FGV EMAp
ESCOLA DE MATEMÁTICA APLICADA

Aula Prática 2

Álgebra Linear Numérica

Gerardo Mikael Do Carmo Pereira

Professor: Antônio Carlos Saraiva Branco

RIO DE JANEIRO
2024

Conteúdo

1 Introdução

O presente trabalho refere-se à aula prática 2. Este conjunto de tarefas práticas concentra-se na implementação e análise de métodos iterativos para resolver sistemas lineares no Scilab. Iniciaremos com a implementação dos algoritmos de Jacobi e Gauss-Seidel, explorando diferentes abordagens para aprimorar a convergência. Testaremos essas implementações em vários sistemas lineares, observando seu desempenho e comportamento em diferentes cenários. Além disso, faremos uma análise comparativa do tempo de execução das implementações do Método de Gauss-Seidel em matrizes com diagonal estritamente dominante para diferentes tamanhos.

Ao longo dos exercícios teremos uma melhor compreensão prática dos métodos iterativos e suas aplicações na resolução de sistemas lineares.

2 Implementação

2.1 <Tarefa 1> Implementação do algoritmo iterativo de Jacobi

Dado um sistema linear $Ax = b$, podemos obter uma aproximação da solução x com o algoritmo iterativo de jacobi.

No método, o sistema linear é decomposto em uma matriz diagonal (D) e duas matrizes triangulares (L e U). Durante cada iteração, o método de Jacobi atualiza cada componente do vetor solução utilizando apenas os valores da iteração anterior.

Assim temos:

$$\mathbf{x}^{(k)} = \mathbf{D}^{-1} \cdot \left(\mathbf{b} - (\mathbf{L} + \mathbf{U}) \cdot \mathbf{x}^{(k-1)}\right)$$

$$\mathbf{x}^{(k)} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) \cdot \mathbf{x}^{(k-1)} + \mathbf{D}^{-1} \cdot \mathbf{b}$$

onde:

$\mathbf{x}^{(k)}$ é o vetor solução na k-ésima iteração,
$\mathbf{D}$ é a matriz diagonal de $\mathbf{A}$,
$\mathbf{L}$ é a matriz triangular inferior de $\mathbf{A}$,
$\mathbf{U}$ é a matriz triangular superior de $\mathbf{A}$,
$\mathbf{b}$ é o vetor dos termos independentes,
$\mathbf{x}^{(k-1)}$ é o vetor solução na $(k-1)$-ésima iteração.

Assim, $\mathbf{D}^{-1}$ representa a inversa da matriz diagonal $\mathbf{D}$, e o termo $(\mathbf{b} - (\mathbf{L} + \mathbf{U}) \cdot \mathbf{x}^{(k-1)})$ representa o resíduo do sistema, que é atualizado a cada iteração. O vetor solução $\mathbf{x}^{(k)}$ é obtido multiplicando o resíduo pelo inverso da matriz diagonal $\mathbf{D}$.

A função elaborada será apresentada nos arquivos anexados. Além disso discutiremos o desempenho e os resultados da implementação nos próximos exercícios.

2.2 <Tarefa 2> Implementação do algoritmo iterativo de Gauss-Seidel.

De forma semelhante dado um sistema linear $Ax = b$, podemos obter uma aproximação da solução x com o algoritmo iterativo de Gauss-Seidel. O processo é semelhante ao de Jacobi.

Mas desta vez, temos:

$$\mathbf{x}^{(k)} = (\mathbf{D} + \mathbf{L})^{-1}(\mathbf{b} - \mathbf{U} \cdot \mathbf{x}^{(k-1)})$$

onde:

$\mathbf{x}^{(k)}$ é o vetor solução na k-ésima iteração,
D é a matriz diagonal de **A**,
L é a matriz triangular inferior de **A**,
U é a matriz triangular superior de **A**,
b é o vetor dos termos independentes,
$\mathbf{x}^{(k-1)}$é o vetor solução na $(k-1)$-ésima iteração.

Faremos duas implementações distintas:

Uma das implementações utiliza a função "inv"do Scilab para calcular a inversa da soma das matrizes L e D. Com isso, obtemos a matriz $M_G = -(L+D)^{-1}U$ e o vetor $c_G = (L+D)^{-1}b$, que são utilizados nas iterações $x_{k+1} = M_G \cdot x_k + c_G$. A outra resolve o sistema linear $(L+D) \cdot x_{k+1} = -U \cdot x_k + b$ diretamente.

A semelhante ao algoritmo de Jacobi as funções elaboradas serão apresentadas nos arquivos anexados. Além disso, discutiremos o desempenho e os resultados da implementação nos próximos exercícios.

2.3 <Tarefa 3> Primeiros testes

Testaremos primeiramente as funções disponiveis com os seguintes sistema linear e o vetor aproximação inicial:

$$A = \begin{bmatrix} 1 & -4 & 2 \\ 0 & 2 & 4 \\ 6 & -1 & -2 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, x = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Usando como critério de parada $|x_{k+1} - x_k| < 1 * 10^{-6}$ ou 1000 iterações, além de usar norma 2 com as três funções teremos como resultado.

Algoritmo de Jacobi:

```
--> A = [1, -4, 2; 0, 2, 4; 6, -1, -2];

--> b = [2; 1; 1];

--> x0 = [0; 0; 0];

--> E = 1e-6;

--> M = 1000;

--> [xk_j, diff_norm_j, k_j, residual_norm_j] = jacobi_method(A, b, x0, E, M, 2);
```

```
 "Solução encontrada:"

--> disp(xk_j);

   Nan
   Nan
   Nan

--> disp("Norma da diferença entre as duas últimas aproximações:");

 "Norma da diferença entre as duas últimas aproximações:"

--> disp(diff_norm_j);

   Nan

--> disp("Número de iterações efetuadas:");

 "Número de iterações efetuadas:"

--> disp(k_j);

   1000.

--> disp("Norma do resíduo:");

 "Norma do resíduo:"

--> disp(residual_norm_j);

   Nan
```

Figura 1: Resultado obtido no console Método iterativo de Jacobi

Algoritmo de Gauss-Seidel:

```
--> [xk_gs, diff_norm_gs, k_gs, residual_norm_gs] = gauss_seidel(A, b, x0, E, M, 2);

--> disp("Solução encontrada:");

 "Solução encontrada:"

--> disp(xk_gs);

   Nan
   Nan
   Nan

--> disp("Norma da diferença entre as duas últimas aproximações:");

 "Norma da diferença entre as duas últimas aproximações:"

--> disp(diff_norm_gs);

   Nan

--> disp("Número de iterações efetuadas:");

 "Número de iterações efetuadas:"

--> disp(k_gs);

   1000.

--> disp("Norma do resíduo:");

 "Norma do resíduo:"

--> disp(residual_norm_gs);

   Nan
```

Figura 2: Resultado obtido no console Método iterativo de Gauss Seidel com inversa

```
--> [xk_gs2, diff_norm_gs2, k_gs2, residual_norm_gs2] = gauss_seidel_2(A, b, x0, E, M, 2);

--> disp("Solução encontrada:");

 "Solução encontrada:"

--> disp(xk_gs2);

   Nan
   Nan
   Nan

--> disp("Norma da diferença entre as duas últimas aproximações:");

 "Norma da diferença entre as duas últimas aproximações:"

--> disp(diff_norm_gs2);

   Nan

--> disp("Número de iterações efetuadas:");

 "Número de iterações efetuadas:"

--> disp(k_gs2);

   1000.

--> disp("Norma do resíduo:");

 "Norma do resíduo:"

--> disp(residual_norm_gs2);

   Nan
```

Figura 3: Resultado obtido no console Método iterativo de Gauss Seidel direta

Não obtemos valores como resposta, apenas "nan", valores indeterminados. Seguiremos a atividade e modificaremos a matriz A, permutando suas linhas de forma que ela tenha diagonal estritamente dominante. Assim aplicaremos a seguinte matriz nas funções:

$$A = \begin{bmatrix} 6 & -1 & -2 \\ 1 & -4 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

Algoritmo de Jacobi:

```
A2  =

  6.  -1.  -2.
  1.  -4.   2.
  0.   2.   4.

--> [xk_j, diff_norm_j, k_j, residual_norm_j] = jacobi_method(A2, b, x0, E, M, 2);

--> disp("Solução encontrada:");

 "Solução encontrada:"

--> disp(xk_j);

  0.4166666
 -0.0166668
  0.2583336

--> disp("Norma da diferença entre as duas últimas aproximações:");

 "Norma da diferença entre as duas últimas aproximações:"

--> disp(diff_norm_j);

  0.0000006

--> disp("Número de iterações efetuadas:");

 "Número de iterações efetuadas:"

--> disp(k_j);

  20.

--> disp("Norma do resíduo:");

 "Norma do resíduo:"

--> disp(residual_norm_j);

  0.0000016
```

Figura 4: Resultado obtido no console Método iterativo de Jacobi

Algoritmo de Gauss-Seidel:

```
--> [xk_gs, diff_norm_gs, k_gs, residual_norm_gs] = gauss_seidel(A2, b, x0, E, M, 2);

--> disp("Solução encontrada:");

 "Solução encontrada:"

--> disp(xk_gs);

   0.4166667
  -0.0166667
   0.2583334

--> disp("Norma da diferença entre as duas últimas aproximações:");

 "Norma da diferença entre as duas últimas aproximações:"

--> disp(diff_norm_gs);

   0.0000004

--> disp("Número de iterações efetuadas:");

 "Número de iterações efetuadas:"

--> disp(k_gs);

   13.

--> disp("Norma do resíduo:");

 "Norma do resíduo:"

--> disp(residual_norm_gs);

   0.0000004
```

Figura 5: Resultado obtido no console Método iterativo de Gauss Seidel com inversa

```
--> [xk_gs2, diff_norm_gs2, k_gs2, residual_norm_gs2] = gauss_seidel_2(A2, b, x0, E, M, 2);

--> disp("Solução encontrada:");

 "Solução encontrada:"

--> disp(xk_gst);

   0.2500002
  -0.4166659
   0.4583326

--> disp("Norma da diferença entre as duas últimas aproximações:");

 "Norma da diferença entre as duas últimas aproximações:"

--> disp(diff_norm_gst);

   0.0000009

--> disp("Número de iterações efetuadas:");

 "Número de iterações efetuadas:"

--> disp(k_gst);

   21.

--> disp("Norma do resíduo:");

 "Norma do resíduo:"

--> disp(residual_norm_gst);

   2.0883228
```

Figura 6: Resultado obtido no console Método iterativo de Gauss Seidel direta

Devido à matriz ter diagonal estritamente dominante, as implementações do método de Gauss-Seidel e jacobi devem convergir. Isso ocorre porque a estrita dominância garante que o elemento na diagonal principal de cada linha seja maior do que a soma dos elementos fora da diagonal nessa linha. Como resultado, ao atualizar cada componente da solução em cada iteração, o impacto dos elementos fora da diagonal é atenuado, permitindo que o método convirja para uma solução única.

No entanto, é importante ressaltar que essa condição de convergência é uma implicação e não garante a não convergência dada a negativa. As implementações podem não convergir devido a diferentes fatores.

2.4 <Tarefa 4> Testes com diferentes matrizes

Parte 1

Testaremos o algoritmo de jacobi com o seguinte sistema linear e vetor inicial:

$$A = \begin{bmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{bmatrix}, b = \begin{bmatrix} -1 \\ 4 \\ -5 \end{bmatrix}, x = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Com solução prevista:

$$x = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

```
--> A4 = [2 -1 1; 2 2 2; -1 -1 2];

--> b4 = [-1; 4; -5];

--> x0 = [0; 0; 0];

--> M = 25;

--> E = 1e-6;

--> [xk_j, diff_norm_j, k_j, residual_norm_j] = jacobi_method(A4, b4, x0, E, M, %inf);

-->

--> disp("Solução encontrada aproximada:");

 "Solução encontrada aproximada:"

--> disp(xk_j);

 -20.827873
  2.
 -22.827873

--> disp("Diferença das últimas duas normas:");

 "Diferença das últimas duas normas:"

--> disp(diff_norm_j);

  34.924597
```

Figura 7: Resultado obtido no console usando o Método iterativo de Jacobi

Obtemos tanto um valor muito diferente do esperado como aproximação da solução, quanto uma diferença muito alta das duas últimas iterações. Verificaremos se o problema é o número baixo de iterações, usemos então 1000 iterações como limite:

```
--> M = 1000;

-->

--> [xk_j, diff_norm_j, k_j, residual_norm_j] = jacobi_method(A4, b4, x0, E, M, %inf);

--> disp("Solução encontrada aproximada:");

 "Solução encontrada aproximada:"

--> disp(xk_j);

 -1.711D+48
 -6.843D+48
  1.711D+48

--> disp("Diferença das últimas normas:");

 "Diferença das últimas normas:"

--> disp(diff_norm_j);

  6.843D+48
```

Figura 8: Resultado obtido no console usando o Método iterativo de Jacobi e 1000 iterações

Mais uma vez temos resultados incongruentes, verificamos então que a matriz não tem diagonal estritamente dominante e o valor do módulo do maior autovalor (obtemos usando max(abs(spec(A4)))) é 3, logo maior que 1. Apesar de haver matrizes que convergem fora destes casos estas são as condições padrão que devem ser levadas em consideração, e nenhuma delas foi atendida.

Parte 2

Testando o mesmo sistema com o método de Gauss Seidel:

```
--> A4 = [2 -1 1; 2 2 2; -1 -1 2];

--> b4 = [-1; 4; -5];

--> x0 = [0; 0; 0];

--> M = 25;

--> E = 1e-6;

--> [xk_gs, diff_norm_gs, k_gs, residual_norm_gs] = gauss_seidel(A4, b4, x0, E, M, %inf);

--> disp("Solução encontrada aproximada:");

 "Solução encontrada aproximada:"

--> disp(xk_gs);

   1.0000006
   1.9999993
  -1.0000000

--> disp("Diferença das últimas normas:");

 "Diferença das últimas normas:"

--> disp(diff_norm_gs);

   0.0000020
```

Figura 9: Resultado obtido no console usando o Método iterativo de Gauss Seidel

Gauss Seidel convergiu com sucesso a solução para o sistema dado.

2.5 <Tarefa 5> Impacto da dominancia da diagonal na convergência

Parte 1

Usaremos o método de Gauss-Seidel para aproximar à solução apresentada, neste primeiro caso observamos que a solução é aproximada corretamente.

```
--> A5 = [1 0 -1; -0.5 1 -0.25; 1 -0.5 1];

--> b5 = [0.2; -1.425; 2];

--> E = 1e-2
 E  =

   0.01

--> M = 300
 M  =

   300.
```

```
--> [xk_gs, diff_norm_gs, k_gs, residual_norm_gs] = gauss_seidel(A5, b5, x0, E, M, %inf);

--> disp("Solução encontrada aproximada:");

 "Solução encontrada aproximada:"

--> disp(xk_gs);

   0.8975131
  -0.8018652
   0.7015543

--> disp("Diferença das últimas normas:");

 "Diferença das últimas normas:"

--> disp(diff_norm_gs);
```

Figura 10: Resultado obtido no console usando o Método iterativo de Gauss Seidel para a matriz dada

Já no segundo caso, com a modificação na matriz, não temos uma boa aproximação, como veremos a seguir:

```
--> A5 = [1 0 -2; -0.5 1 -0.25; 1 -0.5 1];

--> [xk_gs, diff_norm_gs, k_gs, residual_norm_gs] = gauss_seidel(A5, b5, x0, E, M, %inf);

--> disp("Solução encontrada aproximada:");

 "Solução encontrada aproximada:"

--> disp(xk_gs);

  2.157D+41
  1.348D+41
 -1.483D+41

--> disp("Diferença das últimas normas:");

 "Diferença das últimas normas:"

--> disp(diff_norm_gs);

  3.726D+41
```

Figura 11: Resultado obtido no console usando o Método iterativo de Gauss Seidel para a segunda matriz

A convergência do método de Gauss-Seidel não é garantida para todos os tipos de matrizes. Em particular, matrizes que não são diagonalmente dominantes podem apresentar problemas de convergência. No caso das duas matrizes analisadas:

$$A = \begin{bmatrix} 1 & 0 & -1 \\ -0.5 & 1 & -0.25 \\ 1 & -0.5 & 1 \end{bmatrix} \textbf{ e } A = \begin{bmatrix} 1 & 0 & -2 \\ -0.5 & 1 & -0.25 \\ 1 & -0.5 & 1 \end{bmatrix}$$

A segunda matriz não é diagonalmente dominante, pois em algumas linhas, o módulo do elemento diagonal não é maior que a soma dos módulos dos outros elementos da linha. Isso pode levar a problemas de convergência no método de Gauss-Seidel. Por outro lado, a primeira matriz é diagonalmente dominante, o que significa que o módulo do elemento diagonal em cada linha é maior que a soma dos módulos dos outros elementos da linha. Portanto, o método de Gauss-Seidel irá convergir para uma solução precisa.

2.6 <Tarefa 6> Impacto da variação do tamanho da matriz nas variações de métodos

Foram criadas duas funções para as duas variações da função de aproximação por Gauss-Seidel ambas recebem o tamanho da dimensão da array que terá diagonal estritamente dominante, retornando pontos que iremos analisar, a variação da norma da última iteração, o número de iterações e o tempo de execução da aproximação (Não é contado a geração da matriz). As matrizes tem como única diferença as variações que criamos na tarefa 2, e ambas serão adicionadas aos anexos que acompanharão o relatório.

Resultados obtidos:

Na primeira função, é calculada explicitamente a inversa da matriz $L + D$, e em seguida é usada para calcular a matriz do método $MG = -(L + D)^{-1}U$. Este cálculo da inversa pode ser computacionalmente custoso, especialmente para matrizes grandes,

Tabela 1: Resultados da aproximação da solução de sistemas lineares usando o método de Gauss-Seidel

Dimensão	Variação da Norma	Número de Iterações	Tempo de Execução (s)
10	9×10^{-7}	6	0.00118
100	4×10^{-7}	7	0.00642
1000	3×10^{-7}	6	1.654
2000	8×10^{-7}	5	20.91

Tabela 2: Resultados da aproximação da solução de sistemas lineares usando o método de Gauss-Seidel (segunda função)

Dimensão	Variação da Norma	Número de Iterações	Tempo de Execução (s)
10	2.3×10^{-6}	25	0.0163
100	7×10^{-7}	18	0.0703
1000	9×10^{-7}	12	0.6329
2000	7×10^{-7}	11	2.3671

mas uma vez que a matriz MG é calculada, as iterações subsequentes são computacionalmente simples, envolvendo apenas multiplicações de matriz-vetor.

Por outro lado, na segunda função, o sistema linear $(L + D) \cdot x_{k+1} = -U \cdot x_k + b$ é resolvido diretamente em cada iteração. Embora isso evite o custo computacional da inversão da matriz $L + D$, cada resolução do sistema linear pode ser computacionalmente mais intensiva, especialmente para matrizes grandes. No entanto, como o método de Gauss-Seidel é iterativo, o número de iterações necessárias pode ser maior para convergir para uma solução precisa.

Portanto, embora a segunda função possa ser mais rápida devido à eliminação do custo de calcular a inversa da matriz $L + D$, ela pode exigir mais iterações para alcançar a convergência, resultando em um maior número total de operações realizadas.