|  |
| --- |
| Métodos de Computación Científica - 2014 - |
| **Trabajo Práctico N° 3: “Resolución de Sistemas Lineales”**  **GARAT, Fabiana Yamel - LU 89108 -** |
|  |
|  |



**Ejercicio 1-** Encuentre por descomposición LU (usando para esto funciones de Matlab) la solución del siguiente sistema de ecuaciones:



**Solución** **–** Primeramente ingresamos la matriz A la consola de Matlab

>> 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

De igual manera procedemos a ingresar el vector b

>> b = [-200 -400 0 -200]

b =

-200 -400 0 -200

Y, por último, calculamos la descomposición LU de A como

>> [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

Dado el sistema Ax = b podemos descomponer a la matriz de coeficientes A en su descomposición LU de tal manera que:

Ax = b

LUx = b

Llamemos z a Ux y sustituyamos en la ecuación anterior:

Ux = z

Lz = b

A continuación resolvemos el siguiente sistema:

Lz = b

Obteniendo el vector solución de ese sistema. Por último con el vector z obtenido en el paso anterior, calculamos el vector solución x del sistema Ux = z.

>> z = b/L

z =

-328.5714 -457.1429 -57.1429 -200.0000

>> z = inv(L)\*b'

z =

-200.0000

-450.0000

-80.0000

-342.8571

>> x = inv(U)\*z

x =

100.0000

150.0000

50.0000

100.0000

**Ejercicio 2:** Resuelva este sistema por método SOR:



También encuentre el valor óptimo del factor de relajación *w*

**Solución –** Obtengo una función en Matlab para hallar la aproximación de la solución “x” aplicando el método SOR.

function [ x ] = SOR( A, b, xViejo, w, tol, maxIt )

% Inputs:

% A matriz de coeficientes del sistema lineal (CUADRADA!!)

% b vector del lado derecho del sistema lineal.

% xViejo vector que contiene los valores iniciales para la solución del

% sistema lineal.

% w parámetro de relajación.

% tol tolerancia de convergencia. Aplicada a la máxima norma

% de la diferencia entre las sucesivas aproximaciones.

% maxIt cantidad máxima de iteraciones a realizar.

%

% Output:

% x solución aproximada del sistema lineal.

n = length(b);

[f c] = size(A);

if (c ~= n)

disp ('error: dimensiones incompatibles')

return

end;

xNuevo = zeros(1,n);

for k = 1:maxIt

xNuevo(1) = (1 - w) \* xViejo(1) + w \* (b(1) - sum(A(1,2:n) .\* xViejo(2:n))) / A(1,1)

for i = 2:(n-1)

xNuevo(i) = (1 - w) \* xViejo(i) + w \* (b(i) - sum(A(i,1:i-1) .\* xNuevo(1:i-1)) - sum(A(i,i+1:n) .\* xViejo(i+1:n))) / A(i,i)

end;

xNuevo(n) = (1 - w) \* xViejo(n) + w \* (b(n) - sum(A(n,1:n-1) .\* xNuevo(1:n-1))) / A(n,n)

convergencia = max( abs( xNuevo - xViejo ) );

if(convergencia < tol)

x = xNuevo;

return

else

xViejo = xNuevo;

end;

end

x = xNuevo;

end

Una vez que contamos con esta función, ingresamos los datos por consola de la siguiente manera:

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

A =

4 -2 1

1 5 -3

2 2 5

>> b = [11 -6 7]

b =

11 -6 7

>> x0 = [0 0 0]

X0 =

0 0 0

Una vez cargados los datos y la función, llamamos a la función SOR con los siguientes parámetros:

* La matriz de coeficientes A
* EL vector b (del lado derecho del sistema lineal)
* W=1(Gauss-Siedel)
* Y un máximo de 10 iteraciones.

>> Rta = SOR(A,b,x0,1,eps,10)

De esta llamada obtenemos la siguiente salida

k =

1

xNuevo =

2.7500 0 0

xNuevo =

2.7500 -1.7500 0

xNuevo =

2.7500 -1.7500 1.0000

k =

2

xNuevo =

1.6250 -1.7500 1.0000

xNuevo =

1.6250 -0.9250 1.0000

xNuevo =

1.6250 -0.9250 1.1200

k =

3

xNuevo =

2.0075 -0.9250 1.1200

xNuevo =

2.0075 -0.9295 1.1200

xNuevo =

2.0075 -0.9295 0.9688

k =

4

xNuevo =

2.0431 -0.9295 0.9688

xNuevo =

2.0431 -1.0273 0.9688

xNuevo =

2.0431 -1.0273 0.9937

k =

5

xNuevo =

1.9879 -1.0273 0.9937

xNuevo =

1.9879 -1.0014 0.9937

xNuevo =

1.9879 -1.0014 1.0054

k =

6

xNuevo =

1.9980 -1.0014 1.0054

xNuevo =

1.9980 -0.9964 1.0054

xNuevo =

1.9980 -0.9964 0.9994

k =

7

xNuevo =

2.0020 -0.9964 0.9994

xNuevo =

2.0020 -1.0008 0.9994

xNuevo =

2.0020 -1.0008 0.9995

k =

8

xNuevo =

1.9997 -1.0008 0.9995

xNuevo =

1.9997 -1.0002 0.9995

xNuevo =

1.9997 -1.0002 1.0002

k =

9

xNuevo =

1.9998 -1.0002 1.0002

xNuevo =

1.9998 -0.9998 1.0002

xNuevo =

1.9998 -0.9998 1.0000

k =

10

xNuevo =

2.0001 -0.9998 1.0000

xNuevo =

2.0001 -1.0000 1.0000

xNuevo =

2.0001 -1.0000 1.0000

res =

2.0001 -1.0000 1.0000

Vemos claramente que para la iteración 10, el método converge para la solución correcta.

xRes2 = SOR(A,b,x,1.4322,eps,1000) solución correcta

xRes3 = SOR(A,b,x,1.432**3**,eps,1000) solución con error pequeño

xRes3 =

2.0001 -1.0000 0.9999

xRes4 = SOR(A,b,x,1.5,eps,1000) solución con error

xRes 4=

1.0e+043 \*

0.8106 0.4751 -1.3053

Para hallar un w optimo, el mismo debe ser tal que minimicemos el radio espectral de la matriz de iteración del método SOR. Para tal objetivo planteo una función que halla la matriz de iteración del método SOR y que l mimo tiempo calcule el radio espectral:

function [ r ] = radioEspectralSOR( A, w)

% Obtiene la descomposición de A en D, L y U

D = diag(diag(A));

L = tril(A) - D;

U = triu(A) - D;

% Matriz de iteración SOR

M = inv(D + w\*L)\*(-w\*U + D\*(1 - w));

% calcula el radio espectral como el mayor autovalor en valor absoluto de la

% matriz de iteración M.

radioEspectral = max( abs( eig( M ) ) );

r = radioEspectral;

end

Con esta nueva herramienta calculamos el w de tal manera que sea optimo

function [ wOpt] = wOptimo( A )

%inicializa el valor mínimo en el peor caso

%inicializa wOpt en un valor inválido para detectar errores.

min = 2;

wOpt = -1;

% Calcula wOpt con 4 decimales de precisión

% prueba por fuerza bruta valores para w entre 0.0001 y 1.9999

% con pasos de 0.0001

for w = 0.0001:0.0001:1.9999

r = radioEspectralSOR(A,w);

if(r < min)

min = r;

wOpt=w;

end;

end

disp('w optimo:');

disp(wOpt);

end

Comprobamos el funcionamiento mediante el uso de Matlab, y obtenemos un w óptimo. Los resultados se devuelven como pares [r,w] donde r el radio y w el óptimo.

>> wOptimo(A)

ans =

0.9183 0.1000

ans =

0.8362 0.2000

ans =

0.7538 0.3000

ans =

0.6713 0.4000

ans =

0.5888 0.5000

ans =

0.5073 0.6000

ans =

0.4282 0.7000

ans =

0.3567 0.8000

ans =

0.3118 0.9000

ans =

0.3464 1.0000

**Ans =**

**0.3103 0.9178**

ans =

0.4647 1.1000

ans =

0.6133 1.2000

ans =

0.7722 1.3000

ans =

0.9364 1.4000

ans =

1.1043 1.5000

ans =

1.2752 1.6000

ans =

1.4489 1.7000

ans =

1.6251 1.8000

ans =

1.8036 1.9000

**ans =**

**0.9178**

Para no hacer extenso el informe, y ya que no es el objetivo del mismo, no se muestran todas las iteraciones. Fue realizado con valores de w pertenecientes a [0.0001, 1.9999] saltando de a 0.0001. Obteniéndose el mismo resultado.

Esta función de obtención del w no resulta eficiente, ya que a mayor precisión en el resultamos mayor será la cantidad de iteraciones.

**Ejercicio 3 –** Determine la ecuación cuadrática que pasa a través de los puntos (1,1), (2,3) y (3,5)

**Solución –** Conformo un sistema de ecuaciones, que se obtiene al reemplazar los puntos dados en la ecuación

Luego

Entonces en de donde se despeja Por ultimo despejamos , de donde se despeja

Finalmente la ecuación cuadrática es