Universidade Federal do Espírito Santo
Departamento de Informática
1º Trabalho de Algoritmos Numéricos I - 2015/2

Resolução de Sistemas Lineares Esparsos

Data de entrega: 04 de outubro de 2015

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. Sendo assim, faz-se necessário estabelecer métodos que sejam eficazes para resolver tais sistemas. Diante desse fato, vale lembrar que os métodos de solução de sistemas lineares são divididos em dois grandes grupos: os métodos diretos e os métodos iterativos.

Os métodos diretos calculam a solução de um sistema em um número finito de passos. Estes métodos resultariam na resposta precisa se eles fossem realizados com precisão infinita. Exemplos incluem a eliminação de Gauss, a decomposição LU, a fatoração Cholesky, entre outros. Na prática, é utilizada precisão finita e o resultado é uma aproximação da solução real.

Em contraste aos métodos diretos, os métodos iterativos não terminam em um determinado número de passos. Atribuído um valor inicial, eles realizam sucessivas aproximações que convergem para a solução exata em seu limite. Um teste de convergência é especificado para decidir quando uma solução suficientemente precisa foi encontrada. Como exemplo dessa classe podem ser citados os métodos de Jacobi, de Gauss-Seidel e SOR (*Sucessive Over Relaxation*).

No processo de resolução de sistemas esparsos de grande porte é importante ressaltar que a utilização de técnicas de armazenamento é indispensável, visto que há um número considerável de elementos nulos que não precisariam ser calculados. Há uma gama de técnicas que podem ser empregadas, mas a maioria delas utiliza o mesmo princípio 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), *Compressed Sparse Column* (CSC) ou *Skyline* são largamente utilizadas nessas situações.

2 Objetivo do trabalho

Implementar em linguagem C o algoritmo de eliminação de Gauss e o algoritmo SOR, ambos utilizando técnicas de armazenamento de matrizes esparsas. Ou seja, o objetivo é armazenar apenas os elementos indispensáveis para resolução do sistema linear, desprezando assim, uma infinidade de coeficientes iguais a zero.

3 Experimentos Sugeridos

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.

3.1 Implementação dos métodos

Implemente em linguagem C o algoritmo de eliminação de Gauss e o algoritmo SOR. Ambos devem utilizar estratégias de armazenamento de matrizes esparsas. Organize seu código de modo que o mesmo seja legível e modularizado. Escreva cada função em um arquivo diferente. Seu programa deve ter no mínimo quatro módulos com as seguintes funcionalidades:

- Entrada de dados;
- Eliminação de Gauss;
- SOR;
- Saída de dados.

Seu algoritmo deve ser compilado através de um Makefile que utiliza o compilador gcc. Quanto a execução em si, é necessário que o programa obedeça a forma do exemplo:

```
./programa gauss exemplo
./programa sor 1e-5 1.5 exemplo
```

o que significa que o executável chamado `programa` necessitará de alguns argumentos. No primeiro caso, o tipo de algorimo (`gauss`) e o nome da matriz de teste (`exemplo`). Já no segundo caso, quando o algortimo for o `sor`, além desse parâmetro, serão necessários mais três, ou seja, a tolerância, o parâmetro de relaxação ω e o nome da matriz de teste.

Os algoritmos implementados devem garantir que a solução do sistema linear seja escrita em um arquivo de saída conforme o exemplo a seguir:

- Sistema linear exemplo:

$$\begin{bmatrix} 5.0 & 0.0 & -1.0 & 0.0 \\ 1.0 & 3.0 & 0.0 & 2.0 \\ 0.0 & 0.0 & 4.0 & 1.0 \\ 1.0 & 0.0 & 0.0 & 6.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 17.0 \\ 2.0 \\ -7.0 \\ 9.0 \end{bmatrix}$$

- Arquivo de entrada `exemplo.txt`:

```
4 4 8
1 1  5.0
2 1  1.0
4 1  1.0
2 2  3.0
1 3 -1.0
3 3  4.0
2 4  2.0
3 4  1.0
17.0
2.0
-7.0
9.0
```

- Execução:

```
./programa gauss exemplo
```

- Arquivo de saída `saida_exemplo.txt`:

```
4
3.0
-1.0
-2.0
1.0
```

3.2 Matrizes para experimentos

Utilize as matrizes esparsas descritas na Tab. 1. O formato indicado do arquivo para todas as matrizes é <`nome da matriz`>`.txt`. Essas matrizes foram adaptadas a partir daquelas obtidas no site `https://www.cise.ufl.edu/research/sparse/matrices/`.

Matriz	**Área de Aplicação**	**n**	**nnz**
rdb968	Dinâmica de Fluidos Computacional	968	5632
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

Tabela 1: Matrizes a serem utilizadas nos experimentos numéricos

Importante: As matrizes listadas na Tab. 1, já com as devidas adaptações, encontram-se armazenadas no arquivo `matrizes.tgz` em anexo.

3.3 Como medir o tempo de processamento

Utilize a função `clock_gettime` dentro do seu código, conforme apresentado na Fig. 1. Não esqueça de compilar com a diretiva `-lrt`.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    struct timespec Start, End;
    double Elapsed_Time;


    /********** Inicia  a medicao ******************** */
    clock_gettime(CLOCK_MONOTONIC, &Start);



    /* TRECHO QUE GOSTARIA DE MEDIR O TEMPO DE EXERCUCAO */



    /********** Finaliza  a medicao ******************** */
    clock_gettime(CLOCK_MONOTONIC, &End);

    /********* Calcula o tempo gasto **********************/
    Elapsed_Time = End.tv_sec - Start.tv_sec + 1e-9*(End.tv_nsec - Start.tv_nsec);


    printf("O Tempo de execucao do trecho foi %lf segundos\n", Elapsed_Time);
    /*****************************************************************************/

    return 0;
}
```

Figura 1: Exemplo de como medir o tempo

3.4 Como apresentar os resultados

Os resultados dos experimentos devem ser apresentados conforme descrito na Tabela 2. Não esqueça de descrever hardware e software utilizados nos testes.

			Eliminação de Gauss	**Método SOR**		
Matriz	**n**	**nnz**	**Tempo**	**Iterações**	ω	**Tempo**
rdb968	968	5632				
rail_5177	5177	35185				
aft01	8205	125567				
FEM_3D_thermal1	17880	430740				
Dubcova2	65025	1030225				

Tabela 2: Como apresentar os resultados

4 Estrutura do Relatório

O relatório deve ser escrito na forma de um artigo científico. Utilize para esse fim o LaTeX, empregue as normas do padrão ABNT e salve a versão final em pdf. Siga exatamente o modelo proposto no arquivo `abntex2-modelo-artigo.tex` que segue anexo. O relatório deve conter obrigatoriamente as seguintes seções:

- **Introdução:** onde será apresentado um breve histórico do problema, os objetivos e a estrutura do trabalho.

- **Referencial Teórico:** onde será feito um pequeno resumo descrevendo todas as técnicas e métodos considerados.
- **Implementação:** onde serão apresentados a estrutura e comentários sobre partes significativas do código.
- **Experimentos Numéricos:** onde serão apresentados os exemplos testados, a descrição do hardware e software utilizados, além de comentários sobre cada experimento.
- **Conclusão:** onde serão sintetizadas as conclusões tiradas durante o desenvolvimento do trabalho.

5 Considerações gerais sobre o trabalho

- O trabalho (códigos mais relatório) deve ser enviado por e-mail para lmuniz@ifes.edu.br até o dia 04/10/2015 às 23:59h. Cada dia de atraso implicará em menos 1.0 ponto na nota final.
- O assunto do e-mail deve ser AN152:TRAB1:<nome1>:<nome2>:<nome3>, contendo em anexo, um arquivo do tipo TRAB1.zip. Neste caso <nome1>, <nome2> e <nome3> devem conter o nome e último sobrenome de cada participante (por exemplo, AN152:TRAB1:Leonardo Lima:Jose Silva:Joao Santos). Não utilize caracteres especiais como acentos, cedilha entre outros.
- Qualquer código que você tenha implementado deve ser anexado ao arquivo.zip descrito no item anterior. Não esqueça de anexar também o Makefile.
- Caso o arquivo seja enviado múltiplas vezes, apenas a última versão enviada será considerada.
- O trabalho poderá ser feito em grupo com até 03 participantes.
- A nota do trabalho só será divulgada após entrevista agendada com o grupo.
  OBS_1**:** No dia da entrevista, todos os componentes do grupo deverão estar presentes. Nesse momento, o professor escolherá um represente do grupo para ser avaliado. Os demais serão dispensados. A nota de cada componente do grupo será a mesma.
- A nota do trabalho será de até 10 pontos. Sendo 5.0 pontos pela confecção do relatório e outros 5.0 pontos pelos códigos implementados.
- Em relação aos códigos, obedeça exatamente a formatação de saída proposta na subseção 3.1 pois a correção será automatizada.
- Códigos que não apresentarem nenhuma otimização com respeito ao tratamento das esparsidades terão a nota severamente comprometida.
- Trabalhos com códigos considerados iguais receberão nota zero.
- O grupo que implementar o código mais eficiente, ou seja, mais rápido e com menor consumo de memória receberá um **bônus de 1.0 ponto na média semestral final**.
  OBS_2**:** Todos trabalhos serão executados novamente em uma máquina do LCAD.