) = par de vetores inteiros de entrada contendo a estrutura de adjacência do grafo.

MASK = vetor inteiro de entrada especificando o subgrafo a ser utilizado.

NLVL = variável inteira de saída contendo o número de níveis na estrutura.

(XLS, LS) = par de vetores inteiros de saída contendo a estrutura de níveis rotulados.

Subrotinas requeridas: nenhuma.

2) Subrotina FNROOT

Objetivo: Encontrar um nó inicial para o algorítmo RCM.

Utilização: CALL FNROOT (ROOT, XADJ, ADJNCY, MASK, NLVL, XLS, LS).

Argumentos: (XADJ, ADJNCY) = par de vetores inteiros de entrada contendo a estrutura de adjacências do grafo.

MASK = vetor inteiro de entrada especificando o subgrafo a ser utilizado.

ROOT = variável inteira que na entrada (juntamente com MASK) define a componente conexa na qual o nó inicial para RCM deverá ser procurado; na saída conterá o nó desejado.

NLVL = variável inteira de saída contendo o número de níveis na estrutura.

(XLS, LS) = par de vetores inteiros de saída contendo a estrutura de níveis rotulados.

Subrotinas requeridas: ROOTLS.

3) Subrotina DEGREE

Objetivo: Calcular o grau dos nós da componente conexa especificada por MASK e ROOT.

Utilização: CALL DEGREE (ROOT, XADJ, ADJNCY, MASK, DEG, CCSIZE, LS)

Argumentos: ROOT = variável inteira que na entrada define a componente conexa.

(XADJ, ADJNCY) = par de vetores inteiros de entrada contendo a estrutura de adjacências do grafo.

MASK = vetor inteiro de entrada especificando o subgrafo a ser utilizado.

DEG = vetor inteiro de saída contendo os graus.

CCSIZE = parâmetro inteiro de saída especificando o tamanho da componente conexa.

LS = vetor temporário usado para armazenar os nós das componentes níveis por níveis.

Subrotinas requeridas: nenhuma.

4) Subrotina RCM

Objetivo: RCM ordena os nós da componente conexa do grafo especificada por MASK e ROOT, utilizando o algorítmo RCM.

Utilização: CALL RCM (ROOT, XADJ, ADJNCY, MASK, PERM, CCSIZE, DEG)

Argumentos: ROOT = variável inteira de entrada define o nó inicial a ser considerado.

(XADJ, ADJNCY) = par de vetores inteiros de entrada contendo a estrutura de adjacências do grafo.

MASK = vetor inteiro de entrada especificando o subgrafo a ser utilizado.

PERM = vetor inteiro de saída contendo a ordenação.

CCSIZE = parâmetro inteiro de saída especificando o tamanho da componente conexa utilizada.

DEG = vetor temporário usado para armazenar os graus dos nós do subgrafo utilizado.

Subrotinas requeridas: DEGREE

5) Subrotina GENRCM

Objetivo: Encaminhar a ordenação pelo algorítmo reverso de Cuthill-McKee para um grafo geral.

Utilização: CALL GENRCM (NEQNS, XADJ, ADJNCY, PERM, MASK, XLS)

Argumentos: NEQNS = variável inteira de entrada especificando o número de equações.

(XADJ, ADJNCY) = par de vetores inteiros de entrada contendo a estrutura de adjacências do grafo.

PERM = vetor inteiro de saída contendo a ordenação pelo algorítmo RCM.

MASK = vetor inteiro de entrada de trabalho, utilizado para marcar as variáveis que já foram numeradas. É inicializado em 1 e depois atribuindo valor nulo quando o nó já foi rotulado.

XLS = vetor inteiro de trabalho, contendo os índices para a estrutura de níveis.

Subrotinas requeridas: FNROOT, RCM.

6) Subrotina FNENV

Objetivo: Encontrar a estrutura de envelope de uma matriz permutada.

Utilização: CALL FNENV (NEQNS, XADJ, ADJNCY, PERM, INVP, XENV, ENVSZE, BANDW).

Argumentos: NEQNS = variável inteira de entrada especificando o núme

ro de equações.

(XADJ, ADJNCY) = par de vetores inteiros de entrada contendo a estrutura de adjacências do grafo.

(PERM, INVP) = par de vetores inteiros de entrada, contendo informações sobre a ordenação da matriz.

XENV = vetor inteiro de saída contendo os índices para a estrutura de níveis a ser utilizada no armazenamento do envelope inferior (ou superior) da matriz reordenada.

ENVSZE = variável inteira de saída, contendo o tamanho do envelope.

BANDW = variável inteira de saída, contendo o tamanho da banda.

Subrotinas requeridas: nenhuma.

7) Subrotina ELSLV

Objetivo: Resolução do sistema triangular inferior LX = RHS. O fator L deverá ser armazenado no formato envelope.

Utilização: CALL ELSLV (NEQNS, XENV, ENV, DIAG, RHS).

Argumentos: NEQNS = variável inteira de entrada especificando o número de equações.

XENV = vetor inteiro de entrada contendo os índices para a estrutura de níveis utilizada no armazenamento do envelope.

ENV = vetor real de entrada contendo o envelope de L.

DIAG = vetor real de entrada contendo a diagonal de L.

RHS = vetor real que na entrada contém o segundo membro da equação, e na saída, contém o vetor solução.

Subrotinas requeridas: nenhuma.

Subrotinas requeridas: ELSLV.

10) Subrotina PAPT

Objetivo: Subrotina para efetuar permutação PAPT dos coeficientes não diagonais de uma matriz original A fornecida sob forma de envelope.

Utilização: CALL PAPT (NEQNS, INVP, ORIENV, XORENV, XENV, ENV).

Argumentos: NEQNS = variável inteira de entrada contendo o número de equações do sistema.

INVP = variável inteira unidimensional de entrada, contendo o vetor de permutação inversa ao PERM.

XENV = variável inteira unidimensional contendo o vetor de apontadores do envelope da matriz permutada.

ENV = variável real unidimensional de saída contendo os coeficientes não diagonais de cada linha da matriz permutada a partir do primeiro não nulo, da esquerda para a direita, em sequência.

ORIENV = variável real unidimensional de entrada contendo os coeficientes não diagonais de cada linha da matriz original a partir do primeiro não nulo, da esquerda para a direita, em sequência.

XORENV = variável inteira unidimensional de entrada contendo o vetor indicador do envelope original.

Subrotinas requeridas: nenhuma

11) Subrotina PERMVT

Objetivo: Efetuar permutação dos elementos de um vetor segundo um vetor de permutação dado.

Utilização: CALL PERMVT (NEQNS, PERM, VT).

Argumentos: NEQNS = variável inteira de entrada contendo o número

8) Subrotina EUSLV

Objetivo: Resolução do sistema triangular superior UX = RHS. O fator U deverá ser armazenado no formato envelope.

Utilização: CALL EUSLV (NEQNS, XENV, ENV, DIAG, RHS).

Argumentos: NEQNS = variável interior de entrada especificando o número de equações.

XENV = vetor inteiro de entrada contendo os índices para a estrutura de níveis utilizada no armazenamento do envelope.

ENV = vetor real de entrada contendo o envelope de U.

DIAG = vetor real de entrada contendo a diagonal de U.

RHS = vetor real que na entrada contém o segundo membro da equação, e na saída, contém o vetor solução.

Subrotinas requeridas: nenhuma

9) Subrotina ESFCT

Objetivo: Efetuar a fatorização de uma matriz definida positiva R em $L*L^T$. A matriz R deverá ser armazenada no formato envelope.

Utilização: CALL ESFCT (NEQNS, XENV, ENV, DIAG, IFLAG).

Argumentos: NEQNS = variável inteira de entrada especificando o número de equações.

XENV = vetor inteiro de entrada contendo os índices para a estrutura de níveis utilizada no armazenamento do envelope.

ENV = vetor real que na entrada contém o envelope de L e na saída contém o envelope de L.

DIAG = vetor real que na entrada contém a diagonal de R e na saída contém o envelope de L.

IFLAG = variável real de saída na qual é atribuído valor 1 se R não é simétrica definida positiva.

de equações do sistema.

PERM = variável inteira unidimensional de entrada contendo o vetor de permutação.

VT = variável real unidimensional contendo na entrada os elementos do vetor que sofrerá a permutação indicada por PERM e na saída o vetor permutado.

Subrotinas requeridas: nenhuma.

12) Subrotina GERADJ

Objetivo: Gerar automaticamente o vetor de adjacência ADJNCY e seu correspondente vetor de indicadores XADJ a partir do grafo associado à matriz de coeficientes, fornecido na forma de uma tabela de incidência de conecções entre nós.

Utilização: CALL GERADJ (NM1, NI, NF, XADJ, ADJNCY).

Argumentos: NM1 = variável inteira de entrada com o valor do número de equações do sistema mais um, ou seja, o número de elementos do vetor XADJ.

NI = variável inteira de entrada com o número do nó escolhido como inicial de uma dada aresta do grafo.

NF = variável inteira de entrada com o número do nó escolhido como final de uma dada aresta do grafo.

XADJ = variável inteira unidimensional de saída contendo o vetor de apontadores da posição da lista de adjacências de cada equação no vetor ADJNCY.

ADJNCY = variável inteira unidimensional de saída contendo em sequência as listas de adjacências de cada equação do sistema.

Subrotinas requeridas: nenhuma.

4. EXEMPLO NA ENGENHARIA DE ESTRUTURAS

Como demonstração da aplicação do programa PARENV para solução de sistemas de equações lineares simultâneas definidos positivos esparsos de grande porte na engenharia de estruturas, apresenta-se a análise estática de uma viga balcão modelada como pórtico espacial conforme Brasil [5].

Na Fig. 4.1 apresenta-se a geometria e características físicas da viga balcão em questão, de dimensões bem usuais da prática da engenharia civil. Na Fig. 4.2 é indicada a numeração dos 12 graus de liberdade originalmente adotados, para a viga modelada como pórtico espacial, também a que usualmente se faria nos programas convencionais de análise estrutural.

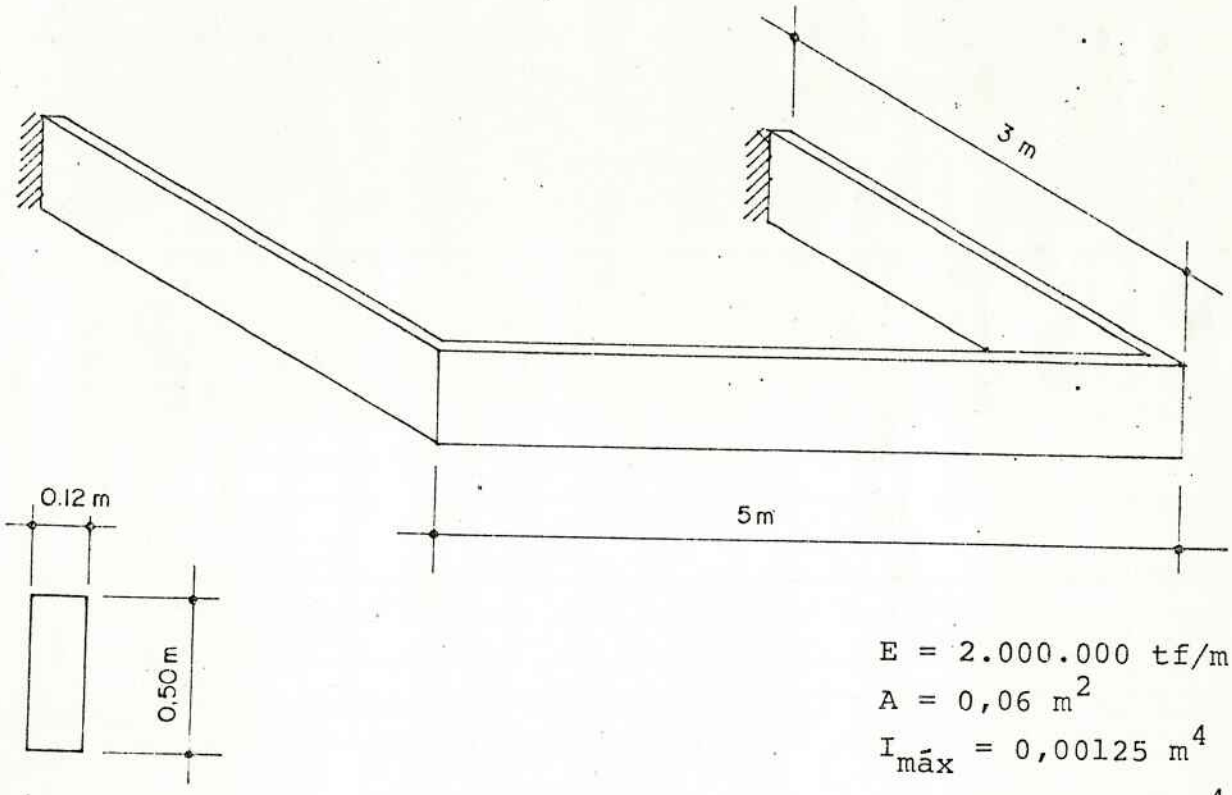
A matriz de rigidez original da estrutura (12x12) é apresentada na figura 4.3, com os coeficientes calculados manualmente até uma precisão típica de trabalho efetuado com auxílio de calculadoras de bolso.

O vetor de carregamento f que a seguir é listado corresponde aproximadamente a uma carga uniformemente distribuída de 1 (uma) tonelada-força por metro linear de viga de cima para baixo, ou seja, no sentido da gravidade.

$$f^t = (0.0,-4.0,0.0,-.75,0.0,-2.0,0.0,-4.0,0.0,-.75,0.0,2.0)$$

Os resultados obtidos, a pesar das dimensões ainda modestas da estrutura analisada, são simplesmente dramáticos: o envelope, que na matriz original continha 52 elementos passou para apenas 20, e a largura de banda diminuiu de 10 para somente 3.

Cabe frisar a precisão dos deslocamentos resultantes da análise não afetada pelo fato de se tratar de um sistema muito mal condicionado, com o número de condição espectral (razão entre o maior e menor valor próprio da matriz de rigidez) igual a 2074, conforme Brasil [5]. Este problema seria melhor resolvido modelando a viga balcão do exemplo como grelha e não como pórtico espacial, em que comparecem modos de deslocamento muito "rígidos" (deformação axial das barras) coexistindo com outros muito "flexíveis" (flexão e torção).



$E = 2.000.000$ tf/m

$A = 0,06\ m^2$

$I_{máx} = 0,00125\ m^4$

$I_{mín} = 0,000072\ m^4$

$I_{tor} = 0,000244\ m^4$

$G = 0,4\ E$

Fig. 4.1 - Viga Balcão: Geometria

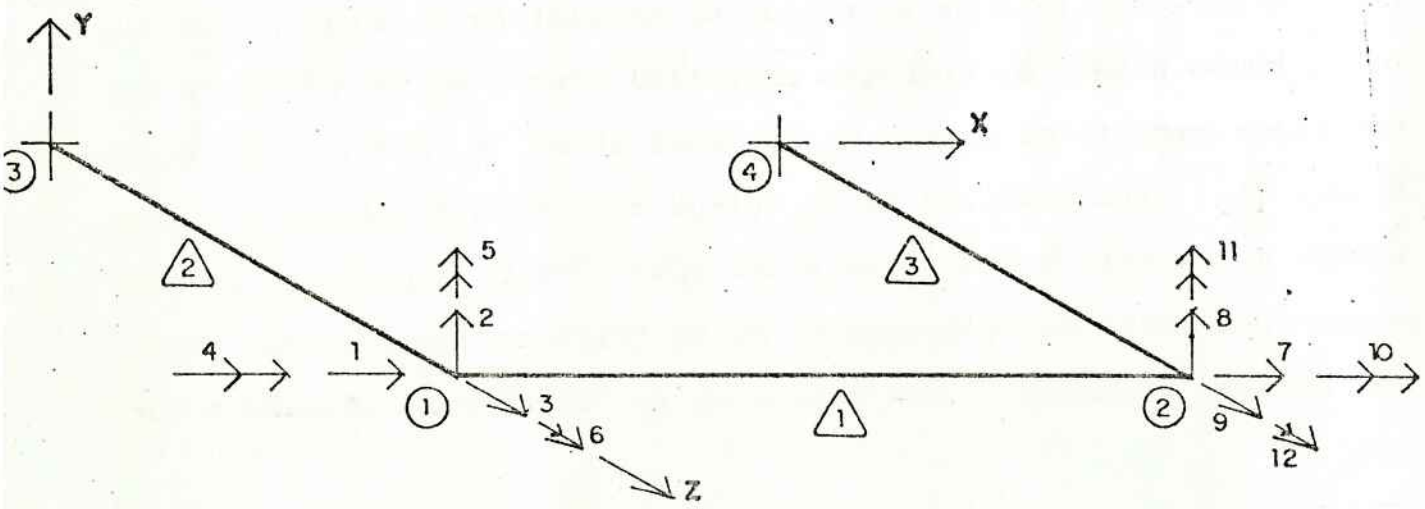


Fig. 4.2 - Graus de Liberdade da Viga Balcão considerada como Pórtico Espacial

24064	0	0	0	-96	0	-24000	0	0	0	0	0
	1351,1111	0	1666,6667	0	600	0	-240	0	0	0	600
		40013,824	0	-34,56	0	0	0	-13,824	0	-34,56	0
			3372,3733	0	0	0	0	0	-39,04	0	0
				307,20	0	0	0	34,56	0	57,6	0
					2065,0666	0	-600	0	0	0	1000
						24064	0	0	0	-96	0
							1351,111	0	1666,6667	0	-600
								40013,824	0	34,56	0
									3372,3733	0	0
										307,2	0
											2065,0666

Fig. 4.3 - Matriz de Rigidez

```
TAMANHO DO ENVELOPE DA MATRIZ ORIGINAL E              52
LARGURA DE BANDA ORIGINAL E                 1Ø
TAMANHO DO ENVELOPE APOS PERMUTACAO E                20
LARGURA DE BANDA APOS PERMUTACAO E           3
SOLUCAO DO SISTEMA
          .ØØØØØØØØØØØØØØØ -1.3Ø4999999997Ø1E-ØØ2          .ØØØØØØØØØØØØØØØ
  6.2999999998251E-ØØ3          .ØØØØØØØØØØØØØØØ -1.877816725Ø888E-ØØ3
          .ØØØØØØØØØØØØØØØ -1.3Ø4999999997Ø1E-ØØ2          .ØØØØØØØØØØØØØØØ
  6.2999999998251E-ØØ3          .ØØØØØØØØØØØØØØØ  1.877816725Ø888E-ØØ3
```

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CRANDALL, S.H. Engineering Analysis. McGraw-Hill, 1956.

[2] TURNER, M.J., CLOUGH, R.W., MARTIN, H.C. & TOPP, L.J. Stiffiness and Deflection Analysis of Complex Structures. Journal of Aeronautical Science, Vol. 23, 1956, pp. 805-823

[3] BATHE, K-J. Finite Element Procedures in Engineering Analysis. Prentice-Hall, 1982.

[4] GEORGE, A. & LIU, J.W-H. Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, 1981.

[5] BRASIL, R.M.L.R.F. Conceituação Matemática e Física do Número de Condição dos Sistemas de Equações de Equilíbrio do Método dos Deslocamentos em Estruturas Reticuladas de Comportamento Linear. Dissertação de Mestrado, PEF-EPUSP, 1984.

APÊNDICE

```
C------PROGRAMA PARENV-VERSAO 1.Ø-25/1Ø/88
C*****************************************************************
C
C      UM ESQUEMA DE OTIMIZACAO DE ARMAZENAGEM E NUMERO DE
C      OPERACOE NA SOLUCAO DE SISTEMAS DE EQUACOES POSITIVODEFINIDAS
C      SIMETRICAS, UTILIZANDO REVERSE CUTHILL-MCKEE E CHOLESKY
C
C      AUTORES:    PRISCILA GOLDENBERG
C                  REYOLANDO BRASIL
C
C*****************************************************************
C
C NEQNS-NUMERO DE EQUACOES LINEARES
C (XADJ,ADJNCY)=PAR DE VETORES CONTENDO A ESTRUTURA DE ADJACENCIAS
C                  DO GRAFO ASSOCIADO A MATRIZ
C (PERM,INVP)=VETOR CONTENDO AS PERMUTACOES PARA A REORDENACAO
C               DA MATRIZ
C XENV=VETOR DE INDICES PARA A ESTRUTURA DE ARMAZENAMENTO A SER
C       UTILIZADA PELO ENVELOPE INFERIOR(SUPERIOR) DA MATRIZ PERMUTADA
C ENV=VETOR CONTENDO O ENVELOPE INFERIOR(SUPERIOR) DA MATRIZ PERMUTADA
C DIAG=VETOR CONTENDO A DIAGONAL DA MATRIZ
C RHS=COMO VETOR DE ENTRADA CONTEM O SEGUNDO MEMBRO DO SISTEMA PERMUTADO
C      COMO VETOR DE SAIDA CONTEM A SOLUCAO DO SISTEMA
C
C*****************************************************************
C
      INTEGER XADJ(2ØØ),ADJNCY(8ØØ),PERM(2ØØ),INVP(2ØØ),MASK(2ØØ)
      INTEGER XENV(2ØØ),ENVSZE,BANDW,CCSIZE,ROOT,XLS(2ØØ),XORENV(2ØØ)
      REAL*8 RHS(2ØØ),DIAG(2ØØ),ENV(2ØØØ),ORIENV(2ØØØ)
      OPEN(5,FILE='PRN')
      WRITE(5,*)'*************************************************'
      WRITE(5,*)'       PROGRAMA PARENV-VERSAO 1.Ø-25/1Ø/88'
      WRITE(5,*)' SOLUCAO DE SISTEMAS POSITIVO-DEFINIDOS SIMETRICOS'
      WRITE(5,*)'COM OTIMIZACAO DE ARMAZENAGEM E NUMERO DE OPERACOES'
      WRITE(5,*)'     AUTORES: PRISCILA GOLDENBERG'
      WRITE(5,*)'                REYOLANDO BRASIL'
      WRITE(5,*)'*************************************************'
      WRITE(*,*)'INDIQUE O NUMERO DE EQUACOES'
      READ(*,*)NEQNS
      NM1=NEQNS+1
      WRITE(*,*)'ENTRADA DO VETOR DE ADJACENCIAS=ADJNCY'
      WRITE(*,*)'E DE SEUS INDICADORES=XADJ. OPCOES:'
      WRITE(*,*)'A) GERA-LOS A PARTIR DO GRAFO ASSOCIADO (DIGITE Ø)'
      WRITE(*,*)'B) FORNECE-LOS DIRETAMENTE              (DIGITE 1)'
      READ(*,*) IADJ
      IF (IADJ .EQ. Ø)
     +THEN
        DO 8Ø I=1,NM1
          XADJ(I)=1
8Ø      CONTINUE
        WRITE(*,*)'ENTRE NUMERO DE ARESTAS DO GRAFO'
        READ(*,*) NAREST
        WRITE(*,*)'PARA CADA ARESTA ENTRE NO INICIAL, NO FINAL'
        DO 9Ø I=1,NAREST
          WRITE(*,*)'ARESTA',I
          READ(*,*) NI,NF
          CALL GERADJ (NM1,NI,NF,XADJ,ADJNCY)
9Ø      CONTINUE
      ELSE
        WRITE(*,*)'ENTRE O VETOR INDICADOR DE ADJACENCIAS=XADJ'
        READ(*,*) (XADJ(I),I=1,NM1)
        M=XADJ(NM1)-1
```

```
      WRITE(*,*)'ENTRE O VETOR DE ADJACENCIAS=ADJNCY'
      READ(*,*)(ADJNCY(I),I=1,M)
    ENDIF
    DO 100 I=1,NEQNS
      PERM(I)=I
      INVP(I)=I
100 CONTINUE
    CALL FNENV(NEQNS,XADJ,ADJNCY,PERM,INVP,XORENV,ENVSZE,BANDW)
    WRITE(5,*)'TAMANHO DO ENVELOPE DA MATRIZ ORIGINAL E ',ENVSZE
    WRITE(5,*)'LARGURA DE BANDA ORIGINAL E ',BANDW
    WRITE(*,*)'INDIQUE O VETOR ENVELOPE DA MATRIZ ORIGINAL'
    READ(*,*)(ORIENV(I),I=1,ENVSZE)
    WRITE(*,*)'INDIQUE A DIAGONAL DO SISTEMA ORIGINAL'
    READ(*,*)(DIAG(I),I=1,NEQNS)
    WRITE(*,*)'INDIQUE O VETOR SEGUNDO MEMBRO DO SISTEMA'
    READ(*,*)(RHS(I),I=1,NEQNS)
    WRITE(*,*)'INDIQUE SE A OTIMIZACAO DEVERA SER EFETUADA'
    WRITE(*,*)'CASO AFIRMATIVO DIGITE ZERO;CASO NEGATIVO DIGITE 1'
    READ(*,*)OPC1
    IF(OPC1)101,102,101
102 CALL GENRCM(NEQNS,XADJ,ADJNCY,PERM,MASK,XLS)
    GO TO 104
101 WRITE(*,*)'INDIQUE O VETOR DE PERMUTACOES'
    READ(*,*)(PERM(I),I=1,NEQNS)
104 DO 103 I=1,NEQNS
103 INVP(PERM(I))=I
    CALL FNENV(NEQNS,XADJ,ADJNCY,PERM,INVP,XENV,ENVSZE,BANDW)
    WRITE(5,*)'TAMANHO DO ENVELOPE APOS PERMUTACAO E',ENVSZE
    WRITE(5,*)'LARGURA DE BANDA APOS PERMUTACAO E',BANDW
    DO 110 I=1,ENVSZE
      ENV(I)=0.0D0
110 CONTINUE
    CALL PAPT(NEQNS,INVP,ORIENV,XORENV,XENV,ENV)
    CALL PERMVT(NEQNS,PERM,DIAG)
    CALL ESFCT(NEQNS,XENV,ENV,DIAG,IFLAG)
105 CALL PERMVT(NEQNS,PERM,RHS)
    CALL ELSLV(NEQNS,XENV,ENV,DIAG,RHS)
    CALL EUSLV(NEQNS,XENV,ENV,DIAG,RHS)
    CALL PERMVT(NEQNS,INVP,RHS)
    WRITE(5,*)'SOLUCAO DO SISTEMA'
    WRITE(5,*)(RHS(I),I=1,NEQNS)
    WRITE(5,*)'**********************************'
    WRITE(*,*)'QUER ENTRAR OUTRO VETOR RHS?(1=NAO,0=SIM)'
    READ(*,*) IRHS
    IF (IRHS) 107,106,107
106 WRITE(*,*)'INDIQUE NOVO VETOR SEGUNDO MEMBRO DO SISTEMA'
    READ(*,*)(RHS(I),I=1,NEQNS)
    GO TO 105
107 CONTINUE
    CLOSE (5)
    END
C-----FNENV---ENCONTRAR  ENVELOPE DA MATRIZ PERMUTADA
    SUBROUTINE FNENV(NEQNS,XADJ,ADJNCY,PERM,INVP,XENV,ENVSZE,BANDW)
    INTEGER ADJNCY(*),INVP(*),XADJ(*),XENV(*),PERM(*)
    INTEGER BANDW,ENVSZE
    BANDW=0
    ENVSZE=1
    DO 200 I=1,NEQNS
    XENV(I)=ENVSZE
    IPERM=PERM(I)
    JSTRT=XADJ(IPERM)
    JSTOP=XADJ(IPERM+1)-1
```

```
      IF(JSTOP.LT.JSTRT)GO TO 200
      IFIRST=I
      DO 100 J=JSTRT,JSTOP
      NABOR=ADJNCY(J)
      NABOR=INVP(NABOR)
      IF(NABOR.LT.IFIRST) IFIRST=NABOR
  100 CONTINUE
      IBAND=I-IFIRST
      ENVSZE=ENVSZE+IBAND
       IF(BANDW.LT.IBAND) BANDW=IBAND
  200 CONTINUE
      XENV(NEQNS+1)=ENVSZE
      ENVSZE=ENVSZE-1
      RETURN
      END
C--ELSLV-RESOLVER SISTEMA TRIANGULAR INFERIOR LX=RHS
C           O FATOR L DEVE SER ARMAZENADO NO FORMATO DE ENVELOPE
      SUBROUTINE ELSLV(NEQNS,XENV,ENV,DIAG,RHS)
      COMMON/SPKOPS/OPS
      REAL*8 DIAG(*),ENV(*),RHS(*),COUNT,OPS,S
      INTEGER XENV(*)
      IFIRST=0
  100 IFIRST=IFIRST+1
      IF(RHS(IFIRST).NE.0.0E0) GO TO 200
      IF(IFIRST.LT.NEQNS) GO TO 100
      RETURN
  200 LAST=0
      DO 500 I=IFIRST,NEQNS
      IBAND=XENV(I+1)-XENV(I)
      IF(IBAND.GE.I) IBAND=I-1
      S=RHS(I)
      L=I-IBAND
      RHS(I)=0.0D0
      IF(IBAND.EQ.0.OR.LAST.LT.L) GO TO 400
      KSTRT=XENV(I+1)-IBAND
      KSTOP=XENV(I+1)-1
      DO 300 K=KSTRT,KSTOP
      S=S-ENV(K)*RHS(L)
      L=L+1
  300 CONTINUE
      COUNT=IBAND
      OPS=OPS+COUNT
  400 IF(S.EQ.0.0D0) GO TO 500
      RHS(I)=S/DIAG(I)
      OPS=OPS+1.0D0
      LAST=I
  500 CONTINUE
      RETURN
      END
C--EUSLV-RESOLVER SISTEMA TRIANGULAR SUPERIOR UX=RHS
C           O FATOR U DEVE SER ARMAZENADO NA FORMA DE ENVELOPE
      SUBROUTINE EUSLV(NEQNS,XENV,ENV,DIAG,RHS)
      COMMON/SPKOPS/OPS
      REAL*8 DIAG(*),ENV(*),RHS(*),COUNT,OPS,S
      INTEGER XENV(*)
      I=NEQNS+1
  100 I=I-1
      IF(I.EQ.0)RETURN
      IF(RHS(I).EQ.0.0D0) GO TO 100
      S=RHS(I)/DIAG(I)
      RHS(I)=S
      OPS=OPS+1.0D0
```

```
      IBAND=XENV(I+1)-XENV(I)
      IF(IBAND.GE.I)IBAND=I-1
      IF(IBAND.EQ.Ø) GO TO 1ØØ
      KSTRT=I-IBAND
      KSTOP=I-1
      L=XENV(I+1)-IBAND
      DO 2ØØ K=KSTRT,KSTOP
      RHS(K)=RHS(K)-S*ENV(L)
      L=L+1
 2ØØ  CONTINUE
      COUNT=IBAND
      OPS=OPS+COUNT
      GO TO 1ØØ
      END
C--ESFCT----FATORAR MATRIZ DEFINIDA POSITIVA EM L*L(TRANSPOSTA).
C  A  MATRIZ DEVE SER ARMAZENADA NA FORMA DE ENVELOPE
C            O  ALGORITMO UTILIZADO FOIDE BORDEAMENTO STANDARD
      SUBROUTINE ESFCT(NEQNS,XENV,ENV,DIAG,IFLAG)
      REAL*8 DIAG(*),ENV(*),COUNT,OPS,TEMP,S
      INTEGER XENV(*)
      IF(DIAG(1).LE.Ø.ØEØ) GO TO 4ØØ
      DIAG(1)=DSQRT(DIAG(1))
      IF(NEQNS.EQ.1)RETURN
      DO 3ØØ I=2,NEQNS
      IXENV=XENV(I)
      IBAND=XENV(I+1)-IXENV
      TEMP=DIAG(I)
      IF(IBAND.EQ.Ø) GO TO 2ØØ
      IFIRST=I-IBAND
      CALL ELSLV(IBAND,XENV(IFIRST),ENV,DIAG(IFIRST),ENV(IXENV))
      JSTOP=XENV(I+1)-1
      DO 1ØØ J=IXENV,JSTOP
      S=ENV(J)
      TEMP=TEMP-S*S
 1ØØ  CONTINUE
 2ØØ  IF(TEMP.LE.Ø.ØDØ)GO TO 4ØØ
      DIAG(I)=DSQRT(TEMP)
      COUNT=IBAND
      OPS=OPS+COUNT
 3ØØ  CONTINUE
      RETURN
 4ØØ  IFLAG=1
      RETURN
      END
C------ROOTLS-----ESTRUTURA ROTULACAO-----------------------------------
      SUBROUTINE ROOTLS(ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,LS)
      INTEGER CCSIZE,ROOT
      INTEGER ADJNCY(*),LS(*),MASK(*),XLS(*),XADJ(*)
      MASK(ROOT)=Ø
      LS(1)=ROOT
      NLVL=Ø
      LVLEND=Ø
      CCSIZE=1
 2ØØ  LBEGIN=LVLEND+1
      LVLEND=CCSIZE
      NLVL=NLVL+1
      XLS(NLVL)=LBEGIN
      DO 4ØØ I=LBEGIN,LVLEND
      NODE=LS(I)
      JSTRT=XADJ(NODE)
      JSTOP=XADJ(NODE+1)-1
      IF(JSTOP.LT.JSTRT) GO TO 4ØØ
```

```
      DO 300 J=JSTRT,JSTOP
      NBR=ADJNCY(J)
      IF(MASK(NBR).EQ.0)GO TO 300
      CCSIZE=CCSIZE+1
      LS(CCSIZE)=NBR
      MASK(NBR)=0
  300 CONTINUE
  400 CONTINUE
      LVSIZE=CCSIZE-LVLEND
      IF(LVSIZE.GT.0) GO TO 200
      XLS(NLVL+1)=LVLEND+1
      DO 500 I=1,CCSIZE
      NODE=LS(I)
      MASK(NODE)=1
  500 CONTINUE
      END
C-----FNROOT--ENCONTRAR NO PSEUDO PERIFERICO-----------------------------
      SUBROUTINE FNROOT(ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,LS)
      INTEGER CCSIZE,ROOT
      INTEGER ADJNCY(*),LS(*),MASK(*),XLS(*),XADJ(*)
      CALL ROOTLS(ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,LS)
      CCSIZE=XLS(NLVL+1)-1
      IF(NLVL.EQ.1.OR.NLVL.EQ.CCSIZE)RETURN
  100 JSTRT=XLS(NLVL)
      MINDEG=CCSIZE
      ROOT=LS(JSTRT)
      IF(CCSIZE.EQ.JSTRT) GO TO 400
      DO 300 J=JSTRT,CCSIZE
      NODE=LS(J)
      NDEG=0
      KSTRT=XADJ(NODE)
      KSTOP=XADJ(NODE+1)-1
      DO 200 K=KSTRT,KSTOP
      NABOR=ADJNCY(K)
      IF(MASK(NABOR).GT.0) NDEG=NDEG+1
  200 CONTINUE
      IF(NDEG.GE.MINDEG) GO TO 300
      ROOT=NODE
      MINDEG=NDEG
  300 CONTINUE
  400 CALL ROOTLS(ROOT,XADJ,ADJNCY,MASK,NUNLVL,XLS,LS)
      IF(NUNLVL.LE.NLVL) RETURN
      NLVL=NUNLVL
      IF(NLVL.LT.CCSIZE) GO TO 100
      RETURN
      END
C-------DEGREE-------DETERMINACAO DE GRAU DOS NOS ESPECIFICADOS
      SUBROUTINE DEGREE(ROOT,XADJ,ADJNCY,MASK,DEG,CCSIZE,LS)
      INTEGER ADJNCY(*),DEG(*),MASK(*),XADJ(*),LS(*)
       INTEGER CCSIZE,ROOT
      LS(1)=ROOT
      XADJ(ROOT)=-XADJ(ROOT)
      LVLEND=0
      CCSIZE=1
  100 LBEGIN=LVLEND+1
      LVLEND=CCSIZE
      DO 400 I=LBEGIN,LVLEND
      NODE=LS(I)
      JSTRT=-XADJ(NODE)
      JSTOP=IABS(XADJ(NODE+1))-1
       IDEG=0
      IF(JSTOP.LT.JSTRT) GO TO 300
```

```
      DO 200 J=JSTRT,JSTOP
      NBR=ADJNCY(J)
      IF(MASK(NBR).EQ.0) GO TO 200
      IDEG=IDEG+1
      IF(XADJ(NBR).LT.0) GO TO 200
      XADJ(NBR)=-XADJ(NBR)
      CCSIZE=CCSIZE+1
      LS(CCSIZE)=NBR
  200 CONTINUE
  300 DEG(NODE)=IDEG
  400 CONTINUE
      LVSIZE=CCSIZE-LVLEND
      IF(LVSIZE.GT.0) GO TO 100
      DO 500 I=1,CCSIZE
      NODE=LS(I)
      XADJ(NODE)=-XADJ(NODE)
  500 CONTINUE
      RETURN
      END
C-----RCM REVERSE CUTHILL-MCKEE ORDENACAO--------------------------------
      SUBROUTINE RCM(ROOT,XADJ,ADJNCY,MASK,PERM,CCSIZE,DEG)
      INTEGER ADJNCY(*),DEG(*),MASK(*),PERM(*),XADJ(*)
      INTEGER FNBR,ROOT,CCSIZE
      CALL DEGREE(ROOT,XADJ,ADJNCY,MASK,DEG,CCSIZE,PERM)
      MASK(ROOT)=0
      IF(CCSIZE.LE.1)RETURN
      LVLEND=0
      LNBR=1
  100 LBEGIN=LVLEND+1
      LVLEND=LNBR
      DO 600 I=LBEGIN,LVLEND
      NODE=PERM(I)
      JSTRT=XADJ(NODE)
      JSTOP=XADJ(NODE+1)-1
      FNBR=LNBR+1
      DO 200 J=JSTRT,JSTOP
      NBR=ADJNCY(J)
      IF(MASK(NBR).EQ.0) GO TO 200
      LNBR=LNBR+1
      MASK(NBR)=0
      PERM(LNBR)=NBR
  200 CONTINUE
      IF(FNBR.GE.LNBR) GO TO 600
      K=FNBR
  300 L=K
      K=K+1
      NBR=PERM(K)
  400 IF(L.LT.FNBR) GO TO 500
      LPERM=PERM(L)
      IF(DEG(LPERM).LE.DEG(NBR)) GO TO 500
      PERM(L+1)=LPERM
      L=L-1
      GO TO 400
  500 PERM(L+1)=NBR
      IF(K.LT.LNBR) GO TO 300
  600 CONTINUE
      IF(LNBR.GT.LVLEND) GO TO 100
      K=CCSIZE/2
      L=CCSIZE
      DO 700 I=1,K
      LPERM=PERM(L)
      PERM(L)=PERM(I)
```

```
      PERM(I)=LPERM
      L=L-1
  700 CONTINUE
      RETURN
      END
C-----GENRCM---ALGORITMO GERAL CUTHILL MCKEE REVERSO
      SUBROUTINE GENRCM(NEQNS,XADJ,ADJNCY,PERM,MASK,XLS)
      INTEGER ADJNCY(*),XADJ(*),PERM(*),XLS(*),MASK(*),
     *ROOT,CCSIZE
      DO 100 I=1,NEQNS
      MASK(I)=1
  100 CONTINUE
      NUM=1
      DO 200 I=1,NEQNS
      IF(MASK(I).EQ.0) GO TO 200
      ROOT=I
      CALL FNROOT(ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,PERM(NUM))
      CALL RCM(ROOT,XADJ,ADJNCY,MASK,PERM(NUM),CCSIZE,XLS)
      NUM=NUM+CCSIZE
      IF(NUM.GT.NEQNS) RETURN
  200 CONTINUE
      RETURN
      END
C-----PAPT---SUBROTINA PARA EFETUAR AS PERMUTACOES NO ENVELOPE
C-----DA MATRIZ  A  ORIGINAL
      SUBROUTINE PAPT(NEQNS,INVP,ORIENV,XORENV,XENV,ENV)
      REAL*8 ENV(*),ORIENV(*)
      INTEGER INVP(*),XORENV(*),XENV(*)
      DO 20 IANT=1,NEQNS
        JI=IANT-(XORENV(IANT+1)-XORENV(IANT))
        JF=IANT-1
        IF (JF .GE. JI)
     +  THEN
          NTEND=XORENV(IANT)-1
          DO 10 JANT=JI,JF
            NTEND=NTEND+1
            INOVO=INVP(IANT)
            JNOVO=INVP(JANT)
            IF (JNOVO .GT. INOVO)
     +      THEN
              IAUX=INOVO
              INOVO=JNOVO
              JNOVO=IAUX
            ENDIF
            IF (ORIENV(NTEND) .NE. 0.0)
     +      THEN
              NOVEND=XENV(INOVO+1)-INOVO+JNOVO
              ENV(NOVEND)=ORIENV(NTEND)
            ENDIF
   10     CONTINUE
        ENDIF
   20 CONTINUE
      RETURN
      END
C-----PERMVT----SUBROTINA PARA PERMUTACAO DE VETOR (DIAG OU RHS)
      SUBROUTINE PERMVT(NEQNS,PERM,VT)
      REAL*8 VT(*)
      INTEGER PERM(*)
      J=1
      VT1=VT(1)
   10 CONTINUE
        VT(J)=VT(PERM(J))
```

```
          VT(PERM(J))=VT1
          J=PERM(J)
        IF ( .NOT. PERM(J) .EQ. 1) GOTO 1Ø
        RETURN
        END
C----GERADJ----SUBROTINA PARA GERACAO DO VETOR DE ADJACENCIAS
C----(ADJNCY) E DO VETOR DE INDICADORES (XADJ)
        SUBROUTINE GERADJ(NM1,NI,NF,XADJ,ADJNCY)
        INTEGER XADJ(*),ADJNCY(*)
        DO 3Ø J=1,2
           IF (J .EQ. 2)
       +   THEN
             AUX=NI
             NI=NF
             NF=AUX
           ENDIF
           KI=XADJ(NM1)-1
           KF=XADJ(NF)
           IF (KI .NE. Ø)
       +   THEN
             DO 1Ø K=KI,KF,-1
               ADJNCY(K+1)=ADJNCY(K)
 1Ø          CONTINUE
           ENDIF
           ADJNCY(XADJ(NF))=NI
           DO 2Ø K=NF+1,NM1
             XADJ(K)=XADJ(K)+1
 2Ø        CONTINUE
 3Ø      CONTINUE
         RETURN
         END
```