Trabajo Práctico nº1:
Métodos de resolución de SEAL

CONSIGNA: Dado el siguiente sistema de ecuaciones lineales:

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

(a) Realice un script que resuelva el sistema para N = 100, utilizando los métodos de Jacobi, Gauss-Seidel, SOR, gradiente conjugado y eliminación de Gauss.

(b) Determine el número de iteraciones necesarias para cada método iterativo, considerando una cota para el residuo de 1e−6. Determine, para el método de SOR, un parámetro de relajación ω óptimo. ¿Todos los métodos convergen? Justifique y grafique el historial del residuo para cada método.

(c) Suponiendo que la solución obtenida corresponde a una función y = x(t) evaluada en N puntos uniformemente distribuidos en el intervalo [0, 1], graficar la solución y = x(t) obtenida con cada método, y saque conclusiones.

RESOLUCIÓN:

En cuanto al inciso (a), el script y las funciones implementadas para resolver el sistema se anexan al final del trabajo a forma de facilitar la lectura del informe.

Contestando al inciso (b), se pueden observar los siguientes resultados de todos los métodos (directos en amarillo e iterativos en azul):

Método utilizado	Radio espectral	Número de iteraciones	Tiempo de ejecución (s)
Jacobi	0.9995	13499	30.273
Gauss-Seidel	0.9990	6751	15.150
SOR (w = 1.9385)	0.9385	173	0.4249
Gradiente Conjugado	-	49	5.8422e-03
Gauss Vectorizado	-	-	6.0868e-03

Ver que los métodos más eficientes para nuestro sistema en particular son los de gradiente conjugado y Gauss vectorizado, siendo los que menos iteraciones y tiempo de ejecución han precisado para realizar el cálculo. Los métodos directos son más efectivos ya que se está manejando un sistema de ecuaciones bastante sencillo, con muchos ceros en la matriz A.

En cuanto al cálculo del parámetro de relajación ω óptimo para el método de SOR, se sabe por teorema que, si la matriz A es definida positiva y tridiagonal, entonces $\omega = \frac{2}{1+\sqrt{1-[\rho(T_j)]^2}}$ es la selección óptima, donde $\rho(T_j)$ es el radio espectral de la matriz de transición del método de Jacobi.

Luego, en el programa determinamos si la matriz A es definida positiva o no al verificar el signo de sus eigenvalores. Tras obtener que todos son positivos, fue posible afirmar que es definida positiva.

Por otro lado, para saber si es tridiagonal es suficiente con observar la forma de la matriz, ya que los únicos términos distintos de 0 se encuentran en la diagonal principal, la diagonal por encima de ella y la diagonal por debajo. Ver, en la Figura 1, el resultado de CrearSistema(N) con N = 10 a forma de visualizar un modelo más pequeño de la matriz con la que se trabajó (N = 100).

```
 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
```

Figura 1: Modelo minimizado de la matriz A.

Todos los métodos iterativos convergen ya que sus radios espectrales son menores que 1, condición necesaria y suficiente. Ver que tanto el tiempo de ejecución como el número de iteraciones bajan a medida que el radio espectral baja: mientras más pequeño el radio espectral, más rápido convergerá el método.

A partir de los historiales del residuo obtenidos r_h, se realiza la gráfica de convergencia de cada método (ver Figura 2). La curva azul está dada por el historial de residuo de Jacobi, la roja por Gauss-Seidel, la verde por SOR y la magenta por el método de gradiente conjugado.

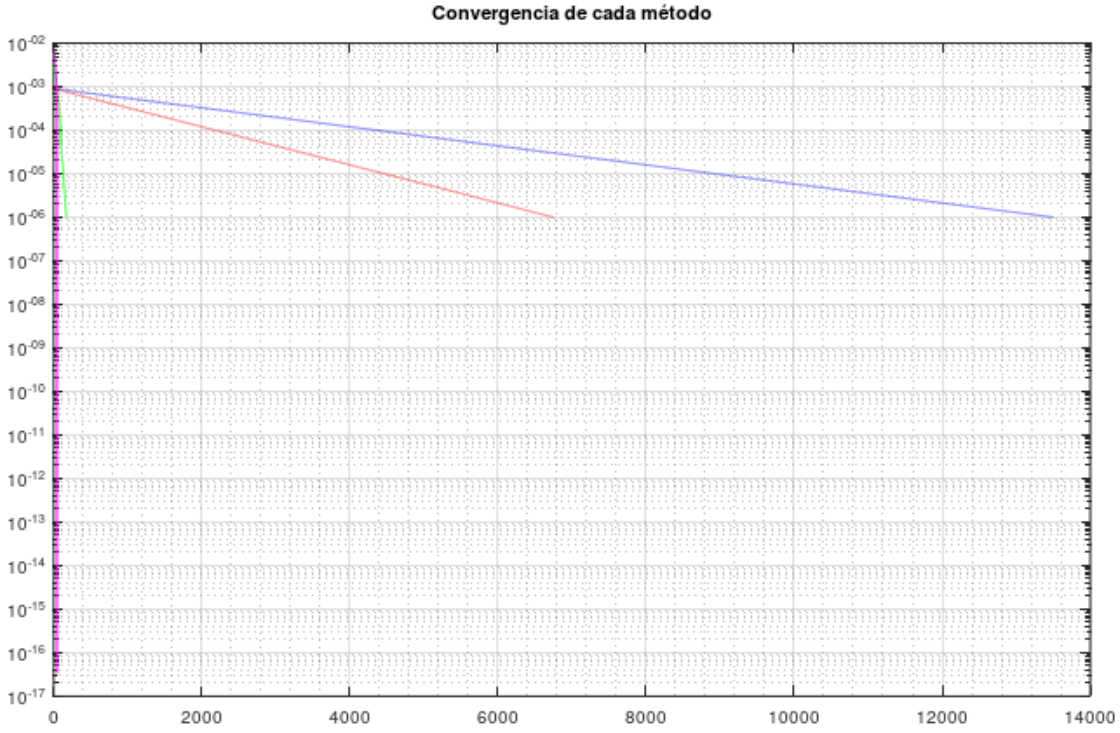

Figura 2: Convergencia de cada método.

En la Figura 2, se puede ver que los métodos iterativos aproximan a la solución con un error de 10^{-6} (cota dada por el ejercicio), Gauss con un valor similar y Gradiente Conjugado con error de 10^{-17} aproximadamente, siendo el más exacto.

Por último, el inciso (c) precisa de la gráfica de las soluciones del sistema con cada uno de los métodos. Ver figura 3.

Como todos los métodos aproximan a la solución pero a distintas velocidades por lo explicado al responder el inciso (b), y como la cota de error dada es pequeña y el máximo de iteraciones es alto, serán indistinguibles las variaciones entre las soluciones sin acercar la gráfica considerablemente.

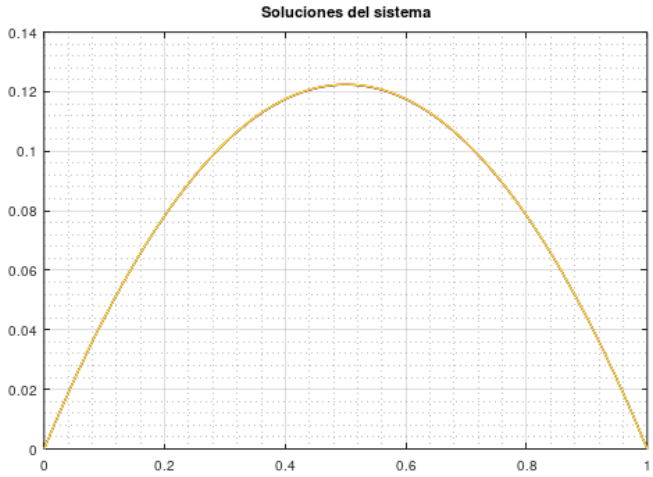

Figura 3: Gráfica de la solución del sistema con todos los modelos.

ANEXO DE ALGORITMOS

1. SCRIPT PRINCIPAL:

```
N = 100;
[A,b,x0] = CrearSistema (N);
maxit = 20000;
tol = 1e-6;

[L D U]=DescomponerMatriz(A);

disp("JACOBI:")
[xJ, r_hJ, itJ, tJ]=Jacobi(A, b, x0, maxit, tol);
r_eJ = max(abs(eig(-inv(D)*(L+U)))) # radio espectral
itJ # cantidad de iteraciones
tJ # tiempo de ejecución

disp("")
disp("GAUSS-SEIDEL:")
[xGS,r_hGS,itGS,tGS] = GaussSeidel(A,b,x0,maxit,tol);
r_eGS = max(abs(eig(-inv(D+L)*U))) # radio espectral
itGS # cantidad de iteraciones
tGS # tiempo de ejecución

disp("")
disp("SOR:")

# verificando s.d.p.
tf = issymmetric(A);
d = eig(A);
isposdef = all(d > 0); # resulta en 1, luego es verdadero y todos los eig > 0.
w = 2/(1+sqrt(1-r_eJ^2));

[xSOR,r_hSOR,itSOR,tSOR] = SOR(A,b,x0,maxit,tol,w);
r_eSOR = max(abs(eig(inv(D+w*L)*((1-w)*D-w*U)))) # radio espectral
itSOR # cantidad de iteraciones
tSOR # tiempo de ejecución
```

```
disp("")
disp("GRADIENTE CONJUGADO:")
[xGC, r_hGC, itGC, tGC] = GradienteConjugado(A, b, x0, tol);
itGC # cantidad de iteraciones
tGC # tiempo de ejecución

disp("")
disp("ELIMINACIÓN DE GAUSS:")
[xG,tG] = Gauss(A,b);
tG # tiempo de ejecución

#GRÁFICA DE CONVERGENCIA DE CADA MÉTODO:
figure(1)
semilogy(r_hJ,'-b')
grid on
grid minor
hold on
semilogy(r_hGS,'-r')
semilogy(r_hSOR,'-g')
semilogy(r_hGC,'-m')
title('Convergencia de cada método')
hold off

#GRÁFICA DE LA SOLUCIÓN DEL SISTEMA
z = linspace(0,1,N);
figure(2)
plot(z,xJ,'-b',z,xGS,'-r',z,xSOR,'-g',z,xGC,'-m',z,xG,'-y')
grid on
grid minor
title('Soluciones del sistema')
hold off
```

2. FUNCION 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
```

3. FUNCIÓN PARA DESCOMPONER LA MATRIZ DE MANERA ADITIVA:

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

4. FUNCIÓN PARA MÉTODO DE JACOBI:

```
function [x,r_h,it,t]=Jacobi(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)*x0(1:i-1)-A(i,i+1:n)*x0(i+1:n))/A(i,i);
   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
```

5. FUNCIÓN PARA 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);
    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
```

6. FUNCIÓN PARA MÉTODO SOR:

```
function [x,r_h,it,t] = SOR(A,b,x0,maxit,tol,w)
  tic();
  n = length(A(1,:));
  x = x0; # debe inicializarse x
  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);
    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
```

7. FUNCIÓN PARA MÉTODO GRADIENTE CONJUGADO:

```
function [x, r_h, it, t] = GradienteConjugado(A, b, x0, tol)
 tic();
 x=x0;
 n = length(b);
 r = b - A*x;
 p = r;
 rho = r'*r;
 rho0 = r;
 it = 1;

 while(it <= n)
 if norm( p ) < tol
 break;
 endif
 a = A * p;
 m = p' * a;
 alfa = rho/m;
 x = x + alfa * p;
 r = r - alfa * a;
 rho0 = rho;
 rho = r'*r;
 r_h(it) = norm(r,2);
 if r_h(it) < tol
 break;
endif
 gamma = rho/rho0;
 p = r + gamma*p;
 it = it + 1;
 endwhile
 t = toc();
endfunction
```

8. FUNCIÓN PARA MÉTODO GAUSS VECTORIZADO:

```
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);
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]=Sust_Atras(A,b);
t = toc();
endfunction
```

9. FUNCIÓN PARA SUSTITUCIÓN HACIA ATRÁS:

```
function [x,t] = Sust_Atras (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);
end
endfunction
```