MINISTÉRIO DA DEFESA
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
Seção de Engenharia de Defesa (SE/10)

EE 600200: Álgebra Linear Computacional

Lista de Exercícios #3

Cap Suzane GAERTNER Martins

Rio de Janeiro, RJ
JUNHO de 2025

$1^{\underline{a}}$ Questão

[FORD] Exercício 9.2: forneça a contagem de flops para cada operação de matriz.

a. Multiplicação de uma Matriz A $m \times n$ por um vetor x $n \times 1$

Cada elemento do resultado é um produto interno de uma linha de A pelo vetor x:
• Cada linha: n multiplicações + $n-1$ somas, resultando em $2n-1 flops$;
• Total para m linhas: $m(2n-1)flops$.

b. O produto xy^T, onde x é $m \times 1$ e y é $p \times 1$

xy^T resulta em uma matriz $m \times p$, onde cada elemento é $x_i \cdot y_j$
• Cada elemento: 1 multiplicação;
• Total: $m \times p$ multiplicações (sem somas).

c. Se u e v são vetores $n \times 1$, calcule $\left(\frac{\langle v,u\rangle}{\|u\|^2}\right) u$

Passos:

1. $\langle v, u\rangle$: n multiplicações + $n-1$ somas resultando em $2n-1flops$;

2. $\|u\|^2$: n multiplicações + $n-1$ somas resultando em $2n-1flops$;

3. Divisão de escalares: 1 flop;

4. Multiplicação escalar por vetor: n multiplicações

Total:
$$(2n-1) + (2n-1) + 1 + n = 5n - 1$$

d. $\|A\|_\infty$ para matriz A $m \times n$

($\|A\|_\infty$ é o maior valor absoluto da soma dos elementos por linha)
Para cada linha:
• $n-1$ somas para cada linha, m linhas $\rightarrow m(n-1)flops$;
• Encontrar máximo entre m somas: $m-1$ comparações/flops.
Total:
$$m(n-1) + (m-1)$$

e. $\|A\|_1$ para matriz A $m \times n$

($\|A\|_1$ é o maior valor absoluto da soma dos elementos por coluna)
Para cada coluna:
• $m - 1$ somas por coluna, n colunas $\rightarrow n(m-1) flops$;
• Encontrar máximo entre n somas: $n - 1$ comparações/flops.
Total:

$$n(m-1) + (n-1)$$

f. traço(A), onde A é uma matriz $n \times n$

O traço de A é a soma dos elementos da diagonal:
• $n - 1$ somas (nenhuma multiplicação é necessária).
Total: $n - 1$ flops

2ª Questão

[FORD] Exercício 9.18: Implemente o problema que calcula o produto vetorial de vetores u e v, em uma função Python chamada crossprod. Após a implementação, crie vetores u e v e calcule:

- $u \times v$
- $v \times u$
- $\langle u \times v, u \rangle$
- $\langle v \times u, v \rangle$

Solução
Arquivo "crossprod.py"

```python
import numpy as np

def crossprod(u, v):
    """
    Calcula o produto vetorial entre dois vetores 3D u e v.
    """
    x = u[1]*v[2] - u[2]*v[1]
    y = u[2]*v[0] - u[0]*v[2]
    z = u[0]*v[1] - u[1]*v[0]
    return np.array([x, y, z])

# Exemplos de vetores
u = np.array([1, 2, 3])
v = np.array([4, 5, 6])

# Produto vetorial
uv = crossprod(u, v)
vu = crossprod(v, u)

print('u x v =', uv)
print('v x u =', vu)

# Produto escalar entre (u x v) e u
dot1 = np.dot(uv, u)
# Produto escalar entre (v x u) e v
dot2 = np.dot(vu, v)
```

```python

print('(u x v) · u =', dot1)
print('(v x u) · v =', dot2)
```

Para os vetores $u = [1, 2, 3]$ e $v = [4, 5, 6]$, a saída é:

$$u \times v = [-3,\ 6,\ -3]$$

$$v \times u = [3,\ -6,\ 3]$$

$$(u \times v) \cdot u = 0$$

$$(v \times u) \cdot v = 0$$

$3^{\underline{a}}$ Questão

[FORD] Exercício 10.6: Seja $f(x) = \mathbf{ln}\ x$

a. Mostre que o número de condicionamento de f em x é $c(x) = \frac{1}{|\ln x|}$

O número de condicionamento relativo de uma função $f(x)$ no ponto x é dado por:

$$c(x) = \left| \frac{x f'(x)}{f(x)} \right|$$

Para $f(x) = \ln x$:

$$f'(x) = \frac{1}{x}$$

Substituindo:

$$c(x) = \left| \frac{x \cdot \frac{1}{x}}{\ln x} \right| = \left| \frac{1}{\ln x} \right|$$

Portanto,

$$\boxed{c(x) = \frac{1}{|\ln x|}}$$

b. Usando o resultado acima, mostre que $\ln x$ é mal-condicionada perto de $x = 1$

Quando x se aproxima de 1, temos que $\ln x \to 0$.
Pela fórmula do número de condicionamento:

$$c(x) = \frac{1}{|\ln x|}$$

Logo, quando $x \to 1$, $|\ln x| \to 0$ e, portanto, $c(x) \to \infty$.
Assim, o número de condicionamento torna-se muito grande quando x está próximo de 1, indicando que pequenas variações em x podem causar grandes variações em $\ln x$.
Conclusão: $\ln x$ é **mal-condicionada** (*ill-conditioned*) perto de $x = 1$, pois o número de condicionamento tende ao infinito nessa região.

4ª Questão

[FORD] Exercício 10.12: Seja $A = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}$**. Para quais valores de** a **a matriz** A **é mal-condicionada?**

O número de condicionamento na norma 2 de uma matriz simétrica é dado por:

$$\kappa_2(A) = \frac{\max(|\lambda_1|, |\lambda_2|)}{\min(|\lambda_1|, |\lambda_2|)}$$

onde λ_1 e λ_2 são os autovalores de A.

Os autovalores de A são obtidos resolvendo:

$$\det(A - \lambda I) = 0$$

$$\begin{vmatrix} 1-\lambda & a \\ a & 1-\lambda \end{vmatrix} = (1-\lambda)^2 - a^2 = 0$$

$$(1-\lambda)^2 = a^2$$

$$1 - \lambda = \pm a$$

$$\lambda = 1 \mp a$$

Logo, os autovalores são:

$$\lambda_1 = 1 + a, \quad \lambda_2 = 1 - a$$

Assim,

$$\kappa_2(A) = \frac{\max(|1+a|, |1-a|)}{\min(|1+a|, |1-a|)}$$

A matriz é mal-condicionada quando o denominador se aproxima de zero, ou seja, quando $|1+a|$ ou $|1-a|$ é próximo de zero. Portanto,

$$A \text{ é mal-condicionada para } a \approx 1 \text{ ou } a \approx -1.$$

O que acontece quando $a \to \infty$**?**

Quando a tende ao infinito,

$$\lambda_1 \approx a, \qquad \lambda_2 \approx -a$$

Os módulos dos autovalores são iguais: $|\lambda_1| \approx |\lambda_2| \approx |a|$.

Assim,

$$\kappa_2(A) = \frac{|a|}{|a|} = 1$$

Portanto, à medida que $a \to \infty$, a matriz A se torna bem-condicionada, pois o número de condicionamento tende a 1.

Resumo:

- A é mal-condicionada para valores de a próximos de 1 ou -1, pois um dos autovalores se aproxima de zero, tornando a matriz quase singular.

- Quando $a \to \infty$ (ou $a \to -\infty$), A se torna bem-condicionada, pois os autovalores têm grandes módulos e o número de condicionamento tende a 1.

5ª Questão

[FORD] Exercício 10.21:

A matriz bidiagonal de Wilkinson tem a forma geral

$$A = \begin{bmatrix} n & n & & & \\ & n-1 & n & & \\ & & \ddots & \ddots & \\ & & & 2 & n \\ & & & & 1 \end{bmatrix}$$

e é frequentemente utilizada para fins de teste.

(a) Crie uma função anônima utilizando `diag` que construa uma matriz bidiagonal de Wilkinson de ordem $n \times n$.

(b) Faça o gráfico do número de condição das matrizes bidiagonais de Wilkinson de ordem 1 até 15. Quais são suas conclusões?

(c) Autovalores são difíceis de calcular com precisão. A matriz bidiagonal de Wilkinson 20×20, A, possui autovalores $\lambda = 20, 19, \ldots, 1$. Esta matriz ilustra que, mesmo quando os autovalores de uma matriz não são iguais ou nem mesmo próximos, um problema de autovalores pode ser mal condicionado. Calcule os autovalores de A, perturbe $A(20, 1)$ por 10^{-10} e calcule novamente os autovalores. Comente os resultados.

(OBS.: ignorar a parte de a função pedida na letra (a) ser uma "anonymous function". Basta criar uma função no padrão do Python.)

Solução
Arquivo "wilkinson_bidiagonal.py"

```python
import numpy as np
import matplotlib.pyplot as plt

def wilkinson_bidiagonal(n):
    A = np.zeros((n, n))
    for i in range(n):
        A[i, i] = n - i     # Diagonal principal
    for i in range(n-1):
        A[i, i+1] = n       # Diagonal superior
    return A
```

```python

# Exemplo
print(wilkinson_bidiagonal(5))

cond_numbers = []
orders = range(1, 16)

for n in orders:
    A = wilkinson_bidiagonal(n)
    cond = np.linalg.cond(A)
    cond_numbers.append(cond)

# Gráfico do número de condicionamento
plt.figure(figsize=(8,5))
plt.plot(orders, cond_numbers, marker='o')
plt.xlabel('Ordem da matriz (n)')
plt.ylabel('Número de condicionamento (cond)')
plt.title('Número de condicionamento da matriz bidiagonal de Wilkinson')
plt.grid(True)
plt.show()

# Matriz original
n = 20
A = wilkinson_bidiagonal(n)
eigvals_original = np.linalg.eigvals(A)

# Perturbação: soma 1e-10 ao elemento (20,1)
A_perturbed = A.copy()
A_perturbed[-1, 0] += 1e-10
eigvals_perturbed = np.linalg.eigvals(A_perturbed)

# Comparação dos autovalores
print("Autovalores originais:\n", np.sort(np.real(eigvals_original)))
print("\nAutovalores após perturbação:\n", np.sort(np.real(eigvals_perturbed)))
print("\nDiferença máxima:", np.max(np.abs(np.sort(np.real(eigvals_original)) - np.sort(np.real(eigvals_perturbed)))))

```

b) Gráfico do número de condicionamento das matrizes Wilkinson de ordem 1 a 15:

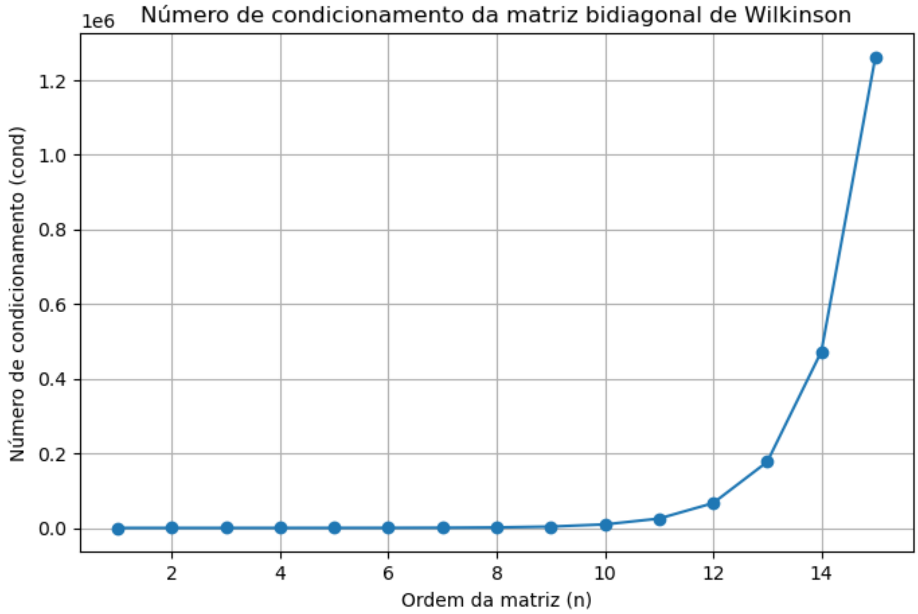

Figura 1: Número de condicionamento da matriz bidiagonal de Wilkinson para ordens de 1 a 15.

Conclusão: O gráfico mostra que o número de condicionamento cresce rapidamente com o aumento de n, tornando a matriz cada vez mais mal-condicionada para valores altos de n. Ou seja, pequenas perturbações nos elementos podem causar grandes variações nos resultados de operações com essas matrizes.

c) Autovalores e instabilidade:

```
Autovalores originais:
 [ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
 19. 20.]

Autovalores após perturbação:
 [ 0.99575439  2.10924185  2.57488136  3.96533065  3.96533065  5.89397763
  5.89397763  8.11807374  8.11807374 10.50000052 10.50000052 12.88192687
 12.88192687 15.10602214 15.10602214 17.03466871 17.03466871 18.42511789
 18.89075841 20.00424561]

Diferença máxima: 1.1060223749669227
PS C:\Users\suzi_\OneDrive\Área de Trabalho\Mestrado\Alg Lin Comp>
```

Figura 2: Autovalores após pertubações.

Comentário: Mesmo uma perturbação minúscula (10^{-10}) pode causar variações relevantes nos autovalores dessa matriz. Isso ilustra a instabilidade numérica: ainda que os autovalores sejam bem separados, o problema de autovalores pode ser mal-condicionado, e pequenos erros nos dados de entrada podem ser amplificados no resultado.

Resumo:

- A matriz Wilkinson bidiagonal é útil para testar algoritmos porque se torna rapidamente mal-condicionada com o aumento da ordem.
- O número de condicionamento cresce muito rápido com n, tornando operações numéricas instáveis para matrizes grandes.
- Uma perturbação muito pequena pode alterar significativamente os autovalores, mostrando que nem sempre autovalores bem separados garantem estabilidade numérica.