Avaliação Prática 1

Cálculo Numérico (SME0104)
Professora Cynthia Lage Ferreira

16 de maio de 2021

Orientações Gerais

- Esta avaliação é **individual** e deverá ser desenvolvida na plataforma Colab (https://colab.research.google.com/).
- Cada aluno deverá produzir um **arquivo .ipynb** contendo a solução dos exercícios.
- Os arquivos deverão estar identificados da seguinte forma: **NOMEDOALUNO-NoUSP-TURMA.ipynb** a fim de facilitar a organização das atividades pela professora.
- Os arquivos deverão ser **enviados até às 20h do dia 18/05** através da plataforma e-disciplinas da USP (https://edisciplinas.usp.br/) respeitando o prazo. **Os arquivos recebidos por e-mail não serão corrigidos.**
- Apenas os alunos que estiverem com a **situação regularizada no Sistema Jupiter** terão suas avaliações corrigidas.
- Todos os códigos utilizados para resolver os problemas deverão ser apresentados, executados e minimamente comentados. **Questões com respostas sem justificativas não serão consideradas.**
- Os alunos que quiserem poderão apresentar um resumo teórico referente aos conteúdos de cada questão. A realização desta tarefa poderá gerar uma bonificação ao aluno, a critério da professora.
- As funções prontas do Python dos métodos estudados poderão ser utilizadas para validar os resultados obtidos, mas não as utilize como ÚNICA forma de solução dos exercícios.

BOM TRABALHO !

1 Sistemas Lineares com Matrizes Simétricas

1) Considere a matriz A e o vetor b dados abaixo

$$A = \begin{bmatrix} 100 & -4 & 1 & 0 & \cdots & \cdots & 0 \\ -4 & 100 & -4 & 1 & \ddots & & \vdots \\ 1 & -4 & 100 & -4 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -4 & 100 & -4 & 1 \\ \vdots & & \ddots & 1 & -4 & 100 & -4 \\ 0 & \cdots & \cdots & 0 & 1 & -4 & 100 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1/n^4 \\ 1/n^4 \\ \vdots \\ \vdots \\ \vdots \\ 1/n^4 \\ 1/n^4 \end{bmatrix}. \tag{1}$$

Seja n a dimensão do problema.

a) Escreva um código que monte a matriz A para $n = 1000$.

b) Escreva um código que faça a decomposição de Cholesky de uma matriz simétrica definida positiva qualquer.

c) Usando as rotinas implementadas anteriormente, escreva um código para resolver um sistema $A\mathbf{x} = \mathbf{b}$.

d) Escreva um código implementando o método de Jacobi para resolver um sistema $A\mathbf{x} = \mathbf{b}$. Utilize o erro absoluto como critério de parada.

e) Vamos comparar o método direto de Cholesky com o iterativo de Jacobi neste exemplo. Observe quanto tempo leva para resolver o sistema usando Cholesky. Quantas iterações foram necessárias no método de Jacobi para obtermos a mesma precisão da solução dada pelo método de Cholesky ?

f) É possível melhorar a implementação da decomposição de Cholesky para o exemplo em questão ?

2 Método de Newton Para Sistemas Não-Lineares

Em muitas aplicações, lidamos com sistemas não-lineares do tipo

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) & = 0 \\ f_2(x_1, x_2, \cdots, x_n) & = 0 \\ \vdots & \\ f_n(x_1, x_2, \cdots, x_n) & = 0 \end{cases}. \tag{2}$$

Escrevendo esse sistema na forma vetorial, temos

$$\mathbf{F}(\mathbf{x}) = 0, \text{ em que } \mathbf{F} = \begin{bmatrix} f_1(x_1, x_2, \cdots, x_n) \\ \vdots \\ f_n(x_1, x_2, \cdots, x_n) \end{bmatrix} \text{ e } \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{3}$$

Podemos generalizar o método de Newton para uma função vetorial $\mathbf{F}(\mathbf{x})$, em que cada iteração é da forma

$$\mathbf{x}^{k+1} = \mathbf{x}^k - J_{\mathbf{F}}(\mathbf{x}^k)^{-1}\mathbf{F}(\mathbf{x}^k). \tag{4}$$

$J_{\mathbf{F}}(\mathbf{x}^k)$ designa a matriz Jacobiana de $\mathbf{F}$ avaliada no ponto $\mathbf{x}^k$. Lembramos que a matriz Jacobiana de uma função $\mathbf{F}$ é dada por

$$J_{\mathbf{F}} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{5}$$

1) Observando a equação (4), vê-se a necessidade de calcular a matriz inversa da Jacobiana. É possível calcular matrizes inversas usando a decomposição LU, sendo assim, implemente um código que:

a) Calcule a fatoração LU de uma matriz qualquer;

b) Resolva um sistema linear cujo lado direito é uma matriz;

c) Finalmente, calcule a inversa de uma matriz qualquer resolvendo vários sistemas lineares.

2) Implemente o método de Newton para sistemas usando a rotina implementada no item anterior. Lembre-se de especificar o critério de parada utilizado.

Calcular inversas de matrizes é um processo que demanda muito custo computacional (como pode ser observado nos itens anteriores, precisa-se de uma fatoração LU e da resolução de vários sistemas lineares). Por isso, ao usar o método de Newton para sistemas, faz-se algumas manipulações na equação (4) evitando o cálculo da inversa da matriz Jacobiana. Define-se um vetor $\mathbf{z} = \mathbf{x}^{k+1} - \mathbf{x}^k$ e resolve-se o sistema linear $J_{\mathbf{F}}(\mathbf{x}^k)\mathbf{z} = \mathbf{F}(\mathbf{x}^k)$. Deste modo, o algoritmo para resolver um sistema de equações não-lineares usando o método de Newton é :

1. Numa dada iteração k, resolve-se o sistema $J_{\mathbf{F}}(\mathbf{x}^k)\mathbf{z} = \mathbf{F}(\mathbf{x}^k)$,
2. Atualiza-se $\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{z}$.

3) Implemente novamente o método de Newton para sistemas usando o algoritmo anterior. Utilize as rotinas já implementadas neste trabalho para resolução do sistema linear, indicando sempre qual está usando.

4) Dada a equação de um círculo $(x-a)^2 + (y-b)^2 = R^2$ e três pontos que passam por esse círculo

x	8.21	0.34	5.96
y	0.00	6.62	-1.12

a) Monte um sistema não-linear para determinar a, b e R.

b) Resolva o sistema não-linear utilizando os códigos feitos nos itens **2** e **3**.

c) Houve melhora no tempo de execução do código implementado no item **3**?

d) Finalmente, com os resultados a, b e R, utilizando a equação do círculo dada por $(x-a)^2 + (y-b)^2 = R^2$, imprima o gráfico que representa esse círculo.