Armazenamento de uma matriz esparsa gerada pelo método dos elementos finitos

December 12, 2001

Abstract

Várias técnicas tem sido definidas com o objetivo principal de armazenar somente os elementos diferentes de zero. Neste texto é feito uma comparação entre técnicas que usam estrutura de dados diferentes, visando explorar a otimalidade com relação a memória utilizada. Considera-se uma matriz esparsa A de ordem n com um número total de elementos diferentes de zero igual a N_z gerada pelo método dos elementos finitos

1 Estrutura de dados usando o formato CSR (*Compressed Sparse Row*)

Na estrutura CSR são necessários 3 vetores, sendo:

- Um vetor real AA, contendo os valores reais $a_{ij} \neq 0$, armazenados linha por linha, da linha 1 até a linha n. AA tem nnz elementos.
- Um vetor de inteiros JA, contendo o índice da coluna de cada elemento a_{ij} armazenado em AA, com nnz elementos.
- um vetor IA de inteiros, contendo os ponteiros onde começam cada linha em AA e JA. Ou seja, o elemento $IA(i)$ contém a posição de AA e JA onde a i-ésima linha começa. O tamanho de IA é $n+1$, com $IA(n+1)$ contendo o número $IA(1) + nnz$, isto é, o endereço em AA e JA do começo de um número de linha fictício $n+1$.

A Figura 1 representa uma malha de elementos triangulares, considerando apenas uma variável em cada nó. A matriz esparsa correspondente possui 41 coeficientes diferentes de zero:

$$A = \begin{bmatrix} \otimes & \times & & \times & \times & & & & \\ \times & \otimes & \times & & \times & \times & & & \\ & \times & \otimes & & & \times & & & \\ \times & & & \otimes & \times & & \times & \times & \\ \times & \times & & \times & \otimes & \times & & \times & \times \\ & \times & \times & & \times & \otimes & & & \times \\ & & & \times & & & \otimes & \times & \\ & & & \times & \times & & \times & \otimes & \times \\ & & & & \times & \times & & \times & \otimes \end{bmatrix}$$

O armazenamento do tipo CSR para esse exemplo pode ser representado por:

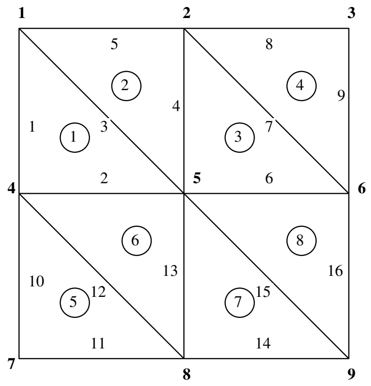

Figura 1: Malha de elementos triangulares lineares com 8 elementos, 9 nós e 16 arestas

$$
\begin{array}{lccccccccccccc}
AA = [& \overbrace{\otimes}^{1} & \times & \times & \times & \overbrace{\times}^{5} & \otimes & \times & \times & \ldots & \overbrace{\times}^{38} & \times & \times & \otimes] \\
JA = [& 1 & 2 & 4 & 5 & 1 & 2 & 3 & 5 & \ldots & 5 & 6 & 8 & 9] \\
IA = [& 1 & 5\ldots & 38 & 42] & & & & & & & & &
\end{array}
$$

Para este caso são necessárias 92 posições de memória.

2 Estrutura de Dados usando o Formato MSR (*Modified Sparse Row*)

Uma variação comum do esquema CSR explora o fato da diagonal principal de A ser usualmente de elementos diferentes de zero e ser acessada com maior frequência que os demais elementos. O formato MSR utiliza somente 2 vetores:

- Um vetor AA de reais, contendo os elementos não nulos de A. A diagonal de A ocupa as n primeiras posições de AA. A posição $(n+1)$ do vetor AA não é usada para coeficientes de A, mas pode ser usada para alocar alguma informação adicional de A. A partir da posição $(n+2)$ os coeficientes não nulos são armazenados por linha.
- Um vetor de inteiros JA, contendo nas $(n+1)$ primeiras posições os ponteiros de começo de cada linha de AA. Nas posições restantes estão as colunas correspondentes de cada elemento em AA, ou seja, para cada elemento $AA(k)$, o inteiro $JA(k)$ representa o índice da coluna na matriz A. No esquema a seguir encontra-se o armazenamento do tipo MSR para a malha ilustrada na Figura 4.2:

$$
\begin{array}{lcccccccccccc}
AA = [& \otimes & \otimes & \ldots & \otimes & * & \overbrace{\times}^{11} & \times & \times & \overbrace{\times}^{14} & \ldots & \overbrace{\times}^{40} & \times & \times] \\
JA = [& 11 & 14 & \ldots & 40 & 42 & 2 & 4 & 5 & 1 & \ldots & 5 & 6 & 8]
\end{array}
$$

Neste exemplo são necessárias 84 posições de memória.

2.0.1 Estrutura de Dados Elemento-por-Elemento

Neste caso os coeficientes da matriz esparsa A são armazenados a nível de cada elemento. Considerando elementos triangulares lineares, tem-se em cada elemento uma matriz de ordem 3, ou seja, 9 coeficientes. Para uma malha de elementos finitos com nel elementos todos os coeficientes não nulos da matriz esparsa A podem ser armazenados na matriz de reais AA, com 9 colunas e nel linhas. Além disso, é necessário um arranjo de inteiros, a matriz de localização $lm(nel, 3)$, que contem o número das equações globais associados a cada nó do elemento. Em geral a diagonal de A é acessada com frequência, sendo guardada separadamente em um vetor $diagA$ de ordem n. Para o exemplo de malha ilustrado na Figura 1, tem-se o seguinte esquema de armazenamento:

$$AA = \begin{bmatrix} \times & \times & \times & \dots & \dots & \times \\ \times & \times & \times & \dots & \dots & \times \\ \times & \times & \times & \dots & \dots & \times \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \times & \times & \times & \dots & \dots & \times \end{bmatrix}, \text{ com 8 linhas e 9 colunas}$$

$$lm = \begin{bmatrix} 1 & 4 & 5 \\ 1 & 5 & 2 \\ 2 & 5 & 6 \\ \vdots & \vdots & \vdots \\ 5 & 9 & 6 \end{bmatrix}, \text{ com 8 linhas e 3 colunas}$$

$$diagA = [\otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes]$$

São necessárias 72 posições de memória para AA, 24 posições de memória para lm e 9 posições para $diagA$, totalizando 105 posições de memória. Observa-se que neste caso são armazenadas informações redundantes, já que cada coeficiente da matriz esparsa global A recebe contribuição dos elementos adjacentes. Além disso, a diagonal principal é armazenada de forma compacta e por elemento. Para aliviar essa redundância pode-se armazenar a diagonal de forma compacta e as matrizes de elemento sem os termos da diagonal. Portanto, neste caso, o arranjo AA ficaria com 8 linhas e 6 colunas resultando em 81 posições de memória.

3 Estrutura de Dados Aresta-por-Aresta

Os coeficientes da matriz esparsa A são agora armazenados a nível de cada aresta da malha. Tem-se em cada aresta uma matriz de ordem 2, com 4 coeficientes. Para uma malha com $nedges$ arestas os coeficientes de A são armazenados na matriz de inteiros AA com 4 colunas e $nedges$ linhas. Da mesma forma que no esquema por elemento, define-se uma matriz de localização para as arestas, $lm(nedges, 2)$, contendo o número das equações globais associadas aos nós das arestas. Além disso é necessário um vetor de n posições para armazenar $diagA$. Para o exemplo de malha ilustrado na Figura 1, tem-se o esquema de armazenamento:

$$AA = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \vdots & \vdots & \vdots & \vdots \\ \times & \times & \times & \times \end{bmatrix}, \text{ com 16 linhas e 4 colunas}$$

$$lm = \begin{bmatrix} 1 & 4 \\ 4 & 5 \\ 5 & 1 \\ \vdots & \vdots \\ 9 & 6 \end{bmatrix}, \text{ com 16 linhas e 2 colunas}$$

$$diagA = [\otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes]$$

São necessárias 64 posições de memória para AA, 32 posições de memória para lm e 9 posições para $diagA$, totalizando 105 posições de memória. Da mesma forma que no esquema por elemento pode-se armazenar somente os coeficientes fora da diagonal das matrizes das arestas, evitando-se assim redundância de informação no armazenamento da diagonal. Portanto, para esse exemplo o arranjo AA ficaria com 16 linhas e 2 colunas, resultando em 73 posições de memória. É importante ressaltar que os termos de fora da diagonal das matrizes das arestas correspondem aos coeficientes da matriz global A. Dessa forma, pode-se dizer que o esquema de armazenamento por aresta usa os mesmos arranjos do esquema elemento-por-elemento, porém para um elemento de 2 nós, isto é, uma aresta.

Para esse exemplo simples o esquema de armazenamento por aresta é o mais econômico. Em malhas de grande porte, onde o número de arestas é aproximadamente de 2 a 3 vezes maior que o número de triângulos, observa-se, ainda assim, que o esquema por aresta é mais vantajoso.