

# Trabalho 2

Cálculo Numérico (SME0104)  
Professora Cynthia Lage Ferreira

06 de junho de 2024

## Orientações Gerais

- Esta avaliação é **individual ou em dupla** e deverá ser desenvolvida na plataforma Colab (<https://colab.research.google.com/>).
- Cada aluno/dupla deverá produzir um **arquivo .ipynb** contendo tanto a parte escrita (teórica) quanto a parte prática (códigos em Python) de cada um dos exercícios.
- Os arquivos deverão estar identificados da seguinte forma: **NOMEDOALUNO1+NOMEDOALUNO2.ipynb** a fim de facilitar a organização das atividades pela professora. Enviem apenas um arquivo por dupla.
- Os arquivos deverão ser **enviados até às 20h do dia 09/06/2024** através da plataforma e-disciplinas da USP (<https://edisciplinas.usp.br/>). **Os arquivos recebidos fora do prazo ou por e-mail não serão corrigidos.**
- Apenas os alunos que estiverem com a **situação regularizada no Sistema Jupiter** terão suas avaliações corrigidas.
- Todos os exercícios deverão conter justificativas teóricas e 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.

# 1 Decomposição em Valores Singulares (SVD)

A decomposição SVD de uma matriz  $A_{m \times n}$  tem a forma

$$A = U \Sigma V^T \quad (1)$$

em que  $U$  é uma matriz  $m \times n$  ortogonal,  $V$  uma matriz também ortogonal com dimensão  $n \times n$  e  $\Sigma$  uma matriz diagonal  $m \times n$  com entradas

$$\sigma_{ij} = \begin{cases} 0 & \text{para } i \neq j \\ \sigma_i \geq 0 & \text{para } i = j \end{cases} \quad (2)$$

Esses valores  $\sigma_i$  são chamados de valores singulares de  $A$  e geralmente são ordenados tais que  $\sigma_{i-1} \geq \sigma_i$ ,  $i = 2, \dots, \min\{m, n\}$ . Já as colunas de  $U$  e  $V$  são os vetores singulares a esquerda e a direita, respectivamente. Esta decomposição está diretamente ligada a algoritmos para calcular autovalores e autovetores de matrizes. Os valores singulares de  $A$  são as raízes quadradas dos autovalores de  $A^T A$  e as colunas  $U$  e  $V$  são os autovetores ortonormais de  $AA^T$  e  $A^T A$  respectivamente.

Ainda, para uma matriz simétrica  $B_{n \times n}$ , a decomposição QR pode ser usada para calcular **todos** os seus autovalores e autovetores usando sucessivas decomposições até que se obtenha uma matriz diagonal (ou muito próxima de uma diagonal). O processo envolvido é

1.  $B_1 = B$  decompõe-se a matriz  $B_1 = Q_1 R_1$
2.  $B_2 = R_1 Q_1$  decompõe-se a matriz  $B_2 = Q_2 R_2$
3.  $B_3 = R_2 Q_2$  e então  $B_3 = Q_3 R_3$
4. Repete-se essas iterações até  $B_k = R_{k-1} Q_{k-1}$

como trata-se de um processo iterativo, é importante escolher um bom critério de parada. Dentre os critérios mais usados, pode-se limitar o número de iterações  $k$  por um máximo de iterações  $k_{max}$ , verificar se os elementos da matriz fora da diagonal estão tão próximos de zero quanto se queira usando uma tolerância

$$\max_{i < j} \{|b_{ij}|\} < \epsilon \quad (3)$$

ou verificar se  $off(B) < \epsilon$  em que

$$off(B) = \sqrt{\|B\|_F^2 - \sum_{i=1}^n b_{ii}^2} \quad (4)$$

Este método é conhecido como método de Francis, ao final do processo iterativo, tem-se

$$B_k = V^T B V \quad (5)$$

em que  $V = Q_1 Q_2 \dots Q_{k-1}$ , ou seja,  $B$  e  $B_k$  são matrizes semelhantes e possuem os mesmos autovalores. Além disso,  $B_k$ , como dito anteriormente, converge para uma matriz diagonal, ou seja, os elementos da diagonal de  $B_k$  fornecem uma aproximação para os autovalores de  $B$  e as colunas da matriz  $V = Q_1 Q_2 \dots Q_{k-1}$  são aproximações dos respectivos autovetores. O método de Francis pode ser usado para obter a decomposição SVD de uma matriz *qualquer*  $A_{m \times n}$  ao ser aplicado nas matrizes simétricas  $AA^T$  e  $A^T A$ , uma vez que

- $AA^T = (U \Sigma V^T)(U \Sigma V^T)^T = U \Sigma (V^T V) \Sigma U^T = U \Sigma^2 U^T$  e
- $A^T A = (U \Sigma V^T)^T (U \Sigma V^T) = V \Sigma (U^T U) \Sigma V^T = V \Sigma^2 V^T$ .

a) Escreva um código implementando o método de Francis usando a função `numpy.linalg.qr` para obter as decomposições QR necessárias.

b) Usando a rotina implementada para o método de Francis, escreva um código que retorne a decomposição SVD de uma matriz qualquer  $A_{m \times n}$ . *Dica: use o método de Francis apenas uma vez em  $AA^T$  ou em  $A^T A$  e obtenha a matriz faltante a partir da expressão da decomposição SVD de  $A$ .*

c) Podemos armazenar uma imagem em uma matriz  $A_{m \times n}$ . Toda imagem consiste em um conjunto de pixels que são os blocos de construção dessa imagem. Cada pixel representa a cor ou a intensidade da luz em um local específico na imagem. Em uma imagem em escala de cinza em formato PNG, cada pixel tem um valor entre 0 e 1, em que 0 corresponde ao preto e 1 corresponde ao branco. Assim, uma imagem em escala de cinza com  $m \times n$  pixels pode ser armazenada em uma matriz  $m \times n$  com valores entre 0 e 1. Use a função `imread()` da biblioteca Matplotlib do Python para carregar a imagem `cat.png` em escala de cinza. Depois, use as decomposições SVD `numpy.linalg.svd` e a implementada no item 2) para comprimir a imagem, representando-a por  $k$  ( $k < \min\{m, n\}$ ) valores singulares, isto é, troque a matriz  $A$  por  $A_k = U[:, :k] * \Sigma[:, :k] * V^T[:, :k]$ . Plote a imagem original e a imagem 50% e 70% comprimida e compare os resultados. Para tal, use a função `imshow()`, também da biblioteca Matplotlib.

## 2 Interpolação

Para a função

$$f(t) = \frac{1}{1 + 25t^2} \quad (6)$$

no intervalo  $[-1, 1]$  faça:

- Implemente as interpolações de Lagrange e de Newton.
- Usando 11 pontos igualmente espaçados dentro do intervalo dado, calcule as interpolações de Lagrange e Newton com o código implementado no item anterior. Mostre os resultados em dois gráficos separados. Que resultado teórico justifica o fato das duas soluções serem iguais?
- Repita o processo com 21 pontos. O que acontece? Exiba o gráfico das soluções comparando com a exata.
- Usando a função `scipy.interpolate.interp1d` calcule a interpolação usando *spline* linear e cúbica, considerando 21 pontos igualmente espaçados. Exiba os gráficos e comente as diferenças das soluções deste item para os anteriores.
- Repita os itens b) e c) com nós de Chebyshev. Comente os resultados obtidos. Por que este resultado é melhor do que os resultados obtidos nos itens b) e c) ?

## 3 Mínimos Quadrados

Vamos supor que os casos acumulados de Covid-19, no período inicial da pandemia, de 26 de fevereiro de 2020 a 18 de junho de 2020 são dados em *casosacumuladosbrasilatuaizado.txt*. O objetivo deste exercício é estudar o ajuste dos dados, no sentido dos mínimos quadrados, a uma função  $g(x) = ab^x$ , com  $a, b \in \mathbb{R}$ . Vamos utilizar os códigos implementados em aula:

```
import numpy as np
```

```
def mmq(x, y, k):
```

```
    X = np.vander(x, k)
    A = np.transpose(X).dot(X)
    b = np.transpose(X).dot(y)
    a = np.linalg.solve(A, b)
    return a
```

```
def mmqQR(x, y, k):
```

```
    X = np.vander(x, k)
    (Q, R) = np.linalg.qr(X)
    b = np.transpose(Q).dot(y)
    a = np.linalg.solve(R, b)
    return a
```

- Explique cada um dos códigos dados acima. O que está sendo calculado ?
- Aproxime, no sentido dos mínimos quadrados, os dados do período completo, de 26 de fevereiro de 2020 a 18 de junho de 2020, por uma função  $g(x) = ab^x$ , com  $a, b \in \mathbb{R}$ . Use um dos códigos dados acima.
- Repita o item b) usando apenas os 20 primeiros dias.
- Repita o item b) usando apenas os 50 últimos dias.
- Compare os itens b), c) e d). Que tipo de informação os dados nos fornecem ?