CI 164 - Introdução à Computação Científica - Turma B - 2° Semestre de 2017

ESPECIFICAÇÃO DO TRABALHO PRÁTICO - Inversão de Matrizes

O objetivo deste trabalho é implementar um programa computacional que, dada uma matriz quadrada **A** de dimensão **n**, encontre a matriz inversa de **A (inv(A))**, tal que **A * inv(A) = I**, onde **I** é a matriz identidade.

Para tal, o programa deve utilizar o **Método da Eliminação de Gauss com Pivotamento Parcial, Fatoração LU e Refinamento**, conforme visto em aula:

1. Cálculo das matrizes L e U (Eliminação de Gauss com Pivotamento Parcial)
2. Solução do Sistema Linear (Retrosubstituição)
3. Refinamento a partir do Resíduo

A medida de erro a cada iteração do refinamento é a norma L^2 do resíduo (**||r||**):

- **R = I - A * inv(A)**
- **||r|| = sqrt(sum(R[i,j] ²)),** [i,j] = {1,2,...,n}

O programa deve "falhar graciosamente" com mensagens de erro apropriadas caso a matriz não seja inversivel, notificando o usuário sobre o erro ocorrido.

Execução do Programa

O pacote de software a ser construído deve gerar o executável chamado **invmat**, a ser executado da seguinte forma:

```
invmat [-i arquivo_entrada] [-o arquivo_saida] [-r N] -i k
```

Onde:

- **-i arquivo_entrada:** parâmetro opcional no qual **arquivo_entrada** é o caminho completo para o arquivo contendo a matriz a ser invertida. Em caso de ausência do parâmetro, a entrada deve ser lida de **stdin**.
- **-o arquivo_saida:** parâmetro opcional no qual **arquivo_saida** é o caminho completo para o arquivo que vai conter a matriz inversa. Em caso de ausência do parâmetro, a saída deve ser impressa em **stdout**.
- **-r N:** parâmetro opcional no qual **N** é a dimensão da matriz de entrada a ser gerada aleatoreamente de acordo com a função especificada ao final desta página
- **-i k:** Número de iterações de refinamento a serem executadas (>0)

Entrada de Dados

A matriz de entrada sempre é quadrada e possui a seguinte formatação (em ASCII):

```
n
a_11 a_12 ... a_1n
a_21 a_22 ... a_2n
 .    .        .
 .    .        .
a_n1 a_n2 ... a_nn
```

onde **1< n < 32768** indica o número de elementos em uma linha/coluna da matriz, e **a_ij** é um número em ponto flutuante de precisão dupla (double).

Saída de Dados

A matriz resultante deve ser impressa no mesmo formato da matriz de entrada. Os valores da matriz devem ser impressos com **printf("%.17g", valor)**. Algumas informações adicionais devem ser impressas, precedidas do caractere '#'

```
#
# iter 1: <||r||>
# iter 2: <||r||>
# ...
# iter k: <||r||>
# Tempo LU: <tempo para Cálculo das matrizes L e U>
# Tempo iter: <tempo médio para resolver o SL>
# Tempo residuo: <tempo médio para calcular o residuo>
#
n
a_11 a_12 ... a_1n
a_21 a_22 ... a_2n
 .   .       .
 .   .       .
a_n1 a_n2 ... a_nn
```

Onde:

- **iter k:** norma do resíduo após a k-ésima iteração;
- **Tempo:** deve ser calculado em segundos, utilizando-se a função especificada abaixo.
- **Tempo LU:** tempo para executar o processo Eliminação de Gauss e geração das matrizes LU. Não incluir tempo de leitura ou geração da matriz de entrada, nem a impressão dos resultados ou cálculo do erro.
- **Tempo iter:** Tempo médio para calcular as retrosubstituições e encontrar uma solução para o S.L. Inclui o tempo de somar o resultado do refinamento à solução original, caso isto seja feito separadamente.
- **Tempo residuo:** Tempo médio para calcular a norma do resíduo.

MENSAGENS DE ERRO:

Em caso de erros, uma mensagem explicando o ocorrido deve ser impressa em **stderr** e a execução do programa deve ser encerrada com código **diferente** de **0**.

Produto a ser Entregue

O trabalho deve ser desenvolvido por um grupo composto por no máximo DOIS alunos regularmente matriculados na disciplina. O grupo **NÃO PODE SER ALTERADO** na próxima etapa do trabalho.

Cada grupo deve entregar um pacote de software completo contendo os fontes em **linguagem C**. O pacote deve ser arquivado e compactado com tar(1) e gzip(1) em um arquivo chamado login1.tar.gz (se grupo com 1 membro) ou login1-login2.tar.gz (se grupo com 2 membros), onde login1 e login2 são os logins dos alunos que compõem o grupo.

O pacote deve ter a seguinte estrutura de diretório e arquivos:

- ./login1-login2/: diretório principal;
- ./login1-login2/doc;
- ./login1-login2/Makefile;

Note que a extração dos arquivos de login1-login2.tar.gz deve criar o diretório login1-login2 contendo todos os arquivo acima. Os arquivos fonte também devem estar contidos no diretório, ou em algum sub-diretório, desde que o Makefile funcione.

DOCUMENTAÇÃO

Os arquivos fonte devem ser documentados com Doxygen de forma a gerar uma documentação que contenha as seguintes informações:

- autoria do software, isto é, nome e RA dos membros do grupo;
- Documentação das estruturas de dados e de todas funções

Makefile

O arquivo Makefile deve possuir as regras necessárias para compilar os módulos individualmente e gerar o programa executável. As seguintes regras devem existir OBRIGATORIAMENTE:

- **all**: compila e produz um executável chamado **invmat** no diretório login1-login2/;
- **doc:** gera uma documentação em formato html utilizando os comentários do código escritos no formato do "Doxygen". Utilizar a opção `SOURCE_BROWSER=YES` para incluir o código fonte na documentação

- **clean**: remove todos os arquivos temporários e os arquivos gerados pelo Makefile (*.o, executável, etc.).

Entrega

O prazo final para a entrega deste trabalho é dia **24 de setembro de 2017, 23:55:00h**, IMPRETERIVELMENTE.

- O trabalho deve ser entregue via email para <**kunzle** at **inf** pt **ufpr** pt **br**>
- No texto de entrega DEVE CONSTAR OBRIGATORIAMENTE o Nome e Números de Registro Acadêmico (RA) dos membros do grupo;

Critérios de Avaliação

APENAS OS TRABALHOS QUE FUNCIONAREM SERÃO CORRIGIDOS. Se o trabalho não compilar ou acusar falha de segmentação (Segmentation fault) prematura durante os testes realizados pelo professor (sem que qualquer operação se efetue a contento), trará para o grupo NOTA 0 (ZERO). Também receberão NOTA 0 (ZERO) trabalhos plagiados de qualquer fonte, e/ou com códigos idênticos ou similares. Além disso, apenas trabalhos entregues no prazo marcado receberão nota.

Os itens de avaliação do trabalho e respectivas pontuações são:

- Qualidade da documentação: Doxygen (20 pontos)
- Entrada e saída: funcionamento do programa de acordo com a especificação no que tange execução, entrada e saída (10 pontos)
- Funcionamento: corretude das respostas nos testes executados (60 pontos)
- Eficiência das estruturas de dados utilizadas, desde que devidamente justificadas na documentação (10 pontos)

Defesa: A defesa do trabalho será oral, e definirá a nota individual de cada membro da equipe, de acordo com seu conhecimento a respeito do trabalho.

Função para Medição de Tempo

O tempo de execução deve ser medido em segundos, considerando tempo de relógio, utilizando a função especificada abaixo:

```
#include <time.h>

double timestamp(void){
    struct timeval tp;
    gettimeofday(&tp, NULL);
    return((double)(tp.tv_sec*1000.0 + tp.tv_usec/1000.0));
}
```

onde o tempo decorrido é medido pela diferença do "timestamp" medidos antes e depois da região de interesse.

Função para geração de matrizes aleatórias

As matrizes a serem invertidas, de tamanho NxN, podem ser geradas com a função especificada abaixo (ou similar).

```
#include <stdlib.h>

double *generateSquareRandomMatrix( unsigned int n )
{
  double *mat = NULL;

  /* return NULL if memory allocation fails */
  if ( ! (mat = (double *) malloc(n*n*sizeof(double))) )
    return (NULL);

  /* generate a randomly initialized matrix in row-major order */
  double *ptr = mat;
  double *end = mat + n*n;

  double invRandMax = 1.0/(double)RAND_MAX;

  while( ptr != end ) {
    *ptr++ = (double)rand() * invRandMax;
  }

  return (mat);
}
```

Antes da primeira chamada da função "generateSquareRandomMatrix()", e **somente uma vez em todo código**, você deve inicializar a sequência de números aleatóreos chamando a função:

```
srand( 20172 );
```