UNL - FICH

Cálculo numérico - Entregable 1

Centis Mateo

Enunciado

Dado el siguiente sistema de ecuaciones lineales:

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

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.

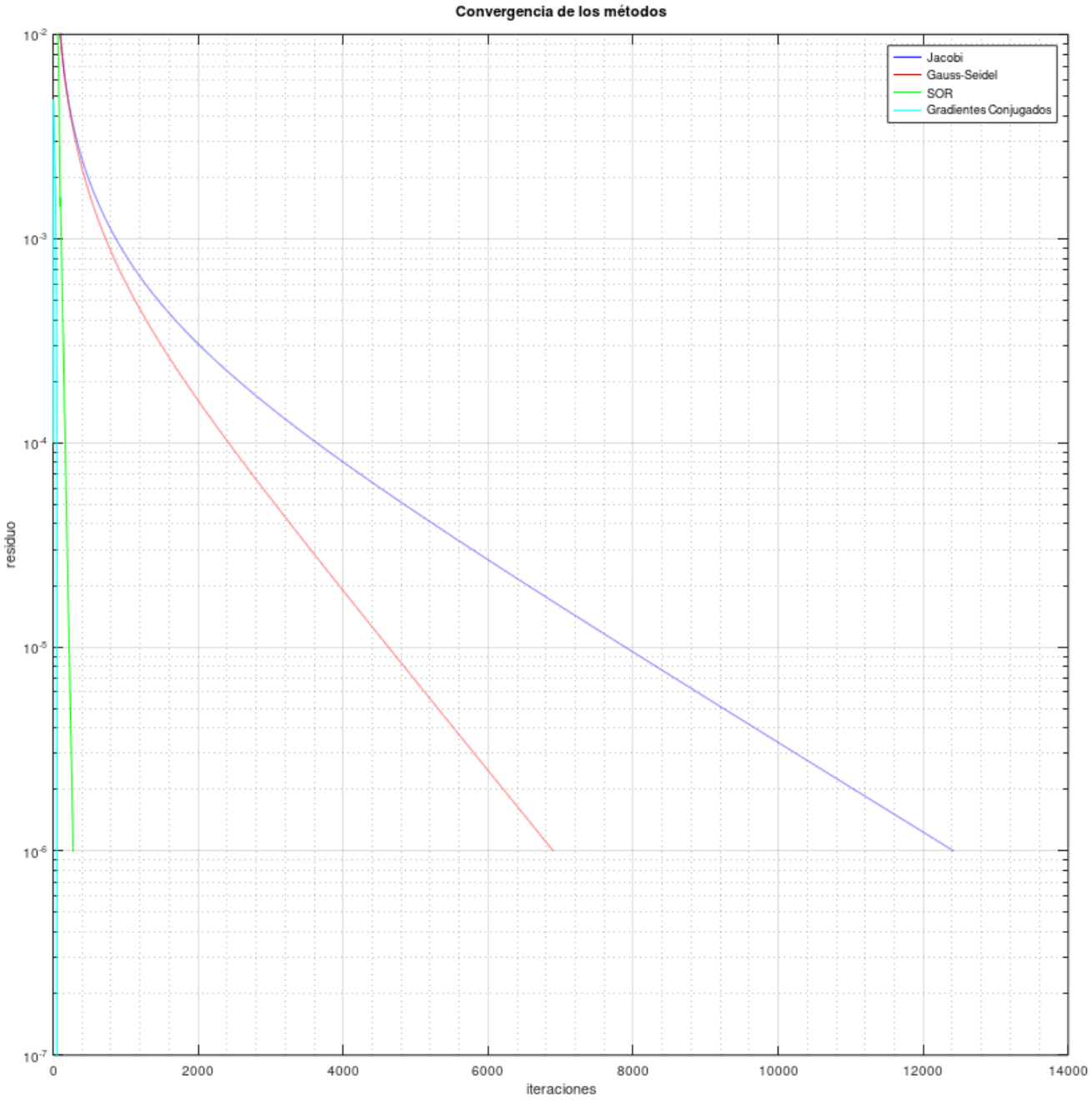
Resolución

A fin de llevar a cabo la resolución del inciso *a)* y *b)* se implementó un script para cada método, tal que permita resolver el sistema para el caso requerido en el actual ejercicio *N=100*, donde se realizó el conteo del total de iteraciones necesarias para resolver el sistema para cada uno, considerando una tolerancia de 10^{-6} y una cantidad máxima de iteraciones de 15000. En el caso del método SOR, se calculó el parámetro ω estimando un posible valor en el rango [1.4,2], dado que puede tomar valores entre 0 y 2 y, por lo general, es mejor estimar de más. Luego se iteró entre este rango de valores unas 10 veces para obtener el parametro de relajación mínimo, dando así un valor de 1.9333.

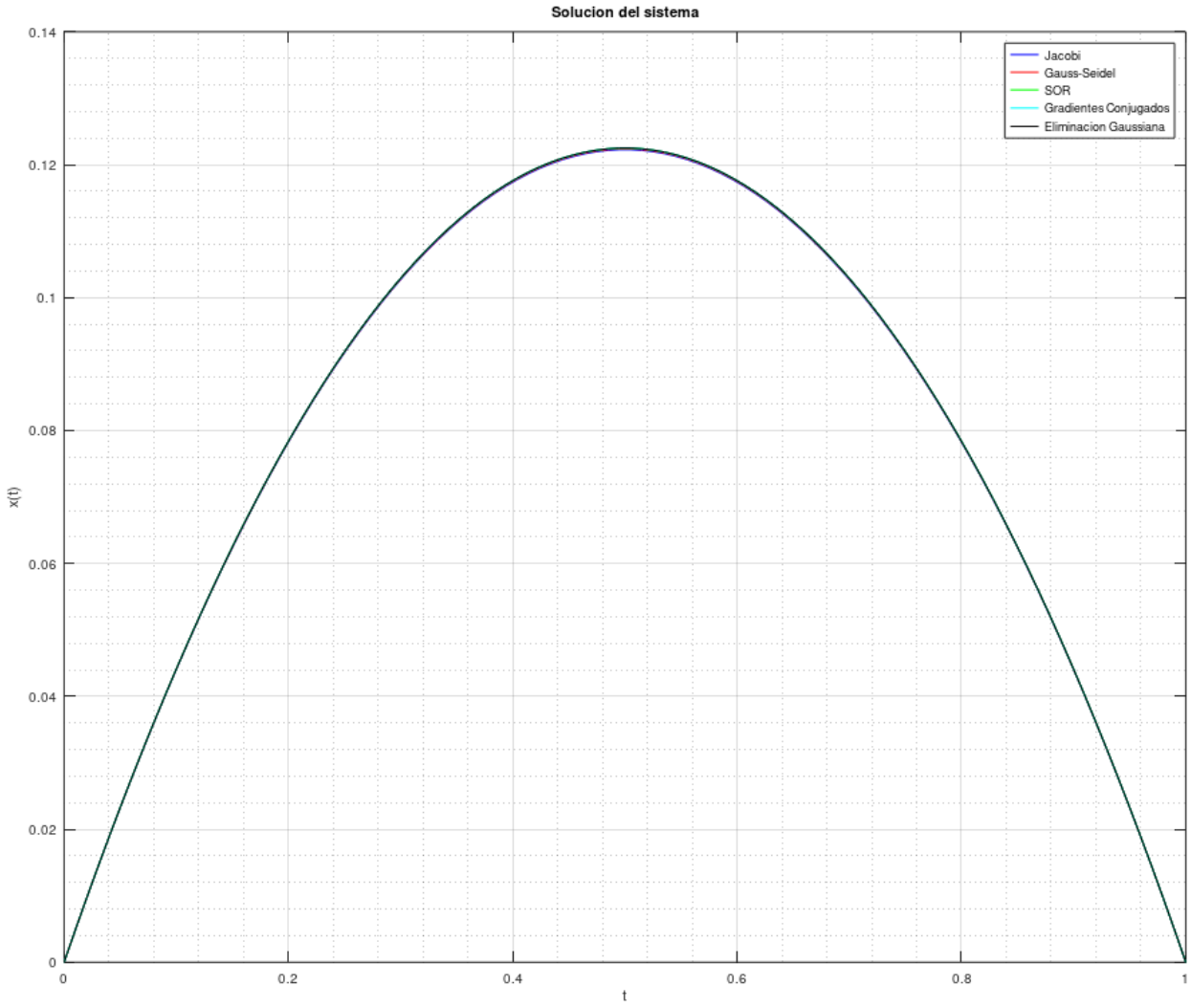
Luego, se analizó la convegencia de cada método, donde se sabe que Jacobi converge si su matriz es *estrictamente diagonal dominante*, Gauss-Seidel si su matriz es *diagonal dominante* o *simétrica y positiva definida* y gradientes conjugados si es *simétrica y definida positiva*. Además, se comprobó la convergencia a través del cálculo del radio espectral de la matriz de iteraciones, dónde este valor varía entre 0 y 1 y nos indica cuan rápido converge cada método iterativo. Dando los siguientes radios espectrales

- Jacobi: $\rho(T) = 0{,}9995$;
- Gauss-Seidel: $\rho(T) = 0{,}9990$;
- SOR: $\rho(T) = 0{,}9589$.

Esto puede visualizarse en la diferencia de cantidad de iteraciones necesarias para resolver el sistema, Jacobi necesitó 12423 iteraciones, Gauss-Seidel 6901 y SOR 277. Por otro lado, gradientes conjugados 58 y eliminación gaussiana 101. Posteriormente, se graficó el residuo de cada método en escala logaritimica.



Finalmente, se graficó la solución obtenida correspondiente a cada método, como una función $y = x(t)$ dónde t pertenece al intervalo [0,1].



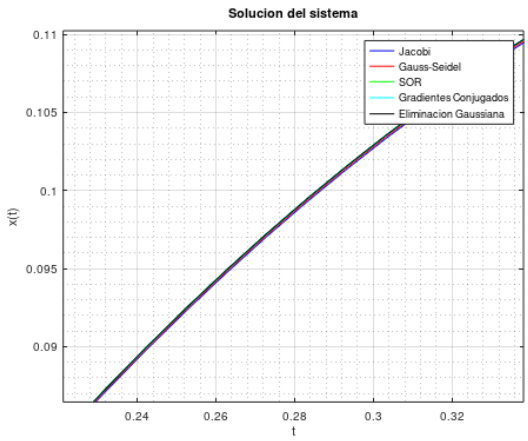


Figura 1: Zoom 1

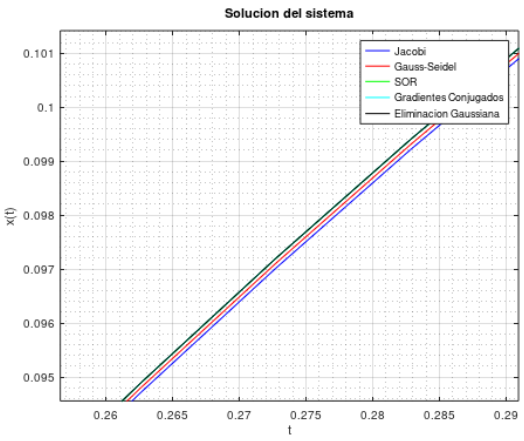


Figura 2: Zoom 2

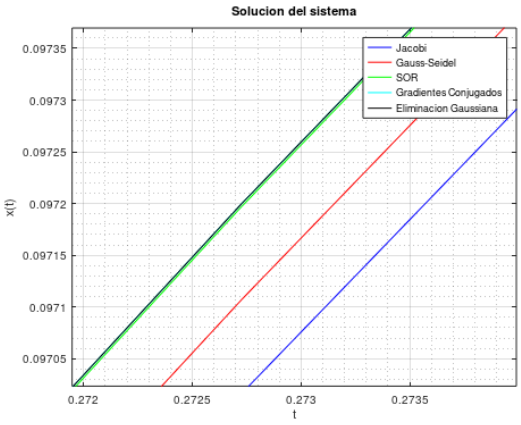


Figura 3: Zoom 3

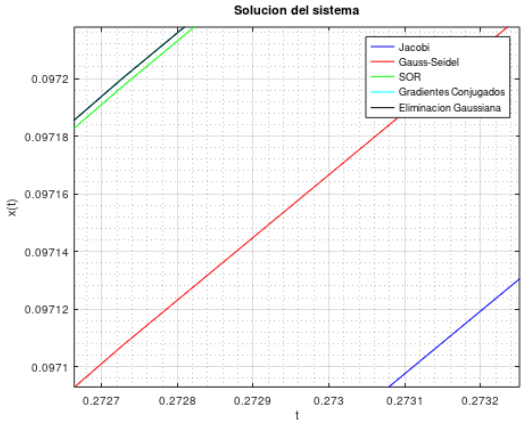


Figura 4: Zoom 4

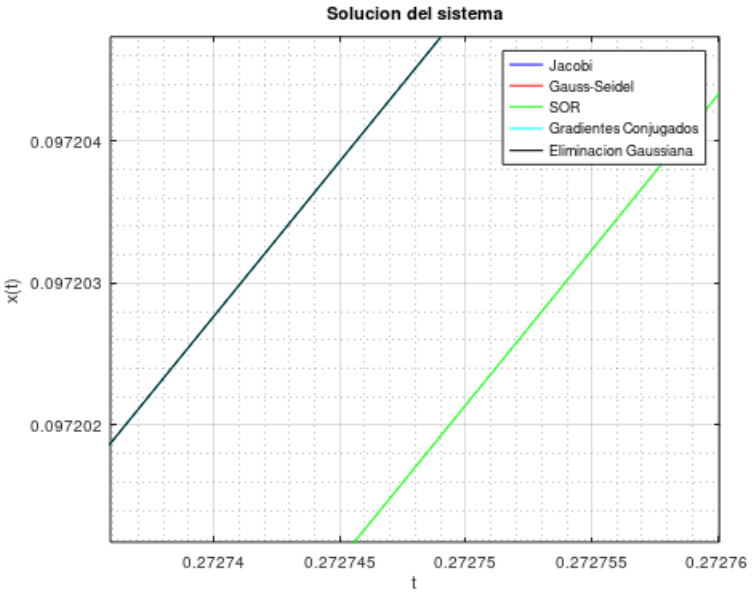


Figura 5: Zoom 5

Aunque a primera vista puede parecer que las soluciones correspondientes a cada método son iguales, al realizar zoom sucesivos se puede observar como estas son diferentes. Se visualiza como en los métodos iterativos la diferencia radica en el orden de la tolerancia elegida y la solución más se aproxima a la real según la velocidad de convergencia del método, en tanto, las soluciónes correspondientes a Gradientes Conjugados y Eliminación Gaussiana son indistinguibles, pues, con estos se obtuvo la solución exacta. Esto se debe a la naturaleza directa de los métodos, pese a que gradientes conjugados puede ser considerado iterativo por su estructura de solución, también puede considerarse como directo, dado la escasa cantidad de iteraciones que necesita para su convergencia.

ANEXO

Listing 1: gauss.m

```
function [x] = gauss(A,b)
  n=length(A(1,:));

  for k=1:n
    m = A(k+1:n,k)/A(k,k);
    b(k+1:n) = b(k+1:n) - m*b(k);
    A(k+1:n,k:n) = A(k+1:n,k:n) - m*A(k,k:n);
  endfor

  x=sust_atras(A,b);
endfunction
```

Listing 2: jacobi.m

```
function[x,it,r_h,t] = jacobi(A,b,x0,tol,maxit)
  %Calculo longitud del vector b
  tic();
  n =length(b);
  it=1;
  error =1;
  r_h = [];
  x = x0;
  while(it<maxit && error > tol)
    %Aplico formula
    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
    %Calculo error
    error = norm(x-x0,inf)/norm(x,inf);
    %Truco para guardar errores
    r_h = [r_h ; error];
    x0 = x;
    ++it;
  endwhile
  t=toc()
endfunction
```

Listing 3: gauss_seidel.m

```
function[x,it,r_h,t] = gauss_seidel(A,b,x0,tol,maxit)
  %Calculo longitud del vector b
  tic();
  n =length(b);
  it=1;
  error =1;
  r_h = [];
  x = x0;
  while(it<maxit && error > tol)
    %Aplico formula
    for i=1:n
    %El cambio con Jacobi es que aca hago uso del vector corriente
    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
    %Calculo error
    error = norm(x-x0,inf)/norm(x,inf);
    r_h = [r_h ; error];
    x0 = x;
    ++it;
  endwhile
  t = toc();
endfunction
```

Listing 4: SOR.m

```
function[x,it,r_h,t] = SOR(A,b,x0,tol,maxit,w)
  tic();
  %Calculo longitud del vector b
  n =length(b);
  it=1;
  error =1;
  r_h = [];
  x = x0;
  while(it<maxit && error > tol)
    %Aplico formula
    for i=1:n
    x(i) = (1-w)*x(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
    %Calculo error
    error = norm(x-x0,inf)/norm(x,inf);
    %Truco para guardar errores
    r_h = [r_h ; error];
    x0 = x;
    ++it;
   endwhile
   t = toc();
endfunction
```

Listing 5: GradienteConjugado.m

```
function [x, it, rh, t] = GradienteConjugado(A, b, x, maxit, tol)
  tic();
  residuo= b - A*x;
  v = residuo;
  c = residuo'*residuo;
  for it=1 : maxit
    if norm(v) < tol
      break;
    endif
    z = A*v;
    t = c/(v*z);
    x = x + t*v;
    residuo -= t*z;
    d = residuo'*residuo;
    rh(it) = norm(residuo,2);
    if rh(it) < tol
      break;
    endif
    v = residuo + d/(c*v)
    c = d;
  endfor
  t = toc();
 endfunction
```

Listing 6: TPentregable1.m

```
%Para el vector solucion x0,xN=0
N=100;
[A,b,x0]=crearMatriz(N);
tol=10^-6;
maxit=15000;
w = 1.5;

% a) y b)

%Jacobi
[xJacobi,itJacobi,r_hJacobi] = jacobi(A,b,x0,tol,maxit);
[TJacobi, cJacobi, tJacobi]=Tc_Jacobi(A,b,x0);
%Gauss-Seidel
[xGS,itGS,r_hGS] = gauss_seidel(A,b,x0,tol,maxit);
[TGS, cGS, tGS]=Tc_GaussSeidel(A,b);
%SOR
%Determinacion de parametro de relajacion para SOR
wsor = linspace(1.4,2,10);
vit=[];%Vector de iteraciones
for j=1:length(wsor)
  [xSor,itSor,r_hSor,tSor] = SOR(A,b,x0,tol,maxit,wsor(j));
  vit(j)=itSor;
 endfor
 dato = [wsor;vit]';%'
 [val pos] = min(dato(:,2));
 display("El parametro de relajacion optimo para SOR es ");
 w = wsor(pos)
 [TSor, cSor, tSor] = Tc_SOR(A, b, w);
 [xSorMin,itSorMin,r_hSorMin,tSorMin] = SOR(A,b,x0,tol,maxit,w);
%GradienteConjugado
[xGC, itGC, rhGC, tGC] = GradienteConjugado2(A, b, x0, maxit, tol);
%EliminacionGauss
[xGauss,itGauss] = gauss(A,b);

%Calculo de radios espectrales de las matrices de iteracion
display("Radio espectral matriz iteracion Jacobi");
radioJacobi = max(abs(eig(TJacobi)))
display("Radio espectral matriz iteracion Gauss-Seidel");
radioGS = max(abs(eig(TGS)))
display("Radio espectral matriz iteracion SOR");
radioSor = max(abs(eig(TSor)))

%Muestra iteraciones
display("Iteraciones Jacobi");
itJacobi
display("Gauss-Seidel");
itGS
display("SOR");
itSorMin
display("Gradientes Conjugados");
itGC
display("Eliminacion Gauss");
itGauss

%Grafico de cada residuo
figure(1)
semilogy(r_hJacobi,'-b;Jacobi;')
ylim([10^(-7) 10^(-2)])
grid on
grid minor
```

```
hold on
semilogy(r_hGS,'-r;Gauss-Seidel;')
semilogy(r_hSorMin,'-g;SOR;')
semilogy(rhGC,'kc;Gradientes Conjugados;')
title('Convergencia de los metodos')
xlabel('iteraciones')
ylabel('residuo')
hold off

%c)
%Grafico de las soluciones
d = linspace(0,1,100);

figure(2)
plot(d,xJacobi,'-b;Jacobi;')
grid on
grid minor
title('Solucion del sistema')
xlabel('t')
ylabel('x(t)')
hold on
plot(d,xGS,'-r;Gauss-Seidel;')
plot(d,xSorMin,'-g;SOR;')
plot(d,xGC,'-c;Gradientes Conjugados;')
plot(d,xGauss,'-k;Eliminacion Gaussiana;')
hold off
```