Resultados obtidos no 2º exercício computacional
Algoritmos numéricos I

Leandro Furlam Turi

17 de setembro de 2019

1 Preparativos

1.1 Implementações

1.1.1 Método de Jacobi

Algoritmo 1: Método de Jacobi

```
function [x, er, iter] = jacobi(A, b, tol, nmaxiter)
  [n, n] = size(A);
  x = zeros(n, 1);
  iter = 1;
  er(iter) = Inf;

  while ((iter <= nmaxiter) && (tol <= er(iter)))
    x0 = x;
    for i = 1:n
      aux = 0;
      for j = 1:(i-1)
         aux = aux + A(i, j)*x0(j);
      endfor
      for j = (i+1):n
         aux = aux + A(i, j)*x0(j);
      endfor
      x(i) = (1/A(i, i))*(b(i) - aux);
    endfor

    iter = iter + 1;
    er(iter) = norm(x - x0, inf) / norm(x, inf);
  endwhile
endfunction
```

1.1.2 Método SOR / Seidel

Algoritmo 2: Método SOR

```
function [x, er, iter] = sor(A, b, tol, nmaxiter, w)
  [n, n] = size(A);
  x = x_seidel = zeros(n, 1);
  iter = 1;
  er(iter) = Inf;
```

```
6
7   while ((iter <= nmaxiter) && (tol <= er(iter)))
8     x0 = x;
9     x0_seidel = x_seidel;
10    for i = 1:n
11      aux = 0;
12      for j = 1:(i-1)
13          aux = aux + A(i, j)*x_seidel(j);
14      endfor
15      for j = (i+1):n
16          aux = aux + A(i, j)*x0_seidel(j);
17      endfor
18      x_seidel(i) = (1/A(i, i))*(b(i) - aux);
19      endfor
20    x = w*x_seidel + (1-w)*x0;
21
22    iter = iter + 1;
23    er(iter) = norm(x - x0, inf) / norm(x, inf);
24  endwhile
25 endfunction
```

1.1.3 Matrizes dos métodos Jacobi, Seidel e SOR

Algoritmo 3: Matrizes dos métodos Jacobi, Seidel e SOR

```
1  function [MJ, MS, MSOR] = fatora(A, w)
2    D = diag(diag(A));
3    E = tril(A, -1);
4    F = tril(A, 1);
5
6    MJ = (-1)*inv(D)*(E + F);
7    MS = (-1)*inv(E + D)*F;
8    MSOR = inv(D + w*E)*((1-w)*D - w*F);
9  endfunction
```

1.1.4 Diagonal dominante

Algoritmo 4: Diagonal dominante

```
1  function [x] = diagonal_dominante(A)
2    [n, n] = size(A);
3    x = 1;
4    for i = 1:n
5    aux = 0;
6      for j = 1:(i-1)
7        aux = aux + abs(A(i, j));
8      endfor
9      for j = (i+1):n
10       aux = aux + abs(A(i, j));
11     endfor
12     if ((aux / abs(A(i, i))) >= 1)
13       x = 0;
14     endif
15   endfor
16 endfunction
```

1.1.5 Exercício computacional

Para realização do exercício, construiu-se um *script* único contendo todos os comandos e chamadas das funções necessárias. Em todos os casos a quantidade máxima de iterações utilizada foram 1000, e a tolerância de 10E-5.

1.2 Obtenção das matrizes

A escolha das matrizes esparsas utilizadas neste exercício computacional se deu única e exclusivamente observando o fato de elas serem de números reais, quadradas e inversíveis, com exceção das matrizes de ordem igual ou superior a 10E4, onde foi necessário buscar matrizes com menor quantidade de termos não nulos para que todas as operações de ponto flutuante ocorressem de maneira viável.

2 Resultados

Como houve um acréscimo da quantidade de operações de ponto flutuante, fez-se necessárias alguns ajustes conforme a ordem da matriz fosse aumentando. Dessa forma, a análise se dará de acordo com a ordem da matriz.

2.1 Matrizes de ordem 10E1 (48 x 48)

Diagonal dominante	Método	W	Raio espectral	Iterações	erro_rel (min)
Não	SOR	0.1	0.827535	1000	0.000518
		0.2	0.652416	1000	0.000511
		0.25	0.563773	1000	0.000509
		0.5	0.107750	1000	0.000507
		0.75	0.591433	1000	0.000506
		1.25	1.513447	0	Inf
		1.5	2.016182	0	Inf
	Jacobi		1.276983	0	Inf
	Seidel		1.008855	0	Inf

Como nota-se pela Tabela 2.1, a matriz não era diagonal dominante, e portanto não operou-se com os métodos de Jacobi e Seidel. para valores de W menores que 1 (método de Seidel), pode-se operar. É interessante ressaltar que o raio espectral não mostrou-se demasiadamente ligado a qualidade da solução.
Como neste caso não foram utilizados os métodos de Jacobi e Seidel, não faria sentido uma análise geométrica do problema.

2.2 Matrizes de ordem 10E2 (115 x 115)

Como a maior parte das matrizes dos métodos não possuíam raio espectral menor que 1 e tampouco eram diagonal dominante, foi feita a alteração descrita no Algoritmo 5, apenas nos testes realizados nas matrizes de ordem 10E2, tornando a mesma pelo menos diagonal dominante, e assim garantir a convergência. Assim, mesmo que o raio espectral fosse menor que 1, aplicou-se os métodos de Jacobi e Seidel.

Algoritmo 5: Tornando a matriz diagonal dominante

```
load(arq);
A = Problem.A;
[n, n] = size(A);
Afull = full(A);
xsum = sum(Afull);
for i = 1:n
  if (xsum(1, i) == 0)
    xsum(1, i) = 1;
  endif
  A(i, i) = xsum(1, i);
endfor
```

Diagonal dominante	Método	W	Raio espectral	Iterações	erro_rel (min)
Sim	SOR	0.25	0.513369	37	0.000008
		0.5	0.021973	17	0.000008
		0.75	0.522528	10	0.000004
		1.25	1.500000	12	0.000003
		1.5	2.000000	21	0.000007
		1.75	2.500000	48	0.000009
	Jacobi		1.061476	21	0.000007
	Seidel		1.000000	8	0.000002

2.2.1 Análise gráfica

Por coincidência, o valor de W que apresentou os melhores resultados nos testes analisados foi 1, equivalente ao método de Seidel. Com isso, a análise gráfica do método SOR será dada com o segundo melhor valor de W, igual a 0.75. A Figura 1 apresenta a comparação entre os métodos.

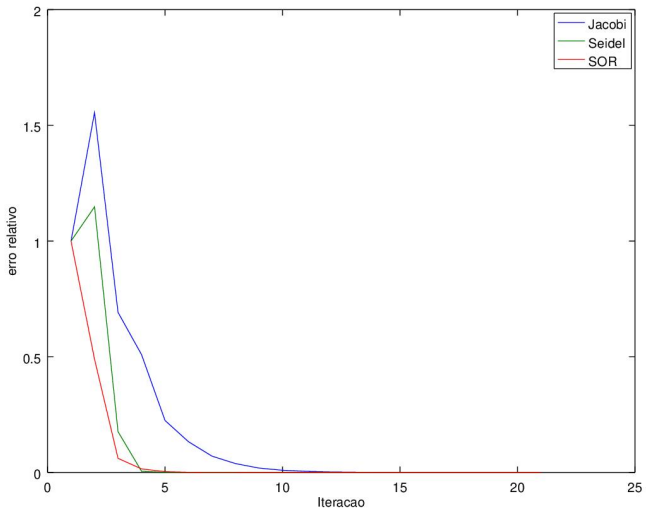

Figura 1: Comparação entre resultados correspondentes.

Através da Figura 1 nota-se que embora o método SOR possua mais iterações que o método de Seidel, o decaimento produzido no erro relativo é mais veloz e sempre ocorre, fato este que não é notado nos métodos de Seidel e principalmente no método de Jacobi, onde há picos antes de iniciar o decaimento.

2.3 Matrizes de ordem 10E3 (1138 x 1138)

Neste caso tornar a matriz diagonal dominante passou a ser uma operação muito custosa, tanto de memória quanto de quantidades de operações. Por isso trabalhou-se com a matriz em sua forma original e com menos variações de W.

Diagonal dominante	Método	W	Raio espectral	Iterações	erro_rel (min)
Não	SOR	0.5	0.638768	1000	0.000046
		1.5	2.034382	0	Inf
	Jacobi		2.411332	0	Inf
	Seidel		1.020217	0	Inf

2.4 Matrizes de ordem 10E4 e 10E5

Nestes casos não foi possível rodar o *script*, pois as matrizes analisadas ou atingiam o limite de memória do *Octave*, ou ultrapassaram 2 dias processando. Isto pode ser explicado pela quantidade de operações de ponto flutuante, além do tamanho das estruturas auxiliares necessárias para a análise conjunta dos métodos.

3 Conclusões

Com os testes realizados pode-se comparar a eficácia dos métodos iterativos para resolução de sistemas lineares, que estão diretamente ligados aos autovalores das matrizes e sua dominância diagonal. Pode-se concluir através dos testes realizados para a matriz de ordem 10E2 que o fato de o raio espectral da matriz do método ser maior que 1 não se mostrou um problema, visto que pode-se observar a convergência de ambos os métodos. Ainda, os métodos se mostraram custosos, uma vez que a partir de ordens 10E3 exigiu-se maiores esforços computacionais.
Quando comparados os resultados entre os métodos, o que se mostrou mais eficiente e aplicável foi o método SOR, já que através do seu fator de relaxamento pode-se alterá-lo de forma a tornar o raio espectral da matriz menor que 1, mesmo que a mesma não seja diagonal dominante, podendo garantir, assim, a convergência do método.