Universidade Federal do Espírito Santo
DI
2º Trabalho de Algoritmos Numéricos II - 16/1

Estudo Sobre a Influência do Reordenamento e Precondicionamento aplicados a Sistemas Esparsos de Grande Porte Utilizando Métodos Iterativos Não Estacionários

Data de entrega: 19 de junho de 2016

1 Introdução

Frequentemente os processos de solução de problemas das mais diversas áreas do conhecimento recaem na necessidade de resolver sistemas lineares. Na maioria das vezes esses sistemas são esparsos e de grande porte. Nesse contexto, o uso de métodos diretos torna-se desaconselhável, por limitações de tempo e memória. Em detrimento, as abordagens iterativas tornam-se mais atraentes.

Como estudado, os métodos iterativos são divididos em duas classes: estacionários, como Jacobi, Seidel e SOR, e não estacionários, como Gradientes Conjugados, GMRES e LCD. Os métodos iterativos não estacionários são formas mais modernas que utilizam algum tipo de processo de projeção. Ou seja, busca-se uma aproximação da solução do sistema linear a partir de um subespaço. Tais métodos também são conhecidos como métodos livres de matrizes e são mais indicados para trabalhar com os tipos de sistemas já mencionados, pois apresentam facilidades no armazenamento de matrizes esparsas.

No processo de resolução de sistemas esparsos de grande porte, a utilização de técnicas de armazenamento é indispensável. Há um número considerável de formatos de armazenamento que podem utilizados, mas a maioria deles emprega a mesma técnica base. Isto é, armazenar todos os elementos não nulos da matriz em uma estrutura que permite o mapeamento das localizações dos elementos não zero na matriz original. Técnicas de compactação como o *Compressed Sparse Row* (CSR) são largamente utilizadas nessas situações.

Embora os métodos iterativos não estacionários tenham sido bem fundamentados teoricamente, todos eles são propensos a sofrer de convergência lenta para os problemas que surgem a partir de aplicações típicas. O sucesso da implementação desses métodos depende do uso de bons precondicionadores, uma vez que estes melhoram o condicionamento da matriz e aceleram a convergência.

O tempo de execução dos métodos iterativos não estacionários aplicados a matrizes esparsas pode ser significativamente reduzido quando efetuadas simples trocas de linhas e colunas sobre essas matrizes. Esse processo, chamado de reordenamento, visa reduzir o número de operações com ponto flutuante durante a montagem das matrizes precondicionadoras. Existe uma série de algoritmos que realizam tal reordenamento: *Sloan*, *Reverse Cuthill Mckee* (RCM), Espectral, *Nested Dissection* (ND) e *Approximate Minimum Degree* (AMD).

2 Resolução de Sistemas Esparsos de Grande Porte

Seja $Ax = b$ um sistema linear, onde A é uma matriz esparsa de grande porte. Para resolver tal sistema de forma eficiente utilizando métodos iterativos não estacionários faz-se necessário armazenar a matriz A compactada, efetuar o reordenamento de linhas e colunas adequado além de aplicar o precondicionamento.

2.1 Técnica de Compactação

Quando a matriz A dos coeficientes é esparsa, o sistema linear correspondente pode ser resolvido de forma mais eficiente se os elementos nulos não forem armazenados. Uma série de esquemas de armazenamento de matrizes esparsas foram idealizados para alocar memória de forma contígua com o objetivo de armazenar os elementos não nulos da matriz e, talvez, um número limitado de zeros. Isto, naturalmente, requer uma maneira de saber onde os elementos se encaixam no interior da matriz original. Dentre esses esquemas, podem ser citados o *Compressed Sparse Row* (CSR), *Compressed Column Storage* (CCS), armazenamento Diagonal e o armazenamento Skyline.

A técnica de compactação CSR será utilizada no presente trabalho. Ela substitui a matriz A, por três vetores auxiliares AA, JA e IA. O vetor AA armazena todas as contribuições não nulas da matriz A, o vetor JA armazena a coluna correspondente que cada coeficiente não nulo ocuparia em A e o vetor IA mapea em AA o primeiro elemento não nulo de cada linha de A.

2.2 Reordenamento

Como já mencionado, o reordenamento está baseado simplesmente nas trocas de linhas e colunas da matriz esparsa originada pelo sistema linear. Tais abordagens visam reduzir o preenchimento de posições originalmente nulas por elementos não nulos durante o processo de precondicionamento.

A criação de cada um desses elementos não nulos é conhecida como *fill-in*. Quanto maior o número de *fill-ins*, pior será o desempenho das técnicas de compactação de matrizes, como o CSR por exemplo, uma vez que a estimativa de valores não nulos armazenados na matriz original tende a crescer significativamente.

Algoritmos como o ND ou AMD visam simplesmente reduzir o *fill-in*, enquanto que RCM, *Sloan* e Espectral tentam aproximar os elementos não-nulos da diagonal principal, o que consequentemente também causa redução do *fill-in*.

Para saber quão próximo os elementos não nulos estão da diagonal principal foram estabelecidas duas métricas: a *largura de banda* e o *envelope*. A *largura de banda* pode ser definida como a maior distância entre o primeiro elemento não-nulo da linha i até a diagonal principal. Já o *envelope* como a soma das distâncias entre o primeiro elemento não-nulo de cada linha i até a diagonal principal. A Fig. 1 apresenta o exemplo de uma matriz esparsa de ordem 10 com *largura de banda* igual a 7 e *envelope* igual a 27.

No presente trabalho será utilizado apenas o algoritmo RCM, visto que o mesmo passou a ser uma das heurísticas mais utilizadas para o problema de minimização da *largura de banda* e do *envelope* devido a boa qualidade de solução, o baixo tempo de execução e a facilidade de implementação.

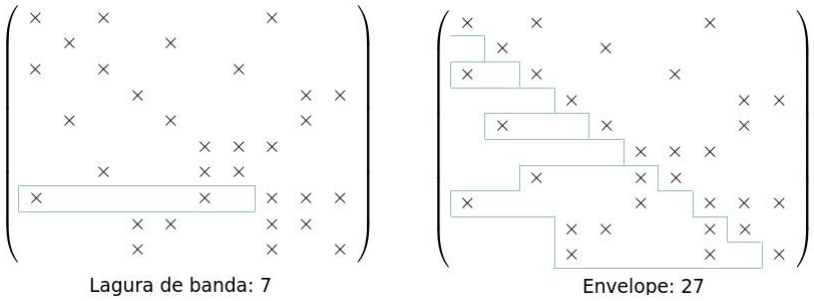

Figure 1: *Largura de banda* e *envelope* de uma matriz exemplo

2.3 Precondicionamento

Precondicionar significa transformar o sistema linear original $Ax = b$ em um sistema de solução equivalente que pode ser resolvido com uma menor quantidade de iterações. O primeiro passo ao precondicionar um sistema linear é encontrar uma matriz de precondicionamento M que satisfaça alguns requerimentos mínimos, como por exemplo, melhorar as propriedades espectrais da matriz A.

A ideia básica é substituir o sistema $Ax = b$ pelo sistema $M^{-1}Ax = M^{-1}b$. Na prática, em lugar de determinar M^{-1} explicitamente, a ação do precondicionador é determinada na operação do produto matriz-vetor $p = M^{-1}Av$, como segue: primeiro, computa-se $z = Av$ e então a ação de M^{-1} em $p = M^{-1}z$ pela resolução do sistema linear $Mp = z$. De um ponto de vista prático, resolver o sistema precondicionado $Mp = z$ é mais eficiente tanto no consumo de memória quanto na quantidade de operações aritmética envolvidas.

Existem inúmeras possibilidades de escolhas da matriz de precondicionamento. No presente trabalho será utilizada apenas uma forma para M:

$$M \quad = \quad \tilde{L}\tilde{U} \text{ sendo } \tilde{L} \text{ e } \tilde{U} \text{ fatores da decomposição } ILU(p) \tag{1}$$

A fatoração incompleta ILU da matriz original A exige uma decomposição da forma $A = LU - R$ onde L e U têm a mesma estrutura de elementos não nulos como as partes inferior e superior de A, respectivamente, e R é o resíduo ou erro da fatoração. Tal fatoração incompleta, conhecida como $ILU(0)$, é de preferência facilmente computável. Em geral, as fatorações $ILU(p)$ foram desenvolvidas considerando um maior número de preenchimentos em L e U, ou seja, um maior número de coeficientes não nulos são introduzidos durante o processo de eliminação em posições nas quais inicialmente eram nulas.

3 Objetivo

Realizar um estudo sobre a utilização de métodos iterativos não estacionários no contexto de matrizes esparsas de grande porte. Verificar a influência do ordenamento e o precondicionamento aplicado à essas matrizes.

4 Experimentos Numéricos

Para alcançar o objetivo do presente trabalho deve ser feito um relatório detalhado sobre alguns experimentos numéricos. Siga exatamente os passos descritos a seguir para realizar os experimentos.

4.1 Matrizes para Experimentos

Utilize as matrizes esparsas descritas na Tab.1. O formato indicado do arquivo para todas as matrizes é `<nome da matriz>.mtx`.

Matriz	Área de Aplicação	n	nnz
rail_5177	Transferência de Calor	5177	35185
aft01	Problema de Acústica	8205	125567
FEM_3D_thermal1	Problema Térmico	17880	430740
Dubcova2	Problema 2D/3D	65025	1030225

Table 1: Matrizes a serem utilizadas nos experimentos numéricos

Importante: As matrizes listadas na Tab. 1 encontram-se armazenadas no arquivo `matrizes.tgz`.

4.2 Leitura das Matrizes

Utilize a função `MATRIX_readCSR` (função definida dentro do arquivo `matrix.c`) para ler os arquivos .mtx e carregar as informações nos vetores AA, JA e IA correspondentes ao armazenamento CSR de cada matriz definida na Tab. 1. A Fig. 2 apresenta todos os campos que serão utilizados no formato CSR. Note que além dos vetores convencionais AA, JA e IA, a estrutura nomeada tipo MAT, ainda armazena separadamente os elementos da diagonal no vetor de double D e inteiros m, n, e nz.

```
typedef struct
{
        double*     AA;
        double*      D;
        int*        JA;
        int*        IA;
        int     m,n,nz;
} MAT;
```

Figure 2: Estrutura de dados MAT: contendo os vetores do CSR

Observe que os coeficientes não nulos estão listados coluna a coluna e não existe informações sobre os coeficientes do vetor de termos independentes b. Nesse trabalho será considerada a matriz transposta (A^t) como a matriz dos coeficientes do sistema

$$A^t x = b \tag{2}$$

Considerando a solução do sistema conhecida, ou seja, tomando $x = (1, 1, ..., 1)^t$, o vetor b será obtido da seguinte forma:

$$b = A^t x \implies b_i = \sum_{i=1}^{n} a_{ij}. \tag{3}$$

Sendo A^t a transposta da matriz oriunda do arquivo .mtx.

4.3 Reordenamento Através do RCM

A permutação responsável pelas trocas de linhas e colunas propostas pelo algoritmo RCM serão obtidas pela função `REORDERING_RCM_opt` (função definida dentro do arquivo `rcm.c`). Essa função definirá um vetor de permutações que deverá ser aplicado a função de permutação `MATRIX_permutation`, também definida no arquivo `matrix.c`, a qual determinará os vetores AA, JA, IA, agora permutados.

4.4 Obtenção da Matriz de Precondicionamento

Para obter a matriz de precondicionamento M será utilizada a função ILUP (definida dentro do arquivo `ilup.c`). Ela produzirá os fatores L e U, em um formato alternativo, os quais serão armazenados em uma estrutura do tipo SparILU, conforme descrito na Fig. 3.

```
typedef struct
{
        int           n;
        int*     nzcount;  /* length of each row                          */
        int**         ja;  /* pointer-to-pointer to store column indices  */
        double**      ma;  /* pointer-to-pointer to store nonzero entries */
} SparMAT;
```

Figure 3: Estrutura de dados SparILU: contem os fatores L e U

A função ILU, conforme mencionada, utilizará como entrada os dados da matriz original armazenados em um outro formato alternativo denominado SparMAT (conforme decrito na Fig. 4). Esse formato pode ser obtido utilizando a função `CSRto_SPARMAT`, definida no arquivo `ilup.c`.

```
typedef struct
{
        int           n;
        int*     nzcount;  /* length of each row                          */
        int**         ja;  /* pointer-to-pointer to store column indices  */
        double**      ma;  /* pointer-to-pointer to store nonzero entries */
} SparMAT;
```

Figure 4: SparMAT: forma alternativa de armazenar os dados da matriz A

Além dos dados de entrada informado no parágrafo anterior, a função ILUP também precisa saber qual o nível de *fill-in*, p que será considerado.

4.5 Resolução dos Sistemas Lineares

Uma vez realizadas as etapas da leitura de dados, ordenamento e a obtenção da matriz de precondicionamento, o próximo passo é obter a solução do sistema linear permutada. Nesse passo, a função que solucionará o sistema receberá como parâmetros de entrada a matriz dos coeficientes no formato CSR já permutada, a matriz de precondicionamento, o vetor de termos independentes, a tolerância, o número máximo de vetores na base de Krylov e por fim o número máximo de iterações (caso o vetor de termos independentes não esteja permutado, é necessário enviar também o vetor de permutações). Os parâmetros de saída serão a solução do

sistema linear permutada e o número total de iterações realizadas pelo GMRES.
Importante: Você deverá implementar a função `gmres.c` bem como as operações de produto matriz-vetor em CSR e as ações de precondicionamento.

4.6 Desfazer o Ordenamento da Solução do Sistema Linear

Como relatado anteriormente, se os dados da matriz dos coeficientes e termo independente foram reordenados, a solução do sistema linear encontra-se numa ordem diferente da original. Dessa forma, é necessário aplicar a permutação obtida no passo do ordenamento RCM para desfazer o efeito dessa operação sobre o vetor de solução.
Importante: Nos experimentos que serão realizados, esta etapa pode parecer desnecessária, pois o vetor de solução terá todas posições iguais a 1. Em situações gerais, a solução não estará na ordem correta.

4.7 Experimentos

Para cada uma das matrizes listadas na Tab. 1, resolva o sistema linear associado utilizando o método GMRES sem precondicionamento e com precondicionamento ILU(p), com e sem ordenamento RCM. Varie a quantidade de vetores na base de Krylov com valores 20, 50 e 100 (parâmetro k) e o número de níveis *fill-in* (parâmetro p) igual a 0 e igual a um valor não nulo de sua escolha. Anote em uma tabela o número de iterações e o tempo de processamento, em segundos, de cada conjunto de parâmetros. A Tab. 2 ilustra como apresentar os dados da matriz rail_5177, quando escolhido o método GMRES com ordenamento RCM. Lembre-se que para cada matriz utilizada, haverá outra tabela associada a matriz dada. Essa segunda referente a utilização do método GMRES sem considerar o ordenamento RCM. Como a Tab. 1 apresenta quatro matrizes, o relatório conterá 8 tabelas.

	GMRES (s/ Precond.)		**GMRES + ILU(0)**		**GMRES + ILU(p>0)**	
k	**Iterações**	**Tempo**	**Iterações**	**Tempo**	**Iter.**	**Tempo**
20						
50						
100						

Table 2: Matriz rail_5177 - Método GMRES com ordenamento RCM

Para uma melhor compreensão do estudo da convergência, escolha para cada uma das matrizes apresentadas na Tab. 1 o valor de k e p que resultaram em um menor tempo de processamento utilizando o método GMRES. Para cada configuração escolhida, faça um gráfico do número de iterações em função da logaritmo do resíduo da solução (iter $\times \log(||r||)$). O gráfico deverá conter uma comparação entre o método GMRES precondicionado com o método sem precondicionamento. Ao todo serão construídos 04 gráficos, sendo um para cada matriz, ou seja, o melhor caso do método GMRES para cada matriz. A Fig. 5 ilustra o gráfico do resíduo da solução em função do número de iterações comparando o método GMRES com precondicionador $ILU(0)$, $ILUP(10)$ e sem precondicionador.

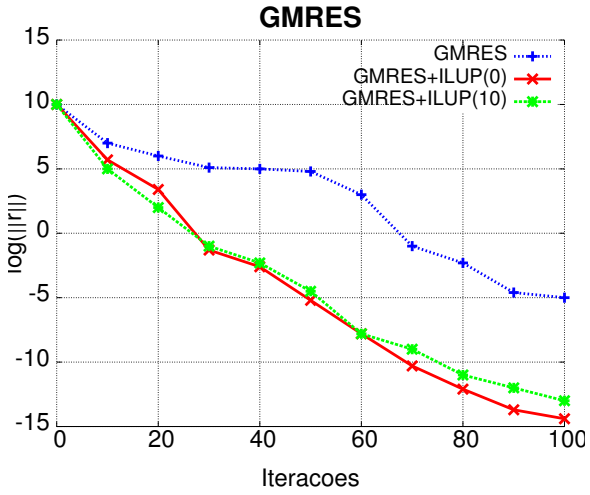

Figure 5: Matriz rail_5177 - Estudo da convergência do método GMRES

5 Estrutura do relatório

O relatório deve ser escrito observando as normas do padrão ABNT e salvo em pdf. A divisão do relatório deve ser de acordo com as seguintes seções:

- **Introdução:** apresentar a estrutura do trabalho e os objetivos
- **Referencial Teórico** um pequeno resumo considerando todos os métodos e técnicas abordados.
- **Experimentos Numéricos:** onde serão apresentadas as 8 tabelas, os 4 gráficos bem como seus respectivos comentários. É imperativo descrever nessa seção qual foi o *software* e *hardware* utilizado.
- **Conclusão:** onde serão discutidos os resultados obtidos.

6 Considerações gerais sobre o trabalho

- O relatório deve ser enviado por e-mail para luciac@inf.ufes.br até o dia 19/06/2016.
- O assunto do e-mail deve ser AN2161:TRAB2:<nome1><nome2>, contendo em anexo, um arquivo do tipo TRAB2<nome1><nome2>.zip. Neste caso <nome1><nome2> deve conter o nome e último sobrenome (por exemplo, AN2161:TRAB2:LuciaCatabrigaRamoniSerrano)
- Qualquer código que você tenha implementado deve ser anexado ao arquivo .zip descrito no item anterior.
- Caso o arquivo seja enviado múltiplas vezes, apenas a última versão enviada será considerada.