*Trabajo práctico 3*

*Materia*: Métodos en Computación Científica

*Alumno*: Duarte Daniela LU: 88187

*Fecha*: 12 de septiembre de 2013

***Ejercicio 1***

Ingresamos la matriz A y el vector de soluciones b. Hallamos la descomposición LU de A usando la función predefinida lu(A) y procedemos a resolver el sistema.

>> A=[-4 1 1 0;1 -4 0 1;1 0 -4 1;0 1 1 -4]

A =

-4 1 1 0

1 -4 0 1

1 0 -4 1

0 1 1 -4

>> B=[-200;-400;0;-200]

B =

-200

-400

0

-200

>> [L,U]=lu(A)

L =

1.0000 0 0 0

-0.2500 1.0000 0 0

-0.2500 -0.0667 1.0000 0

0 -0.2667 -0.2857 1.0000

U =

-4.0000 1.0000 1.0000 0

0 -3.7500 0.2500 1.0000

0 0 -3.7333 1.0667

0 0 0 -3.4286

A partir de esta factorización resolvemos, para esto utilizamos el comando de MATLAB z=linsolve(L,B) con el resultado resolvemos x=linsolve(U,z) y en x obtenemos el resultado final.

>> z=linsolve(L,B)

z =

-200.0000

-450.0000

-80.0000

-342.8571

>> x=linsolve(U,z)

x =

100.0000

150.0000

50.0000

100.0000

***Ejercicio 2***

Para resolver el sistema utilizando el método SOR se decidió implementar la siguiente función en MATLAB:

%A -> la matriz de coeficientes

%B -> el vector de respuestas del sistema de ecuaciones

%max\_iter -> la maxima cantidad de iteraciones

%w -> omega (Factor)

function [Resultado,iteraciones] = SOR(A,B,maxIt,w,tol\_error)

%acá saco la longitud del vector de respuestas del sistema de ecuaciones

n = length(B);

%armo los vectores donde almaceno los resultados parciales del algoritmo

Resulado = zeros(n,1);

old=Resul; % Guardo el vector de la iteracion anterior para calcular el error

iteraciones=0; % contador de iteraciones

for k=1:maxIt

for i = 1:n

suma = B(i) - A(i,[1:i-1,i+1:n])\*Resultado([1:i-1,i+1:n]);

Resultado(i) = (w\*(suma / A(i,i)))+((1-w)\*Resultado(i));

end

error=norm(Resultado-old);

if ( error <= tol\_error )

break

end

old=Resultado;

iter=iter+1;

end

end

Luego se cargan los datos en MATLAB:

>> A=[4 -2 1;1 5 -3;1 1 5];

>> B=[11;-6;7];

>> epsilon= 10.^-5;

>> [r,i]=sor(A,B,20,1,epsilon)

r =

2.0081

-0.8952

1.1774

i =

10

El valor de óptimo de omega puede ser calculado con la siguiente función:

function w = optimo(A, b)

n = size(A,1); % hallamos L, U and D de A

D = diag(diag(A));

L = tril(-A,-1);

U = triu(-A,1);

Tj = inv(D)\*(L+U); % hallamos la matriz de iteraciones de Jacobi

radesp\_Tj = max(abs(eig(Tj))); % hallamos el radio espectral de Tj

w = 2./(1+sqrt(1- radesp \_Tj^2)); % hallamos el omega óptimo

Para nuestro sistema aplicamos esta función en MATLAB:

>> optimo(A,b)

ans =

1.0715

***Ejercicio 3***

a) Para resolver el sistema utilizando el método de Jacobi se decidio implementar la siguiete función en MATLAB:

%A -> la matriz de coeficientes

%B -> el vector de respuestas del sistema de ecuaciones

%maxIt -> la maxima cantidad de iteraciones

function [Resultado,iteracion] = jacobi(A,B,maxIt,max\_error)

%calulo la longitud del vector de respuestas del sistema de ecuaciones

n = length(B);

%inicializo el vector de resultado

Resultado = zeros(n,1);

ResulNew = Resultado;

iteracion = 0;

for k=1:maxIt

for i = 1:n

suma = B(i) - A(i,[1:i-1,i+1:n])\*Resultado([1:i-1,i+1:n]);

ResulNew(i) = suma/A(i,i);

end

error = norm(abs(Resultado-ResulNew));

if(error <= max\_error)

break

end

Resultado = ResulNew;

iteracion = iteracion+1;

end

end

Luego se cargan los datos en MATLAB:

>> A=[9 4 1; 1 6 0; 1 -2 -6];

>> b=[-17 4 14]';

>> epsilon= 10.^-5;

>> [Resultado,iteracion]=jacobi(A,b,20,epsilon)

Resultado =

-2.0000

1.0000

-3.0000

iteracion =

9

b) Para resolver el sistema utilizando el método de Gauss-Siedel se decidió implementar la siguiente función en MATLAB:

%A -> la matriz de coeficientes

%B -> el vector de respuestas del sistema de ecuaciones

%maxIter -> la maxima cantidad de iteraciones

function [Resultado,iteracion] = gausssiedel(A,B,maxIter,max\_error)

%calculo la longitud del vector de respuestas del sistema de ecuaciones

n = length(B);

%inicializo vectores donde almaceno los resultados parciales del algoritmo

Resultado = zeros(n,1);

ResulOld = Resul;

iteracion = 0;

for k=1:maxIter

for i = 1:n

suma = B(i) - A(i,[1:i-1,i+1:n])\*Resultado([1:i-1,i+1:n]);

Resultado(i) = suma / A(i,i);

end

error = norm(abs(Resultado-ResulOld));

if(error <= max\_error)

break

end

ResulOld = Resultado;

iteracion = iteracion+1;

end

end

>> epsilon= 10.^-5;

>> [Resulado,iteraciones]=gausssiedel(A,b,20,epsilon)

Resultado =

-2.0000

1.0000

-3.0000

iteraciones =

5

C) La convergencia del resultado se alcanza casi en la mitad de iteraciones utilizando el método de Gauss-Siedel