DCC639: Álgebra Linear Computacional **(Prazo para submissão: xx/03/2021 23:55)**

Lista de Exercícios 05

Professores: Erickson e Fabricio

Política da Disciplina: Leia todas as instruções abaixo cuidadosamente antes de começar a resolver a lista, e antes de fazer a submissão.

- As questões podem ser discutidas entre até três alunos (conjuntos disjuntos). Os nomes dos colegas precisam ser incluídos na submissão. Contudo, a escrita das soluções e submissão deve ser feita individualmente.
- A submissão deve ser feita em formato PDF através do Moodle, mesmo que tenham sido resolvidas a mão e escaneadas.
- Todas as soluções devem ser justificadas.
- Todas as fontes de material precisam ser citadas. O código de conduta da UFMG será seguido à risca.

Problema 1: Dada a matriz $A = \begin{bmatrix} 3 & 0 & 2 \\ 9 & 1 & 7 \\ 1 & 0 & 1 \end{bmatrix}$ **e sua inversa** $A^{-1} = \begin{bmatrix} 1 & 0 & -2 \\ -2 & 1 & -3 \\ -1 & 0 & 3 \end{bmatrix}$**, calcule o número de condição de A com relação a cada uma das normas a seguir. Neste exercício, vamos assumir que uma matriz cujo número de condição é maior que 10^6 é mal condicionada.**

(a) Norma-1. É bem condicionado?

(Solução) $\|A\|_1 = \max_{1\leq j\leq 3} \sum_{i=1}^{3} |a_{ij}| = \max(13, 1, 10) = 13$
$\|A^{-1}\|_1 = \max_{1\leq j\leq 3} \sum_{i=1}^{3} |a_{ij}| = \max(4, 1, 8) = 8$
$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1 = 13 \times 8 = 104$
$\kappa_1(A) < 10^6$ portanto não é mal condicionada.

(b) Norma-infinito. É bem condicionado?

(Solução) $\|A\|_\infty = \max_{1\leq i\leq 3} \sum_{j=1}^{3} |a_{ij}| = \max(5, 17, 2) = 17$
$\|A^{-1}\|_\infty = \max_{1\leq i\leq 3} \sum_{j=1}^{3} |a_{ij}| = \max(3, 6, 4) = 6$
$\kappa_\infty(A) = \|A\|_1 \|A^{-1}\|_1 = 17 \times 6 = 102$
$\kappa_\infty(A) < 10^6$ portanto não é mal condicionado.
OBS: é válido lembrar que o número de condicionamento varia com a norma utilizada, mas não pode ser muito discrepante. Isso pôde ser observado a partir dos exercícios anteriores.

Problema 2: Assinale V para verdadeiro ou F para falso e JUSTIFIQUE sua resposta

( ) Resolver um problema de quadrados mínimos linear usando a fatoração QR é menos custoso computacionalmente que resolver as equações normais.
( ) O número de condição de uma matriz depende da norma escolhida.
( ) O uso da decomposição QR não reduz o número de condição do sistema a ser resolvido para encontrar a solução do problema de quadrados mínimos linear.
( ) Na decomposição QR de uma matriz não-singular A, o determinante de A é igual ao determinante de R.

(Solução)
(F) Resolver um problema de quadrados mínimos linear usando a fatoração QR é menos custoso computacionalmente que resolver as equações normais.

Tanto a fatoração QR (no caso geral) quanto a resolução de um sistema linear tem custo $\mathbb{O}(n^3)$.
(V) O número de condição de uma matriz depende da norma escolhida.
O valor exato do número de condição depende sim da norma escolhida, como visto no Problema 1. No entanto, vale ressaltar que uma matriz com número de condição alto para uma determinada norma, não pode ter um número de condição baixo para outra.
(F) O uso da decomposição QR não reduz o número de condição do sistema a ser resolvido para encontrar a solução do problema de quadrados mínimos linear.
O sucesso da decomposição QR para este tipo de problema vem justamente do fato de não trabalharmos com a matriz $X^\top X$*, cujo número de condição é o quadrado daquele da matriz* X*. Por isso, a resolução via QR é mais estável numericamente.*
(F) Na decomposição QR de uma matriz não-singular A, o determinante de A é igual ao determinante de R.
O determinante de uma matriz ortogonal é +1 ou -1. Portanto, $\det(A) = \det(QR) = \det(Q)\det(R)$*, só implica que os determinantes de* A *e* R *são iguais em módulo.*

Problema 3: Considerando a matriz A abaixo, encontre sua decomposição QR usando o processo de Gram-Schmidt e depois aplique a primeira iteração do algoritmo QR. Quais os autovalores encontrados? Para facilitar a realização dos cálculos, considere até a segunda casa decimal dos resultados.

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 0 & 2 & 1 \\ 0 & 4 & 3 \end{bmatrix}$$

(Solução)
Os autovalores são 4.8, 1 e 0.2

Problema 4: Assinale V para verdadeiro ou F para falso e JUSTIFIQUE sua resposta

( ) A decomposição QR usa o algoritmo QR.
( ) Toda matriz converge para uma matriz triangular superior com o algoritmo QR.
( ) Na prática, deve-se evitar calcular $A^\top A$ para descobrir-se os valores singulares de A.
( ) A decomposição $A = QR$ é uma boa solução para o problema de se encontrar autovalores, pois com custo $\mathcal{O}(n^3)$, irá encontrar os autovalores de A na diagonal de R.

(Solução)
(F) A decomposição QR usa o algoritmo QR.
Pelo contrário, a decomposição QR é realizada a cada iteração do algoritmo QR.
(F) Toda matriz converge para uma matriz triangular superior com o algoritmo QR.
Em alguns casos é obrigatório utilizar-se Wilkinson shifts para que a matriz convirja.
(V) Na prática, deve-se evitar calcular $A^\top A$ para descobrir-se os valores singulares de A. ***Obs.:*** *Esta parte não foi dada na disciplina portanto podem desconsiderar esta alternativa. Porém, para quem interessar, a resposta é que o melhor é transformar* A *em uma matriz bidiagonal* B *e calcular-se os autovalores da matriz tridiagonal (portanto Hessenberg)* $T = B^\top B$.
(F) A decomposição $A = QR$ é uma boa solução para o problema de se encontrar autovalores, pois com custo $\mathcal{O}(n^3)$, irá encontrar os autovalores de A na diagonal de R.
A matriz A *e* R *não são similares. Portanto, os autovalores de* R *(elementos da sua diagonal) não são os autovalores de* A*, ou seja, a decomposição QR não é capaz de obter os autovalores de* A.

Problema 5: Calcule a inversa da matriz $A = \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix}$**. Dica: esta matriz pode ser escrita na forma** $(I - uv^\top)$ **para algum** u **e** v**.**

(Solução)
$(I - uv^T)^{-1} = I^{-1} + \frac{uv^T}{1-u^Tv}$

$$\left(I - \begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}1 & 1 & 1\end{bmatrix}\right)^{-1} = I^{-1} + \frac{uv^T}{-2}$$
$$= \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}\left(-\tfrac{1}{2}\right)\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$
$$= \begin{bmatrix}1/2 & -1/2 & -1/2\\-1/2 & 1/2 & -1/2\\-1/2 & -1/2 & 1/2\end{bmatrix}$$

Problema 6: Calcule a inversa da matriz $C = A - \begin{bmatrix}1 & -2 & 2\\2 & -4 & 2\\2 & -4 & 1\end{bmatrix}$, onde A é a matriz do problema 5. Dica 1: utilize a inversa A^{-1} anteriormente junto com as matrizes U e $V^\top$ apropriadas. Dica 2: $\begin{bmatrix}3/2 & -1/2\\1/2 & 5/2\end{bmatrix}^{-1} = \begin{bmatrix}0.625 & 0.125\\-0.125 & 0.375\end{bmatrix}$.

(Solução)

$$(A - UV^T)^{-1} = A^{-1} + A^{-1}U(I_K - V^TA^{-1}U)^{-1}V^TA^{-1}$$
$$V^TA^{-1}U = \begin{bmatrix}1 & -2 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1/2 & -1/2 & -1/2\\-1/2 & 1/2 & -1/2\\-1/2 & -1/2 & 1/2\end{bmatrix}\begin{bmatrix}1 & 2\\2 & 2\\2 & 1\end{bmatrix}$$
$$V^TA^{-1}U = (\tfrac{1}{2})\begin{bmatrix}1 & -2 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & -1 & -1\\-1 & 1 & -1\\-1 & -1 & 1\end{bmatrix}\begin{bmatrix}1 & 2\\2 & 2\\2 & 1\end{bmatrix}$$
$$V^TA^{-1}U = (\tfrac{1}{2})\begin{bmatrix}3 & -3 & 1\\-1 & -1 & 1\end{bmatrix}\begin{bmatrix}1 & 2\\2 & 2\\2 & 1\end{bmatrix}$$
$$V^TA^{-1}U = (\tfrac{1}{2})\begin{bmatrix}-1 & 1\\-1 & -3\end{bmatrix}$$
$$V^TA^{-1}U = \begin{bmatrix}-1/2 & 1/2\\-1/2 & -3/2\end{bmatrix}$$
$$I_2 - V^TA^{-1}U = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} - \begin{bmatrix}-1/2 & 1/2\\-1/2 & -3/2\end{bmatrix} = \begin{bmatrix}3/2 & -1/2\\1/2 & 5/2\end{bmatrix}$$
$$(I_2 - V^TA^{-1}U)^{-1} = \begin{bmatrix}0.62 & 0.12\\-0.12 & 0.38\end{bmatrix}$$
$$A^{-1}U(I_2 - V^TA^{-1}U)^{-1}V^TA^{-1} = \begin{bmatrix}-1.12 & 0.5 & -0.62\\-0.25 & 0.5 & -0.25\\0.12 & 0.5 & -0.38\end{bmatrix}$$
$$A^{-1} + A^{-1}U(I_2 - V^TA^{-1}U)^{-1}V^TA^{-1} = \begin{bmatrix}-0.62 & 1.0 & -1.12\\-0.75 & 1.0 & -0.75\\-0.38 & 0.0 & 0.12\end{bmatrix}$$

Problema 7: Na regressão linear, temos que se $X'X$ é inversível, os coeficientes são dados por $(X'X)^{-1}X'b$. Se $X'X$ não possui inversa, podemos usar a pseudo-inversa $V\Sigma^{-1}U'$. Prove que se $X'X$ possui inversa, ela e sua pseudo-inversa são iguais.

(Solução) Se ela possui inversa, então podemos obter essa inversa a partir da decomposição SVD de X:

$$X = U\Sigma V^\top \implies$$
$$(X^\top X)^{-1}X^\top = \left((U\Sigma V^\top)^\top(U\Sigma V^\top)\right)^{-1}(U\Sigma V^\top)^\top$$
$$= V(\Sigma^2)^{-1}V^\top(U\Sigma V^\top)^\top$$
$$= V(\Sigma^2)^{-1}V^\top V\Sigma U^\top$$

$$= V(\Sigma^2)^{-1}\Sigma U^\top$$
$$= V\Sigma^{-1}U^\top$$

Problema 8: Podemos obter a decomposição de Cholesky a partir de um algoritmo iterativo, que em cada iteração encontra uma coluna de L seguindo os seguintes passos:

- Obtém o elemento $L_{i,i}$ como $\sqrt{A[i,i]}$.
- Obtém os outros elementos da coluna (abaixo de $L_{i,i}$) a partir de coluna correspondente de A e $L_{i,i}$.
- Usa o produto externo para cancelar o efeito dessa coluna de L na matriz total, e depois disso descobre uma submatriz de tamanho $(n-1xn-1)$

Descreva com suas palavras ou esboce um pseudo-código para esse algoritmo iterativo. (Dica: lembre-se de como fazemos a multiplicação de matrizes por produtos externos, e depois use essa abordagem para observar como fazemos a multiplicaçao da decomposição de Cholesky: $A = LL'$.)

(Solução)

```
1   def chol(A):
2       ## a matriz L precisa ser do tipo float para tratar divisões não inteiras
3       L = np.zeros(shape=A.shape, dtype=np.float)
4       ## dimensão da matriz
5       n = L.shape[0]
6       ## i é o indice da coluna atual de L que estamos "descobrindo"
7       for i in range(n):
8           ## descobre o elemento na primeira linha/coluna
9           L[i, i] = np.sqrt(A[i, i])
10          ## descobre os outros elementos da coluna
11          for j in range(i + 1, n):
12              L[j, i] = A[j, i] / L[i, i]
13          ## subtrai o produto externo da coluna recém descoberta de A
14          col = L[:, i:(i + 1)]
15          A = A - (col @ col.T)
16
17      return L
```

Figura 1: Implementação do algoritmo iterativo que realiza a decomposição de Cholesky.

Problema 9: Assinale V para verdadeiro ou F para falso e JUSTIFIQUE sua resposta.

( ) A matriz de Vandermonde V $(n \times n)$ não é uma matriz em si, mas uma família de matrizes.
( ) Se dois elementos da 2a. coluna da matriz de Vandermonde são idênticos, a matriz é singular.
( ) A matriz de Vandermonde é a matriz X que aparece nas equações normais $X^\top X\beta = X^\top y$ quando fazemos uma regressão polinomial.
( ) A decomposição de Cholesky pode ser aplicada a qualquer matriz $A^\top A$ tal que as entradas de A são números reais.
( ) Seja A uma matriz cuja decomposição QR é $A = QR$. R^T é a matriz L da decomposição de Cholesky de $A^T A$.

(Solução)

`(V)` A matriz de Vandermonde V $(n \times n)$ não é uma matriz em si, mas uma família de matrizes. *Correto. Existem infinitas "instanciações" da matriz de Vandermonde, que vão depender da escolha de* $x_1, \ldots, x_n$.

`(V)` Se dois elementos da 2a. coluna da matriz de Vandermonde são idênticos, a matriz é singular *Se dois elementos da segunda coluna,* x_i *e* x_j*, são iguais, temos que as linhas* i *e* j *serão iguais, pois* $x_i^2 = x_j^2$, $x_i^3 = x_j^3$, *etc.*

`(V)` A matriz de Vandermonde é a matriz X que aparece nas equações normais $X^\top X\beta = X^\top y$ quando fazemos uma regressão polinomial. *Verdadeiro, pois cada coluna* i *de* X *possui os elementos elevados à potência de* i

`(F)` A decomposição de Cholesky pode ser aplicada a qualquer matriz $A^\top A$ tal que as entradas de A são números reais. *Aqui consideramos a decomposição de Cholesky como foi vista, sem pivotação. Toda matriz* A^TA *é simétrica. Pode-se mostrar também que* A^TA *é semidefinida positiva:* $v^TA^TAv = (Av)^TAv = (Av)(Av) = \|Av\|_2^2 \geq 0$. *Existem então dois casos possíveis: (i) todos os autovalores são maiores que 0 e, portanto, pode-se aplicar Cholesky a* A^TA *dado que ela é definida positiva; (ii) pelo menos um autovalor é igual a 0 e, portanto, Cholesky não se aplica (sem pivotação).*

`(V)` Seja A uma matriz cuja decomposição QR é $A = QR$. R^T é a matriz L da decomposição de Cholesky de A^TA. *Sendo* $A = QR$, *temos que* $A^TA = (QR)^TQR = R^TQ^TQR = R^TR$. *Considerando que* A^TA *pode ser decomposta via Cholesky, ou seja, é definida positiva, temos que* $A^TA = R^TR = LL^T \implies R^T = L$.

Problema 10: Considere o sistema linear $A\mathbf{x} = \mathbf{b}$, onde:

$$A = \begin{bmatrix} 5 & -2 & 1 \\ 3 & 4 & 0 \\ -1 & 2 & 3 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} 0.1818 \\ 0.3636 \\ 0.8181 \end{bmatrix}.$$

Considerando como solução inicial $\hat{\mathbf{x}}^{(0)} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top$:
(a) Resolva o sistema aplicando duas iterações do método Gauss-Jacobi.
(b) Resolva o sistema aplicando duas iterações do método Gauss-Seidel.
(c) Calcule qual o erro relativo e absoluto dos dois métodos na última iteração. Qual método está convergindo mais rapidamente? Qual deles está mais próximo do valor de referência $\mathbf{x}$?
(d) Os métodos usados irão convergir para um valor próximo do valor de referência após um certo número de iterações neste sistema linear? Justifique.

(Solução)

(c) *Podemos ver pelo output do programa na Figura 2 que o método Gauss-Seidel converge mais rápido e se aproxima mais rapidamente do valor de referência* $\mathbf{x}$

(d) *Como a matriz não é estritamente diagonal dominante, não podemos afirmar que os métodos Gauss-Jacobi e Gauss-Seidel irão convergir neste sistema, porém, na prática eles convergem.*

```python
import numpy as np
A = np.array([[5.,-2,1],
              [3,4,0],
              [-1,2,3]])
b = np.array([1.,2,3])

x = np.array([1.,1,1])

iters = 2

#METODO GAUSS-JACOBI
for i in range(iters):
  x_old = x.copy()
  x[0] = (1 + 2 * x_old[1] - 1 * x_old[2]) / 5
  x[1] = (2 -3 * x_old[0]) / 4
  x[2] = (3 + 1* x_old[0] - 2*x_old[1]) / 3

print("         Método Gauss-Jacobi")
print(x)
print("Erro Absoluto: %.4f" % (np.max(np.abs(x - x_old))))
print("Erro Relativo: %.4f" % (100. * np.max(np.abs(x - x_old)/np.abs(x_old))))

print("\n----------------------------------------\n")
print("         Método Gauss-Seidel")
#METODO GAUSS-SEIDEL
x = np.array([1.0,1,1])

for i in range(iters):
  x_old = x.copy()
  x[0] = (1 + 2 * x[1] - 1 * x[2]) / 5
  x[1] = (2 -3 * x[0]) / 4
  x[2] = (3 + 1* x[0] - 2*x[1]) / 3

print(x)
print("Erro Absoluto: %.4f" % (np.max(np.abs(x - x_old))))
print("Erro Relativo: %.4f" % (100. * np.max(np.abs(x - x_old)/np.abs(x_old))))
```

```
         Método Gauss-Jacobi
[-0.03333333  0.2         1.3       ]
Erro Absoluto: 0.6333
Erro Relativo: 180.0000 %

----------------------------------------

         Método Gauss-Seidel
[0.08       0.44       0.73333333]
Erro Absoluto: 0.3200
Erro Relativo: 120.0000 %
```

Figura 2: Implementação dos métodos Gauss-Jacobi e Gauss-Seidel para as alternativas (a) (b) e (c).