**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS**

**MÉTODOS NUMÉRICOS COMPUTACIONAIS**

**Lista de exercícios 01**

Thiago Henrique Gonçalves Mello - 201612060188 - thiagohgmello@gmail.com

**Questão 1)**

Para resolução da primeira questão da lista de exercícios, foi desenvolvida uma rotina em linguagem C. Primeiramente, criou-se um arquivo *header* nomeado “matrix.h” com o intuito de separar as rotinas desenvolvidas para cada tipo de problema. Nele, foram inseridas as funções responsáveis pela verificação da possibilidade de multiplicação entre matrizes, inicializador de matrizes, multiplicador e uma rotina específica para desalocar matrizes uma vez que estas são criadas dinamicamente.

A verificação é feita através da comparação entre o número de colunas da primeira matriz na multiplicação e o número de linhas da segunda. O tamanho dos elementos multiplicados é definido pelo usuário exigindo então uma requisição de dados inicialmente.

Com os tamanhos especificados, passa-se então para a multiplicação das matrizes, caso a operação possa ser realizada. Uma rotina em MATLAB foi feita com o intuito de comparar os resultados oriundos da rotina em C com precisões de ponto flutuante (*float*) e *double*. Para explorar de forma mais completa as vantagens e desvantagens de determinado tipo de variável, as entradas de cada uma das matrizes foram determinadas a partir de uma função matemática, a saber:

em que é uma variável semi-aleatória criada através da função “rand” da biblioteca *stdlib*.

A norma implementada para comparação de resultados foi a norma de Frobenius, uma vez que esta expressa de forma similar a média geométrica dos elementos da matriz.

A título de comparação entre as duas precisões, vários tamanhos de matrizes distintos foram gerados aleatoriamente e então comparadas com o resultado oriundo da rotina em MATLAB.

Com a precisão *float*, 5 tamanhos distintos de matrizes foram apresentados com os erros da norma de Frobenius expressos na Tabela I.

Tabela I. Erros percentuais sobre as normas de Frobenius para multiplicação de matrizes considerando precisão float. Comparação entre programa C e resultado do MATLAB.

|  |  |  |  |
| --- | --- | --- | --- |
| Tamanho das Matrizes | | | Erro (%) |
| A | B | C |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

De forma similar, o mesmo procedimento com matrizes do mesmo tamanho, porém com possibilidade de serem diferentes, foi realizado considerando precisão estendida (*double*). Os resultados estão na Tabela II.

Tabela II. Erros percentuais sobre as normas de Frobenius para multiplicação de matrizes considerando precisão double. Comparação entre programa C e resultado do MATLAB.

|  |  |  |  |
| --- | --- | --- | --- |
| Tamanho das Matrizes | | | Erro (%) |
| A | B | C |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Comparando os dados, vê-se que, mesmo apresentando erros baixos, variáveis *float* cometem erros cerca de um milhão de vezes superiores aos casos em que se utiliza *double*. Para matrizes maiores como por exemplo a última experimentada, os erros absolutos com números de ponto flutuante chegaram na casa de milhares de unidades, o que pode representar grandezas demasiadamente grandes para problemas específicos.

Outro ponto que merece ser ressaltado é o efeito da propagação do erro. Para o caso de multiplicação, houve apenas dois pontos de combinação de incertezas: 1) no cálculo do elemento em e 2) no produto entre os elementos, entretanto, caso houvesse mais operações subsequentes, a propagação ocasionaria problemas maiores, como será visto em outras questões da lista.

A desvantagem em se utilizar tamanhos maiores de variáveis ficou explícita ao avaliar o arquivo .txt gerado como saída do produto. Para o caso de precisão *double*, o arquivo teve cerca de de memória, o que impossibilitava até sua abertura em programas para edição deste tipo de extensão.

Portanto, o que se pode concluir neste problema é que, caso não seja necessária uma precisão exorbitante, variáveis do tipo *float* são interessantes.

**Questão 2)**

No caso de solução de sistemas lineares, tamanhos menores de matrizes foram implementados, uma vez que são geradas aleatoriamente, fazendo com que a chance de surgimento de matrizes singulares seja consideravelmente elevada. Com base nisto, sistemas tanto superiores quanto inferiores foram utilizados. Os resultados da comparação entre as normas de Frobenius para o sistema diagonal superior utilizando variáveis do tipo *float* e MATLAB estão na Tabela III enquanto que para sistemas diagonais inferiores estão na Tabela IV. De forma similar, as Tabelas V e VI apresentam os comparativos quando as variáveis são do tipo *double*.

Tabela III. Erros percentuais sobre as normas de Frobenius para solução de sistemas lineares triangulares superiores considerando precisão float. Comparação entre programa C e resultado do MATLAB.

|  |  |
| --- | --- |
| Execução | Erro (%) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tabela IV. Erros percentuais sobre as normas de Frobenius para solução de sistemas lineares triangulares inferiores considerando precisão float. Comparação entre programa C e resultado do MATLAB.

|  |  |
| --- | --- |
| Execução | Erro (%) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tabela V. Erros percentuais sobre as normas de Frobenius para solução de sistemas lineares triangulares superiores considerando precisão double. Comparação entre programa C e resultado do MATLAB.

|  |  |
| --- | --- |
| Execução | Erro (%) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tabela VI. Erros percentuais sobre as normas de Frobenius para solução de sistemas lineares triangulares superiores considerando precisão double. Comparação entre programa C e resultado do MATLAB.

|  |  |
| --- | --- |
| Execução | Erro (%) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Mais uma vez, como esperado, percebe-se uma precisão consideravelmente maior quando se utiliza variáveis do tipo *double*. Como a aplicação foi pequena (solução de sistemas lineares ), a diferença no tempo de execução não foi perceptível. Apesar disso, caso se tenha sistemas de mais alta ordem, operações com maior precisão apresentam maior custo computacional.

Outro ponto que merece destaque é o caso de propagação de erros. Para sistemas lineares, como a determinação de uma variável pode depender diretamente da definição de outras, a propagação de erros pode ser considerável à medida que se eleva o tamanho do sistema. Com isso, deve-se conhecer bem o problema de forma a escolher o menor custo de implementação, seja ele de tempo ou de espaço.

**Questão 3)**

A implementação do pivoteamento foi o problema que mais demandou tempo dentre aqueles apresentados neste trabalho. Apesar disso, alguns pontos são de suma importância de serem ressaltados: 1) quando se trata de problemas essencialmente numéricos, igualdade não se mostram bem-vindas em comparações; 2) processos de operação contínua entre elementos da matriz aumentam a propagação de erros e 3) operações intermediárias devem ser feitas com a maior precisão dentro daquelas disponíveis.

O primeiro ponto que mereceu destaque foi oriundo de um problema deparado durante o desenvolvimento do algoritmo. Por serem comparações numéricas, aproximações são inerentes ao processo e, com elas, os erros numéricos também o são.

No caso estudado, comparações entre números e o valor nulo retornavam falso mesmo quando o processo era, na realidade, verdadeiro. Isso se dava porque havia propagação de erro inerente aos cálculos. Mesmo incorporando variáveis de maior precisão, o problema tende a existir, uma vez que dois valores nunca serão comparados em toda sua magnitude. Por ser algo inerente à programação, tal erro não foi atribuído à escolha de variáveis, sendo então presente nos dois casos abordados.

O outro ponto ressaltado fica ainda mais evidente quando se tem sucessivas operações com os mesmos elementos. Isto porque um número que essencialmente é uma dízima, seja ela periódica ou não, será sempre representado por um valor finito. Ao ser alterado de qualquer forma, o erro do truncamento também é alterado na mesma magnitude, tornando-o mais significativo.

Por fim, o terceiro ponto se refere às variáveis como de somatório, que auxiliam no resultado final, mas existem apenas por limitação do ambiente de programação. Com isso, diminui-se a chance de propagação de erros.

Assim como nos casos anteriores, o mesmo programa foi executado cinco vezes com determinada precisão e então esta foi alterada e o programa executado cinco vezes. A geração de valores das matrizes segue a mesma regra expressa em e por ser aleatória, possui extrema limitação na geração de matrizes não singulares e, consequentemente, sistemas lineares possíveis e determinados. Como saídas, tem-se as variáveis de interesse de , a saber, os valores de , o determinante da matriz e a norma euclidiana de . A Tabela VII expressa a comparação entre o mesmo sistema solucionado utilizando rotina em C e em MATLAB para variáveis do tipo *float* e a Tabela VIII para variáveis do tipo *double*.

Tabela VII. Erros percentuais sobre as normas euclidianas e o determinante para solução de sistemas lineares pelo método de Gauss com pivoteamento considerando precisão float. Comparação entre programa C e resultado do MATLAB.

|  |  |  |
| --- | --- | --- |
| Execução | Erro determinante (%) | Erro norma (%) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Tabela VIII. Erros percentuais sobre as normas euclidianas e o determinante para solução de sistemas lineares pelo método de Gauss com pivoteamento considerando precisão double. Comparação entre programa C e resultado do MATLAB.

|  |  |  |
| --- | --- | --- |
| Execução | Erro determinante (%) | Erro norma (%) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Como esperado mais uma vez, os erros se mostraram consideravelmente maiores quando as simulações foram feitas com variáveis do tipo *float*. Os erros elevados na Tabela VIII quando se compara os valores dos determinantes é explicado pela precisão com a variáveis é impressa na tela. Para o caso dos determinantes, os valores foram dados com precisão de seis casas decimais. Mesmo com isso, percebe-se uma diferença entre os dois tipos de variáveis de cerca de vezes, explicitando ainda mais as consequências da propagação de erros de aproximação nas operações.

Com isso, assim como foi dito nos casos anteriores, deve-se possuir bem definida a relação entre custo computacional e precisão demandada pelo projeto.