MAC 300 – Métodos Numéricos de Álgebra Linear

Exercício Programa 1: Resolução de sistemas de equações lineares

Este exercício programa tem dois objetivos: (i) estudar e desenvolver algoritmos para resolver sistemas de equações lineares; (ii) constatar que os algoritmos devem ser desenvolvidos levando em consideração as estruturas de dados disponíveis na plataforma onde serão implementados.

Para que os programas possam ser facilmente testados, adotaremos um formato de entrada padronizado para os sistemas de equações. Os arquivos de entrada deverão ter o seguinte formato: na primeira linha do arquivo deverá haver um único número inteiro n indicando a dimensão do sistema (número de variáveis e equações); depois n^2 linhas com três números (dois inteiros e um real) indicando linha, coluna e valor do elemento da matriz de coeficientes, respectivamente. Finalmente n linhas com um inteiro e um real indicando a posição e o valor dos elementos do lado direito do sistema.

O exercício programa deverá ser implementado em C e Fortran. O exercício programa consta de três partes:

Primeira parte: Linguagem C

Nesta etapa devem ser implementadas as seguintes funções:

- `int lucol(int n, double A[][nmax], int p[]);`

  Recebe um inteiro n e uma matriz $A \in \mathbb{R}^{n\times n}$ e devolve, na própria A, as matrizes L e U da decomposição LU da matriz PA, onde P é a matriz de permutação obtida ao utilizar pivoteamento parcial. A matriz P deve ser devolvida no vetor $p \in \mathbb{N}^n$. A função também devolve 0 se a decomposição LU foi calculada com sucesso e -1 se a matriz A for singular.

- `int sscol(int n, double A[][nmax], int p[], double b[]);`

  Recebe um inteiro n, uma matriz $A \in \mathbb{R}^{n\times n}$ e um vetor $p \in \mathbb{N}^n$ (ambos saída da função anterior) e um vetor $b \in \mathbb{R}^n$ e devolve, no próprio b, a solução do sistema $LUx = Pb$. Para isso, calcula-se primeiro Pb (e guarda-se em b). Em seguida, resolve-se o sistema triangular inferior $Ly = Pb$ (guardando y em b) e, finalmente, resolve-se o sistema triangular superior $Ux = y$ (guardando x em b). A função deve devolver 0 se o sistema foi resolvido com sucesso e -1 se a matriz U for singular.

- As 2 rotinas acima mencionadas devem ser orientadas a coluna. As 2 de baixo devem fazer o mesmo, só que devem ser implementadas orientadas a linha.

  `int lurow(int n, double A[][nmax], int p[]);`

  `int ssrow(int n, double A[][nmax], int p[], double b[]);`

Para testar suas implementações da decomposição LU utilize os sistemas (com matrizes de coeficientes simétricas) que estão nos arquivos m1.dat, m2.dat, ..., m9.dat (você deve gerar

estes arquivos com o gerador genmat.c). Os sistemas dos arquivos m8.dat e m9.dat admitem infinitas soluções pois suas matrizes de coeficientes são singulares. Seu algoritmo deve detectar isso! Já os sistemas que estão nos arquivos m1.dat, m2.dat, ..., m7.dat têm como solução única $x = (x_0, x_1, \ldots, x_{n-1})^T$ tal que $x_i = 1 + i\%(n/100)$, onde n é a dimensão do problema.

Depois dos testes, verifique se as versões orientadas a linha e a coluna estão implementadas corretamente (i.e., percorrem, nos seus laços mais internos, a matriz de coeficientes por linha e coluna, respectivamente). Para isso, meça o tempo de execução das duas versões para cada um dos 7 primeiros problemas. Entregue, junto com o programa (código-fonte) e a tabela relacionada aos seus próprios testes, um breve comentário sobre as diferenças de tempo encontradas para cada versão (orientada a linha e a coluna) do programa.

Segunda parte: Linguagem Fortran

Nesta etapa devem ser implementadas as seguintes funções:

```
      integer function lucol(n,lda,A,p)
      implicit none
C     SCALAR ARGUMENTS
      integer n
C     ARRAY ARGUMENTS
      integer p(n)
      double precision A(lda,n)
C     LOCAL SCALARS
      integer ...
      double precision ...
      ...
      return lucol
      end
```

```
      integer function sscol(n,lda,A,p,b)
      implicit none
C     SCALAR ARGUMENTS
      integer n
C     ARRAY ARGUMENTS
      integer p(n)
      double precision A(lda,n),b(n)
C     LOCAL SCALARS
      integer ...
      double precision ...
      ...
      return sscol
      end
```

Implemente também as respectivas versões, `lurow` e `ssrow`, orientadas a linha. As rotinas devem ter a mesma funcionalidade das suas versões correspondentes na linguagem C. Faça com elas os mesmos testes da etapa anterior.

Terceira parte: `f2c`

Use `f2c` para converter as rotinas da Parte 2 de Fortran para C. Refaça os experimentos com as rotinas convertidas. Tire conclusões sobre o seu comportamento.

Importante 1: Além dos códigos-fonte, deve ser entregue um relatório com as tabelas e uma analise suscinta dos resultados obtidos. O relatório é indispensável para a avaliação do trabalho.

Importante 2: Um detalhe que vale para todas as rotinas que deverão ser implementadas: nenhuma delas precisará da declaração de vetores ou matrizes locais.

Importante 3: Seus programas devem compilar com `gcc` e `gfortran` em Linux. Inclua programas principais que chamem as subrotinas. Se o comando de compilação não for o óbvio, indique como devem ser compilados os seus programas.