CÁLCULO NUMÉRICO

Resolución de SEAL

Trabajo Práctico Entregable N°1

Giuliana Cagnola

23/04/2023

Ejercicio 1:

A) Dado el sistema de ecuaciones algebraicas lineales

$$\begin{cases} x_1 = 0 \\ -x_{i-1} + 3x_i = \dfrac{1}{N^2}, i = 2,3,\dots,N-1 \\ x_N = 0 \end{cases}$$

La matriz A es una matriz tridiagonal (es decir, posee 3 diagonales: la diagonal principal, una por arriba y una por debajo, y el resto de los elementos de la matriz son 0).

```
 1   0   0   0   0   0   0   0   0   0
-1   2  -1   0   0   0   0   0   0   0
 0  -1   2  -1   0   0   0   0   0   0
 0   0  -1   2  -1   0   0   0   0   0
 0   0   0  -1   2  -1   0   0   0   0
 0   0   0   0  -1   2  -1   0   0   0
 0   0   0   0   0  -1   2  -1   0   0
 0   0   0   0   0   0  -1   2  -1   0
 0   0   0   0   0   0   0  -1   2  -1
 0   0   0   0   0   0   0   0   0   1
```

Modelo de la matriz A para N=10

Los métodos directos (Gauss vectorizado en este caso) e iterativos (Jacobi, Gauss-Seidel y SOR) permiten buscar una solución para nuestro sistema matricial **Ax=b**.

Los algoritmos de resolución para **x** son los siguientes:

- **Función para el método de Jacobi:**

```
function [x, r_h ,it ,t] = Jacobi(A, b, x0, Kmax, tol)
    tic(); #arranca a contar el tiempo
    n=length(b); #n toma la longitud de b
    x = x0;
    it=1;
    while(it<=Kmax) #lazo while para las iteraciones
        for i=1:n #lazo for desde 1 hasta n con paso 1
            x(i) =(b(i) - A(i,1:i-1)*x0(1:i-1) - A(i,i+1:n)*x0(i+1:n))/A(i,i) #iteración de Jacobi
        endfor
        r_h(it+1) = norm(A*x  -  b)  #la variable r_h toma la norma de la diferencia entre
A*x y b
```

```
        if r_h(it+1) < tol #si es menor entonces asumo convergencia
            break;
        endif
        x0 = x; #actualiza el vector de valores
        it = it + 1; #aumenta el contador de iteraciones
    endwhile  #finaliza el lazo while
    t=toc(); #finaliza el contador de tiempo
endfunction
```

- **Función para el método de Gauss-Seidel:**

```
function [x,r_h,it,t] = GaussSeidel(A,b,x0,maxit,tol)
    tic();
    n = length(b);
    x = x0;
    it = 1;
    while ( it <= maxit )
        for i = 1:n
            x(i) = (b(i) - A(i,1:i-1) * x(1:i-1) - A(i,i+1:n) * x0(i+1:n) ) /A(i,i);  #iteración de
GS
        endfor
        r_h(it+1) = norm(A*x - b);
        if r_h(it+1) < tol
            break;
        endif
        x0 = x;
        it = it +1;
    endwhile
    t = toc();
endfunction
```

- **Función para el método SOR (Sobre-Relajaciones Sucesivas):**

```
function [x,r_h,it,t] = SOR(A,b,x0,maxit,tol,w)
    tic();
    n = length(A(1,:));
    x = x0;
    it = 1;
    while (it <= maxit)
        for i = 1:n
            x(i) = (1-w) * x0(i) + w *( b(i) - A(i,1:i-1)*x(1:i-1)- A(i,i+1:n)*x0(i+1:n) ) /A(i,i);
```

```
#iteración de SOR
        endfor
        r_h(it +1) = norm(A*x - b);
        if r_h(it +1) < tol
            break;
        endif
        x0 = x;
        it = it +1;
    endwhile
    t = toc();
endfunction
```

- **Función para eliminación gaussiana:**

```
function [x,t] = Gauss(A,b)
    tic();
    n=length(b);
    x=b*0;
    for i=1:1:n
        m=A(i+1:n,i)/A(i,i); #multiplicador
        A(i+1:n,i:n)=A(i+1:n,i:n)-m*A(i,i:n);
        b(i+1:n)=b(i+1:n)-m*b(i);
    endfor
    [x]=SustitucionInversa(A,b);
    t = toc();
endfunction
```

- **Función para sustitución inversa**

```
function [x,t] = SustitucionInversa (A, b)
    n = length(b);
    x(n) = b(n)/A(n,n);
    for i=n-1:-1:1
        x(i)=(b(i)-A(i,i+1:n)*x'(i+1:n))/A(i,i);
    endfor
endfunction
```

- **Función para crear el sistema de ecuaciones:**

```
function [A,b,x0] = CrearSistema (N)
    A = (2*diag(ones(1,N),0)-1*diag(ones(1,N-1),1)-1*diag(ones(1,N-1),-1));
    A(1,[1:2])=[1 0];
    A(N,[N-1:N])=[0 1];
    b = ones(N,1);
    b(1) = 0;
    b(N) = 0;
    b = [b(1) ; (1/N^2).*ones(N-2,1); b(N)];
    x0 = zeros(N,1);
endfunction
```

- **Función para descomponer la matriz de manera aditiva A=L+D+S:**

```
function [L D U]=DescomponerMatriz(A)
    L=tril(A,-1);
    U=triu(A,1);
    D=diag(diag(A));
endfunction
```

- **Script principal:**

```
N = 100;
[A,b,x0] = CrearSistema (N);
maxit = 10000;
tol = 1e-6;
[L D U]=DescomponerMatriz(A);
[xJ, r_hJ, itJ, tJ]=Jacobi(A, b, x0, maxit, tol); #Jacobi
resp_J = max(abs(eig(-inv(D)*(L+U)))) # radio espectral
[xGS,r_hGS,itGS,tGS] = GaussSeidel(A,b,x0,maxit,tol); #Gauss-Seidel
resp_GS = max(abs(eig(-inv(D+L)*U))) # radio espectral
sim = issymmetric(A); #devuelve 1 o 0 según si es simétrica
defPos = all(eig(A)>0); # si es 1 todos los eig > 0.
w = 2/(1+sqrt(1-respJ^2)); #parametro de relajación
[xSOR,r_hSOR,itSOR,tSOR] = SOR(A,b,x0,maxit,tol,w); #SOR
resp_SOR = max(abs(eig(inv(D+w*L)*((1-w)*D-w*U)))) # radio espectral
[xG,tG] = Gauss(A,b); #Gauss
```

B)

Los radios espectrales asociados a cada método son:

Jacobi: resp_J = 0,9995

Gauss-Seidel: resp_GS = 0,9990

SOR: resp_SOR = 0.9385.

En este último método, el parámetro de relajación ω que optimiza la convergencia está dado por $\omega = \frac{2}{1+\sqrt{1-\rho(Tj)^2}}$, donde ρ(Tj) es el radio espectral de la matriz de transición del método de Jacobi. En este caso, ω = 1.9385.

Como todos los radios espectrales son <1, por teorema se sabe que entonces los métodos convergen. Sin embargo, por el método de SOR esta convergencia será más rápida, ya que el radio espectral es menor.

Para el método de Jacobi y Gauss-Seidel también podemos analizar convergencia a partir de los criterios del método. En este caso la matriz es diagonal dominante, lo cual implica convergencia.

Mediante la función **all(eig(A)>0)** se verifica que todos sus autovalores son positivos, por lo que también satisface dicho criterio para Gauss-Seidel.

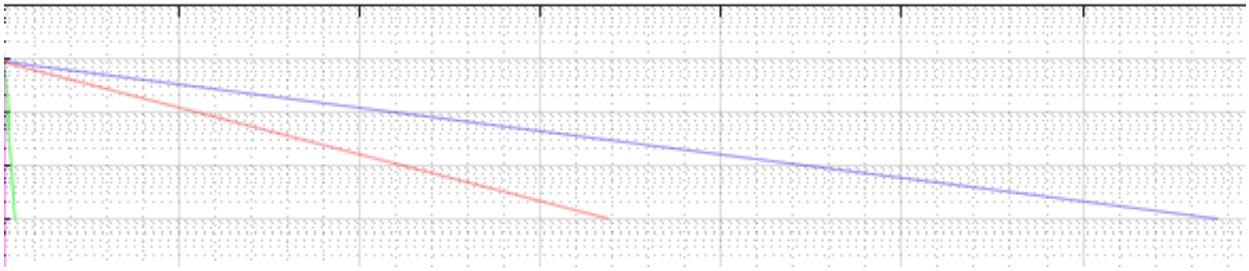
La cantidad de iteraciones para cada método con tolerancia 1e-6 son:

Jacobi: 13499

Gauss-Seidel: 6751

SOR: 173

Esto se condice con los valores de los radios espectrales. Como era de suponer, SOR es el método de convergencia más rápido (0,43 seg) ya que es el de radio espectral menor. Para Jacobi y Gauss-Seidel se obtienen 31 seg y 15,2 seg respectivamente.



La recta azul grafica el historial del residuo de Jacobi, la roja Gauss-Seidel y la verde SOR. En todos los casos, la ecuación $\mathbf{Ae^{(k)} = r^{(k)}}$ relaciona al error e con el residuo r obtenido en la k-esima iteración. Cuando e->0 => r->0.

Ejercicio 2:

Al pedirle a ChatGPT un SEAL de 5 ecuaciones que converja por Jacobi pero que no lo haga por Gauss-Seidel, la IA proporciona la siguiente respuesta:

$$\begin{cases} 2x + y + z + w = 5 \\ x + 3y + z - w = 10 \\ 2x + y + 5z + 2w = 2 \\ x - y + 2z + 7w = 3 \\ 3x + 4y + z - w = -1 \end{cases}$$

Pero como este sistema tiene más ecuaciones (5) que incógnitas (4), es compatible indeterminado, por lo que la IA falló o interpretó mal la consigna solicitada. La hipótesis es que se le pidió un sistema de 5 ecuaciones, pero nunca se le dijo que debía tener 5 incógnitas o que debía ser compatible determinado.

Al pedirle nuevamente un sistema de 5 ecuaciones con 5 incógnitas el SEAL propuesto es:

$$\begin{cases} 2x + y + z + u + v = 5 \\ x + 2y + z + u + v = 5 \\ x + y + 2z + u + v = 5 \\ x + y + z + 2u + v = 5 \\ x + y + x + u + 2v = 5 \end{cases}$$

Podemos reescribir el sistema como Ax=b:

$$\begin{bmatrix} 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ u \\ v \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}$$

Como det(A)=6 ≠0, entonces el sistema tiene solución única. Además, con la función **n=rank(A)** se obtiene n=5, lo cual coincide con la dimensión de la matriz. Cualquiera de estas afirmaciones implica a la otra, y también implican que el único vector x que satisface **Ax=0** es **x=0**. Si cargamos la matriz en Octave y resolvemos utilizando el comando **Xoct=A\b** para el vector de términos independientes **b=0**, efectivamente se obtiene la solución nula.

La solución por método directo para el sistema propuesto por ChatGPT es

$$\begin{bmatrix} x \\ y \\ z \\ u \\ v \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}$$

Como A no es diagonal dominante, no podemos usar el criterio de convergencia para Jacobi, pero si podemos ver que los elementos de la diagonal principal de $D^{-1}A$ son 1, por lo que podemos asegurar convergencia mediante dicho criterio.

Para la convergencia por Gauss-Seidel podemos ver que la función **all(eig(A)>0)** da 1, por lo que se verifica que todos sus autovalores son >0, y también corroboramos que es simétrica con el comando **issymmetric(A)**. Luego, esta es definida positiva, criterio suficiente para asegurar convergencia. Además, el radio espectral de la matriz de iteración es ρ(A)= 0.5437, por lo que el sistema converge.

Podemos ver que la IA proporcionó una respuesta incorrecta, ya que lo que se le pidió era un sistema de 5x5 que no converja por Gauss-Seidel, pero el sistema que proporcionó si lo hace.

Adicionamente, en otro chat se le mostró la matriz y se le preguntó si convergía. Su respuesta fue que no convergía por ningún método iterativo (si por eliminación gaussiana), por lo que podemos ver los errores e inconsistencias que tiene la IA a la hora de efectuar ciertas operaciones matemáticas un poco más complejas.