LISTA 1 Métodos iterativos para sistemas de equações lineares

*(As questões sinalizadas com **(**)** deverão ser entregues até o dia 15 de setembro)*

1. **(**)** Considere a matriz tridiagonal por blocos $A \in \mathbb{R}^{m^2 \times m^2}$

$$A = \begin{bmatrix} T & -I & & & & \\ -I & T & -I & & & \\ & -I & T & -I & & \\ & & -I & T & \ddots & \\ & & & \ddots & \ddots & -I \\ & & & & -I & T \end{bmatrix}$$

com I a matrix identidade de $m \times m$, e T a matriz

$$T = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{bmatrix} \in \mathbb{R}^{m \times m}$$

(*Essa matriz A aparece muito na solução numérica de EDP de tipo Poisson*)

   (a) O método SOR, com $0 < \omega < 2$, aplicado ao sistema $Ax = b$ ($b \in \mathbb{R}^{m^2}$) é convergente $\forall m \in \mathbb{N}$? Justifique bem sua resposta.

   (b) Uma boa escolha do parâmetro ω no método SOR pode levar a uma convergência mais rápida, comparado com Jacobi e Seidel. Em geral determinar o valor ótimo de ω não é um problema fácil, mas em alguns casos em que a matriz do sistema tem uma estrutura específica é possivel achar esse valor ótimo $\omega_{ótimo}$. Por exemplo, para a matriz A acima é conhecido que

$$\omega_{ótimo} = \frac{2}{1 + \sin\left(\frac{\pi}{m+1}\right)}$$

   Implemente o método SOR em Matlab para resolver o sistema $Ax = b$, onde b é um vetor tal que o sistema tem a solução exata $x^* = (2, 2, \ldots, 2, 2)$. Compare a performance do método SOR usando 4 valores diferentes do parâmetro ω: *i)* $\omega = \omega_{ótimo}$, *ii)* $\omega = 1$ (o método de Seidel), *iii)* $\omega = 0.5$, *iv)* $\omega = 0$, para 4 valores diferentes de m: $m = 50,\ 100,\ 1000,\ 5000$. Como criterio de comparação use o número de iterações necessárias para que o erro na norma ∞ seja $\leq 10^{-6}$. Comente sobre os resultados obtidos.

2. **(**)** Considere uma matriz $A \in \mathbb{R}^{n \times n}$ tal que os elementos na diagonal de A são todos 3, na subdiagonal inferior e superior são todos -1, e $A_{i,n+1-i} = \frac{1}{2}\ \forall i = 1, ..., n$ (excepto para $i = \frac{n}{2}$ e $i = \frac{n}{2} + 1$). Por exemplo, para $n = 4$

$$A = \begin{bmatrix} 3 & -1 & & \frac{1}{2} \\ -1 & 3 & -1 & \\ & -1 & 3 & -1 \\ \frac{1}{2} & & -1 & 3 \end{bmatrix}$$

Defina o vetor $b = \left(\frac{5}{2}, \frac{3}{2}, \cdots, \frac{3}{2}, 1, 1, \frac{3}{2}, \cdots, \frac{3}{2}, \frac{5}{2}\right)^\top$. Então a solução do sistema $Ax = b$ é $x^* = (1, 1, \cdots, 1, 1)$.

   (a) Implemente uma função Matlab para resolver o sistema $Ax = b$ usando o método de Jacobi e o método de Seidel. Considere como parâmetros de entradas: a dimensão n do sistema e a tolerancia ε. Considere como saída: a aproximação obtida com Jacobi, a obtida com Seidel e o número de iterações realizadas por cada método.

(b) Para $n = 100000$, itere os métodos acima até obter uma aproximação com 6 digitos de precisão usando Jacobi e Seidel. Para isto utilize como critério de parada o erro exato na norma infinita, isto é: $\left\|x^* - x^{(k)}\right\|_\infty$. Quantas iterações foram necessárias para cada método?

(c) Implemente em MATLAB o método do Gradiente Conjugado e, para $n = 100000$, determine o erro exato $\left\|x^* - x^{(k)}\right\|_\infty$ após k iterações, para $k = k_{jacobi}$ e $k = k_{seidel}$, onde k_{jacobi} e k_{seidel} são a quantidade de iterações que foram necessárias com Jacobi e Seidel no item b). Qual dos 3 métodos teve a melhor performance?

3. Escreva uma função MATLAB que, tendo como dados de entrada uma matriz A (diagonalmente dominante), um vetor b, e uma constante ε, utiliza o método de Jacobi e o método de Seidel para obter aproximacões da solução do SEL $Ax = b$ com uma presição ε.

(a) Teste seu programa com matrizes diagonalmente dominantes por linhas e por colunas.

(b) Teste seu programa com uma matriz tal que $\max(|\lambda_i|) < 1$, onde λ_i são os autovalores da matriz de iteração C, e tal que A não seja uma matriz diagonalmente dominante. Comprove que neste caso o método é convergente. Por que?

(c) Resolva o seguinte sistema de equações lineares com ao menos 5 dígitos de precição de cada x_i

$$\begin{aligned} x_1 - x_2 &= 0 \\ -x_{j-1} + 2.5x_j - x_{j+1} &= e^{-\frac{(i-3)^2}{20}} \qquad 2 \leq j \leq 4 \\ 2x_5 - x_4 &= 0 \end{aligned}$$

4. Considere o sistema Linear

$$\begin{pmatrix} 2 & 1 & -\frac{1}{2} \\ 1 & 2 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

(a) Está garantida a convergencia do método de Gauss-Seidel, independentemente da condição inicial escolhida?

(b) Com $x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ obtenha o vetor $x^{(1)}$ e calcule uma estimativa do error $\left\|x - x^{(1)}\right\|_\infty$.

5. Considere o seguinte sistema de equações lineares:

$$\begin{aligned} x_1 - x_2 &= 0 \\ -x_{j-1} + 5x_j - x_{j+1} &= \cos(\frac{j}{10}) \qquad 2 \leq j \leq 10 \\ x_{11} &= \frac{x_{10}}{2} \end{aligned}$$

(a) Construa a iteração para encontrar a solução deste problema pelos métodos de Jacobi e SOR com $\omega = 1$.

(b) Usando esses métodos, encontre uma solução aproximada com erro absoluto inferior a 10^{-5}.

6. Prove ou refute a seguinte afirmação: "Para toda matriz A simétrica definida positiva e qualquer vetor b, o método de Jacobi aplicado ao sistema $Ax = b$ converge, independentemente da iteração inicial". Justifique sua resposta.

7. Seja a matriz A simétrica definida positiva e b um vetor qualquer. Demonstre que e o método de Seidel aplicado ao sistema $Ax = b$ é convergente.

8. Seja a função quadrática $f(x) = \frac{1}{2}x^\top Ax - b^\top x$, onde $A \in \mathbb{R}^{m\times m}$ é uma matriz simétrica definida positiva e $b \in \mathbb{R}^{m\times 1}$

(a) Prove que: x^*minimiza $f(x) \iff Ax^* = b$ (*não precisa usar cálculo para provar esse resultado*)

(b) Use o método do Gradiente Conjugado para determinar o mínimo da função

$$f(x) = 2x^2 + 5y - 2yz - 8x - 2xy + 19 + 2y^2 - 6z + 2z^2$$

Observação: *Em algumas questões acima, em casos envolvendo matrizes de grandes dimensões e com muitos elementos iguais a zero, é recomendado trabalhar as matrizes como sparse. Isto permite armarzenar e realizar operações de forma muito mais económica sem a necessidade de reservar mémoria para guardar todos os elementos da matriz. Só aqueles elementos diferentes de zero são armazenados. Por exemplo: a matriz identidade de dimensão* 10000×10000 *quando constuida em MATLAB da forma padrão* $I = eye(10000)$*, usa* 800 *Megabytes de memória. Mas, após convertir* I *na forma sparse (*$I = sparse(I)$*) a mesma matriz usa apenas* 0.25 *Megabytes de memória. Pesquise ou use a ajuda do MATLAB para mais detalhes.*