UNIVERSIDADE DE SÃO PAULO – USP
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

RELATÓRIO 1
SME 104 – CÁLCULO NUMÉRICO

Alunos: Adams Vietro Codignotto da Silva - 6791943
Ana Clara Kandratavicius Ferreira - 7276877

São Carlos
2014

Introdução

Em Cálculo Numérico utilizamos métodos iterativos para resolver sistemas lineares de grande porte. Um desses métodos é o de Gauss-Seidel, que é derivado do método de Jacobi, porém com uma velocidade de convergência maior. Neste trabalho, estaremos implementando o método Gauss-Seidel, que nos garante a convergência independentemente da aproximação inicial $x^{(0)}$ atribuída.

Modelagem do problema

Assumimos que as matrizes de entradas satisfaçam a condição de que:

- A matriz A seja **diagonal estritamente dominante por linhas ou colunas**

e portanto, não foi feita a verificação da mesma.

Partindo do método iterativo de Gauss-Seidel para resolver o sistema linear Ax=b, como visto em aula:

$$x_i^{(k+1)} = (D - L)^{-1}(Ux^{(k)} + b)$$

onde x é o vetor de iteração, D é a matriz diagonal de A, L e U são as matrizes da decomposição LU de A e b é a matriz solução do sistema, podemos reescrever fórmula como:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}}$$

Prova da Convergência

Ainda podemos provar que o método iterativo converge, qualquer que seja a aproximação inicial $x^{(0)}$:

Se $\rho(C) < 1$, onde $C = -(L + D)^{-1}U$, tomando $\epsilon = 1 - \rho(C) > 0$, sabemos que existe uma norma $\|.\|$ tal que $\|C\| < \rho(C) + \epsilon = 1$.

Resta ver que $\rho(C) < 1$ é condição necessária.

Supondo que exista um valor próprio z tal que $|z| \geq 1$, então se $x^{(0)} = x - v$, onde x é a solução e v o vetor próprio associado a z, obtemos $e^{(k)} = C^k e^{(0)} = C^k v = z^k v$, e como v está fixo e $|z|^k \geq 1$, então $e^{(k)}$ não tende a zero, não havendo convergência para o $x^{(0)}$ definido.

Experimentos e Resultados

O trabalho foi implementado utilizando o software Matlab R2013a, e realizamos os seguintes testes:

- Dado um sistema linear Ax=b, para uma matriz A pentadiagonal da forma

$$\begin{cases} a_{i,i} = 4 & i = 1, 2, \cdots, n \\ a_{i,i+1} = a_{i+1,i} = -1 & i = 1, 2, \cdots, n-1 \\ a_{i,i+3} = a_{i+3,i} = -1 & i = 1, 2, \cdots, n-3 \\ a_{i,j} = 0 & \text{para o restante} \end{cases} \tag{1}$$

  e o termo independente b da forma

$$b_i = \sum_{j=1}^{n} a_{ij}, \text{ para } i = 1, \cdots, n$$

  Utilizamos como parâmetros n=50 e n=100, erro máximo como 10^{-6} e um limite de 1500 iterações para resolver o sistema. Em ambos os casos, partimos da aproximação inicial $x^{(0)} = 0$. Para o caso n=50, após 557 iterações, obtivemos um vetor x com um resultado muito próximo ao esperado (onde $x_i = 1$, para i=1,...,n).

  Porém, para o caso n=100, 1500 iterações não foram suficientes para a convergência do método, sendo necessárias 1846 iterações para que houvesse a convergência utilizando o mesmo fator de erro. O resultado obtido para n=100 e 1846 iterações está próximo ao esperado como aconteceu com n=50.

- Utilizando uma matriz A definida pela equação (1) e um vetor b onde:

$$b_i = \frac{1.0}{i} \quad i = 1, 2, \cdots, n$$

para n=40, com erro de 10^{-6} e um limite de 1500 iterações, partindo da aproximação inicial $x^{(0)} = 0$, obtemos a convergência dentro de 389 iterações.

Para mostrar que a solução obtida é aceitável, fez-se a seguinte comparação: tomando $\overline{B} = A\overline{x}$ em que $\overline{x}$ é a solução numérica obtida, podemos calcular $\|\overline{B} - B\|_\infty$ em que $B = Ax$ sendo x a solução exata do sistema. O resultado é 1.9687×10^{-6}, um erro baixo, ou seja, uma boa aproximação da solução exata do sistema.

Conclusões

O método de Gauss-Seidel é excelente para aproximar a convergência da solução de sistemas lineares, além de ser fácil de ser implementado. Porém, a velocidade de convergência ainda é muito lenta para que o método seja usado na prática, levando $O(en^2)$ iterações para reduzir o erro por um fator de 10^e de um sistema linear $(n \times n)$ com matrizes de diagonais estritamente dominantes.

Se utilizarmos a fórmula geral $x^{(k+1)} = (D-L)^{-1}(Ux^{(k)}+b)$, apesar de ser de ainda mais fácil implementação, nos leva a algumas condições necessárias, como a matriz D nem sempre poder ser invertível, e $(D^{-1}L)$ deve ter raio espectral menor que 1. Se o raio espectral for maior que 1, as iterações podem nos levar a um limite infinito.

Implementação do método Gauss-Seidel Iterativo

```
function C = GaussSeidel(A,B,n,EPS,ITMAX)
X=zeros(n,1);
iteracoes=0;
erro=1000;

    while erro>EPS && iteracoes<ITMAX
       XOld=X; %X_j = X_i
       for i=1:n %percorrendo todos os x_i's
        T=Soma1(A,X,i); %somatório de j=1 a i-1 de a_ij x_j^{k+1}
        U=Soma2(A,XOld,i,n); %somatório de j=i+1 a n de a_ij x_j^{k}
        X(i)=(B(i)- T - U)/A(i,i);
       end
       erro=NormaInfinita(XOld,X,n); %calcula erro da iteração
       iteracoes=iteracoes+1;
    end

    if iteracoes==ITMAX  %caso chegou ao limite de iterações, divergiu
        fprintf('DIVERGE!\n');
    else
        fprintf('\nQuantidade de Iterações:\n');
        disp(iteracoes);
        fprintf('Converge para:\n');
        C=X
    end
end

function [soma1] = Soma1(A,x_0,i) %calcula primeiro somatório do método
soma1=0;
    for j=1:i-1
        soma1 = soma1 + (A(i,j)*x_0(j));
    end
end

function [soma2] = Soma2(A,x_0,i,n)  %calcula segundo somatório do método
soma2=0;
    for j=i+1:n
        soma2 = soma2 + (A(i,j)*x_0(j));
    end
end

function [max] = NormaInfinita(xk,xk1,n) %calcula norma infinita dados
                                         %dois vetores xk e xk1, de tamanho n

    x=xk1-xk;
    max=0;
    for i=1:n
            soma=abs(x(i,1));
        if(soma>max)
            max=soma;
        end
    end
end
```

Funções Auxiliares

Criar matriz pentadiagonal

```
function [m] = Pentadiagonal(n) %n é tamanho da matriz
    [m]=zeros(n);
    for i=1:n
        m(i,i)=4;
    end

    for i=1:n-1
        m(i,i+1)=-1;
        m(i+1,i)=-1;
    end

    for i=1:n-3
        m(i,i+3)=-1;
        m(i+3,i)=-1;
    end
end
```

Criar Vetor B (teste 1)

```
function [b] = VetorB(A, n)
    b=zeros(n,1);
    for i=1:n
        for j=1:n
        b(i,1)=b(i,1) + A(i,j);
        end
    end
end
```

Criar Vetor B (teste 2)

```
function [B] = VetorB2(n)
    B=zeros(n,1);
    for i=1:n
        B(i,1) = 1.0/i;
    end
```