Calculo numérico

T. P. N°3

26 de abril de 2017

Profesor: Juan José Gómez Barroso
Alumno: Agustín N. Mendoza

Ejercicio 7:

Introduccion:

Un *sistema de ecuaciones lineales*, es un conjunto de *ecuaciones lineales*, conformado por variables independientes de primer grado y coeficientes constantes que multiplican dichas variables.

$$\begin{aligned} &a_{11}x_1+a_{12}x_2+a_{13}x_3+...+a_{1n}x_n=b_1\\ &a_{21}x_1+a_{22}x_2+a_{23}x_3+...+a_{2n}x_n=b_2\\ &... \quad ... \quad ... \quad ... \quad ... \quad ...\\ &a_{n1}x_1+a_{n2}x_2+a_{n3}x_3+...+a_{nn}x_n=b_n \end{aligned}$$

El problema consiste en encontrar los valores de dichas variables independientes. Para esto, se puede expresar el sistema de forma matricial **Ax=b,**

$$A\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix} * x \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_4 \end{bmatrix} = b \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{bmatrix}$$

y a partir de diferentes métodos iterativos, obtener el vector solución **x**.

Entre ellos podemos encontrar:

Método de Jacobi:

Consiste en construir una sucesión convergente definida iterativamente. El límite de esta sucesión es precisamente la solución del sistema. Si el algoritmo se detiene después de un número finito de pasos se llega a una aproximación al valor **x** de la solución del sistema.

El método siempre converge si la matriz **A** es estrictamente diagonal dominante. En caso contrario no se asegura la convergencia.

Método Gauss-Seidel:

Se parte de una aproximación inicial y se repite el proceso hasta llegar a una solución con un margen de error tan pequeño como se quiera. La diferencia entre este método y el de Jacobi es que, en este último, las mejoras a las aproximaciones no se utilizan hasta completar las iteraciones.

Este método converge si la matriz **A** es *diagonalmente dominante* o si es *simétrica* y *positiva definida*.

Método SOR (Successive Over Relaxation):

Es una modificación usada para mejorar la convergencia del método de Gauss-Seidel considerando un promedio ponderado ω de las iteraciones anteriores y actuales.

Si $0<\omega<1$ se conoce como subrelajación, se emplea para un sistema que no converge.

Si $1<\omega<2$ se le llama sobre-relajación, el cual acelera la convergencia del sistema que ya converge.

Si $\omega=1$ el resultado no se modifica.

Si $\omega>2$ el método diverge.

Método del Gradiente Conjugado:

La idea básica consiste en construir una base de vectores ortogonales y utilizarla para realizar la búsqueda de la solución en forma más eficiente.

El método asegura convergencia si la matriz **A** es símetrica y definida positiva.

Ejercicio 7:

Problema:

Resuelva el siguiente sistema de ecuaciones lineales con todos los métodos presentados en esta guía. Utilice una tolerancia en el residuo de 10^{-5} para la convergencia (realizar gráficas semilogarítmicas *norma del residuo vs. número de iteraciones*), etc. Compare los resultados con el método directo de Gauss, justificando si es necesario o no utilizar alguna estrategia de pivoteo. Analizar los resultados obtenidos e indicar cuál método piensa usted que es el más conveniente? Cómo justifica que los métodos iteratios logren o no convergencia? Qué expectativa puede tener sobre la precisión de los resultados obtenidos? Las entradas de la mariz de coeficientes del sistema lineal son,

$$a_{ij}=\begin{cases} 2\,i & si \quad j=i \\ 0.5\,i & si \quad j=i+2 \text{ ó } j=i-2 \\ 0.25\,i & si \quad j=i+4 \text{ ó } j=i-4 \\ 0 & si \quad \textit{en otra posición} \end{cases}$$

con $i = 1,...,N$ y $N = 250, 500, 1000$. Las entradas del vector de términos independientes son $b_i=\pi$. Utilice la norma infinito.

Desarrollo y resultados:

En primer término se genera la matriz correspondiente al sistema, mediante la función *GeneradorMatriz37* en Octave:

```
1 function [A,b,t] = GeneradorMatriz37 (n)
2   tic();
3   A=diag(2:2:2*n)+diag(0.5:0.5:0.5*(n-2),2)
4     +diag(0.25:0.25:0.25*(n-4),4)+diag(1.5:0.5:(0.5*n),-2)
5     +diag(1.25:0.25:(0.25*n),-4);
6   b=pi*ones(n,1);
7   t=toc();
8
9 endfunction
```

Luego se analizan las características de cada matriz generada.

Mediante la función de Octave *min(eig(A))* se halla el mínimo autovalor en cada matriz, si éste es positivo, por definición, la matriz es *definida positiva*.

```
1 >> [A1,b1,t1] = GeneradorMatriz37 (250);
2 >> [A2,b2,t2] = GeneradorMatriz37 (500);
3 >> [A3,b3,t3] = GeneradorMatriz37 (1000);
4 >> min(eig(A1))
5 ans =  1.8097
6 >> min(eig(A2))
7 ans =  1.8097
8 >> min(eig(A3))
9 ans =  1.8097
```

Se concluye que todas las matrices son *definidas positivas*.

Además se puede observar que **A1**, **A2** y **A3** son matrices *estrictamente diagonalmente dominantes*, debido a que el valor absoluto del elemento de la diagonal, para cada fila, es estrictamente mayor que la sumatoria de los valores absolutos de los elementos restantes de la fila. Ésto es:

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| < |a_{ii}|$$

Donde podemos asegurar que es cierto, debido a que cada fila tendrá, además del elemento de la diagonal principal, a lo sumo cuatro elementos no nulos.

$$\sum_{i=1}^{n} |a_{ii}| = |2*i| < \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| = |(2*0.5\,i)| + |(2*0.25\,i)| = |1.5\,i|$$

Las matrices tienen elementos en la diagonal distintos de cero, pero los que están adyacentes por debajo y por encima no lo son, por lo tanto por definición, no

son matrices tridiagonales.

Por último, las matrices no son simétricas, ya que no se cumple con la definición $a_{ij} = a_{ji}$

De acuerdo a las características de las matrices, por hipótesis, los métodos de Jacobi y Gauss-Seidel, con cualquier estimación inicial $x^{(0)}$, generan sucesiones $\{x^{(k)}\}$ que convergen a la única solución del sistema $A * x = b$

Por otro lado, como las matrices son *definidas positivas* y $1 < \omega < 2$ por el teorema de Ostrowki-Reich se asegura que el método SOR, para cualquier elección del vector inicial $x^{(0)}$, converge.

Debido a que las matrices no son símetricas, no se cumple con las hipótesis del método del Gradiente Conjugado, por lo cual no se puede asegurar la convergencia.

Se calculan los *radios espectrales* de las matrices relacionadas a los primeros tres métodos, se generan en Octave las funciones *CalcularTJ, CalcularTG* y *CalcularTS* presentadas en el Anexo, que calculan las matrices T_j, T_g, T_s , respectivamente. El máximo autovalor de estas matrices dará como resultado el *radio espectral* de cada una. Mediante la función de Octave *max(eig(T))* obtenemos los resultados.

```
1 >> [Tj1] = CalcularTJ (A1);
2 >> max(abs(eig(Tj1)))
3 ans =  0.74954
4 >> [Tj2] = CalcularTJ (A2);
5 >> max(abs(eig(Tj2)))
6 ans =  0.74988
7 >> [Tj3] = CalcularTJ (A3);
8 >> max(abs(eig(Tj3)))
9 ans =  0.74997
```

$$\rho(T_{j1}) = 0.74954$$
$$\rho(T_{j1}) = 0.74988$$
$$\rho(T_{j1}) = 0.74997$$

```
>> [TG1] = CalcularTG (A1);
>> max(abs(eig(TG1)))
ans =  0.22943
>> [TG2] = CalcularTG (A2);
>> max(abs(eig(TG2)))
ans =  0.22989
>> [TG3] = CalcularTG (A3);
>> max(abs(eig(TG3)))
ans =  0.23575
```

$$\rho(T_{g1}) = 0.22943$$
$$\rho(T_{g2}) = 0.22989$$
$$\rho(T_{g3}) = 0.23575$$

```
>> [Ts1] = CalcularTS (A1,1.02);
>> max(abs(eig(Ts1)))
ans =  0.54961
>> [Ts2] = CalcularTS (A2,1.02);
>> max(abs(eig(Ts2)))
ans =  0.55052
>> [Ts3] = CalcularTS (A2,1.02);
>> max(abs(eig(Ts2)))
ans =  0.55841
```

$$\rho(T_{s1}) = 0.54961$$
$$\rho(T_{s2}) = 0.55052$$
$$\rho(T_{s3}) = 0.55841$$

En el método SOR se utilizó ω = 1.02.

La rapidez de convergencia de un método iterativo depende directamente del *radio espectral* de la matriz asociada al método, en el cual, a menor *radio* mayor será la rapidez de convergencia.

Teniendo en cuenta los resultados anteriores, se puede concluir que el método de Gauss-Seidel tiene una rapidez de convergencia superior al resto, sin tener grandes variaciones en tanto el tamaño de las matrices aumenta. Es decir, en cada método la convergencia debe ser similar y el número de iteraciones constante, conforme el aumento de tamaño de la matriz.

En el Gradiente Conjugado se sabe que si el número de condición *k* es bajo,

el método es eficiente.

$$k(A)=\frac{|\lambda_{max}|}{|\lambda_{min}|}$$

Mediante las funciones de Octave, calculamos: *abs(max(eig(A)))/abs(min(eig(A)))*.

Para la matriz:

**A1* de *250x250*, obtenemos *k(A1) = 454.61*

**A2* de *500x500*, obtenemos *k(A2) = 929.17*

**A2* de *1000x1000*, obtenemos *k(A3) = 1885.0*

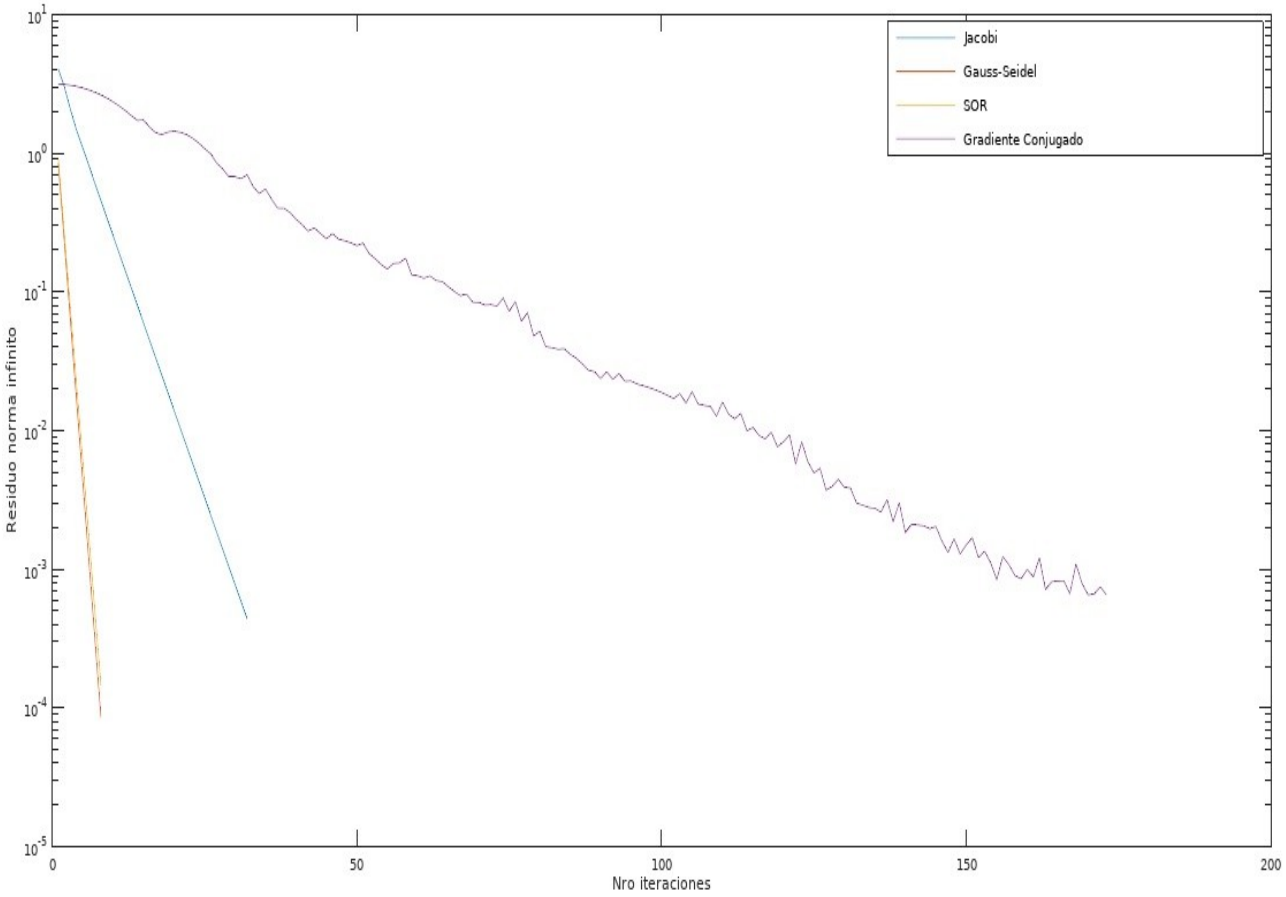
El valor de *k* es alto, por lo tanto el método es ineficiente.

Se aplican los métodos anteriores mediante los algoritmos en Octave *Jaccobi (A, b, x0, maxit, tol), Gauss_Seidel (A, b, x0, maxit, tol), SOR (A, b, x0, w, maxit, tol)* y *GradienteConjugado (A, b, x0, maxit, tol) (ver Anexo)*. Donde *A* es la matriz, *b* es el vector de términos independientes, *x0* es un vector nulo del mismo tamaño que *b*, *maxit* el número máximo de iteraciones (que será igual al tamaño de cada matriz) y *tol* es la tolerancia de error indicada, es decir 10^{-5} .

Para la matriz $A1_{250x250}$:

Método	Iteraciones (it)	Tiempo (s)	r_h(it)
Jacobi	32	0.41519	$4.3798\ x\ 10^{-4}$
Gauss-Seidel	8	0.10910	$8.5498\ x\ 10^{-5}$
SOR	8	0.12509	$1.4477\ x\ 10^{-4}$
Gradiente Conjugado	173	0.22013	$6.5405\ x\ 10^{-4}$

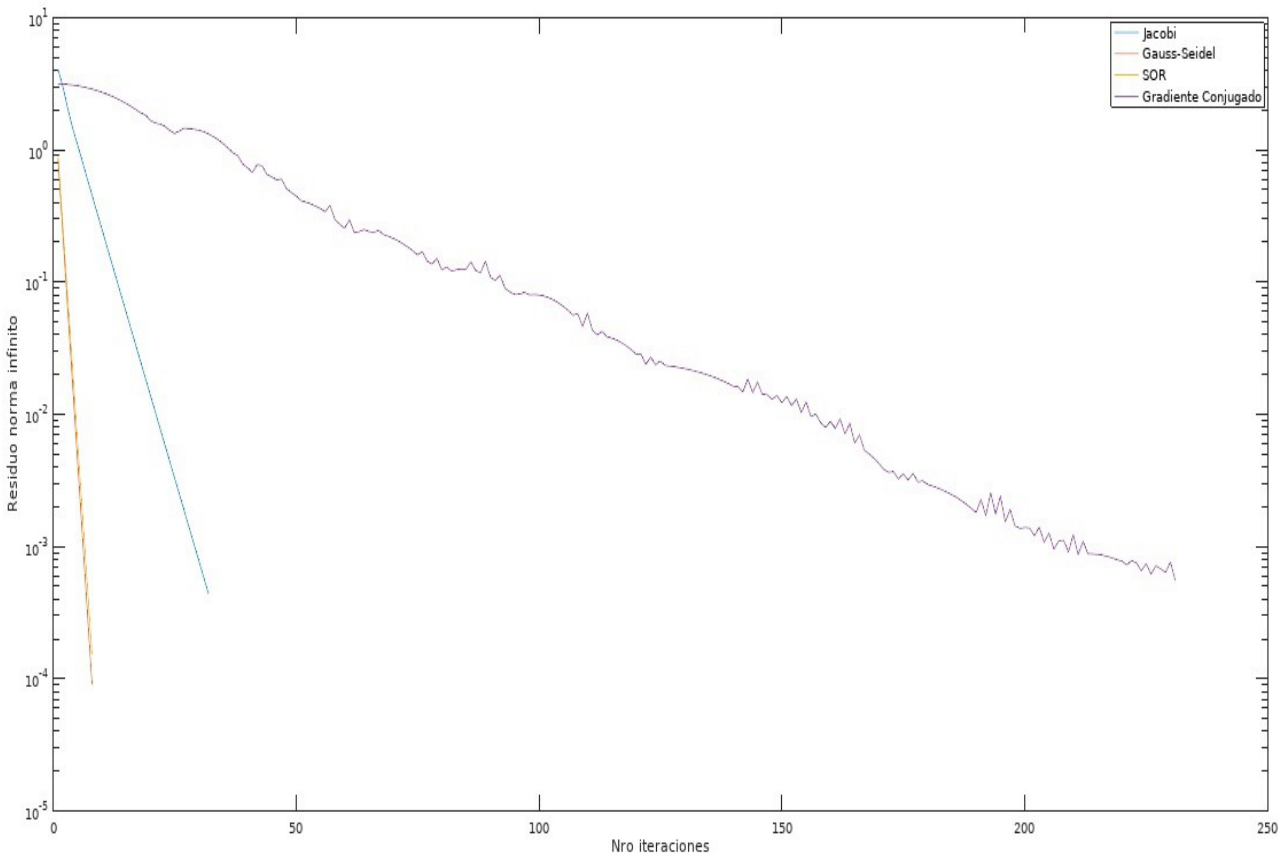
A partir de éstos datos se genera siguiente la gráfica:



Para la matriz $A2_{500x500}$:

Método	Iteraciones (it)	Tiempo (s)	r_h(it)
Jacobi	32	0.83776	$4.3798\ x\ 10^{-4}$
Gauss-Seidel	8	0.20816	$9.0881\ x\ 10^{-5}$
SOR	8	0.23197	$1.5357\ x\ 10^{-4}$
Gradiente Conjugado	231	0.48339	$5.5321\ x\ 10^{-4}$

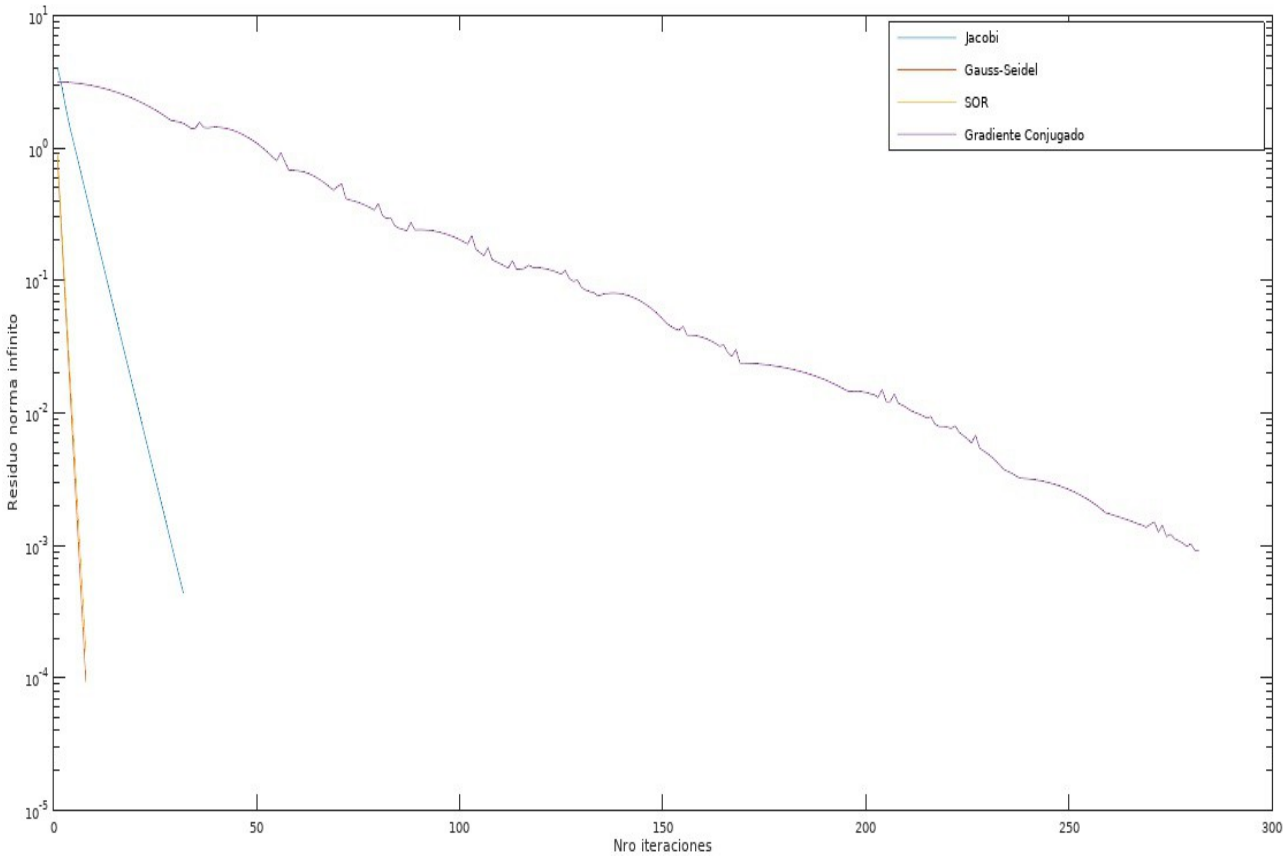
A partir de éstos datos se genera siguiente la gráfica:



Para la matriz $A3_{1000x1000}$:

Método	Iteraciones (it)	Tiempo (s)	r_h(it)
Jacobi	32	1.7755	$4.3798\ x\ 10^{-4}$
Gauss-Seidel	8	0.44933	$9.3532\ x\ 10^{-5}$
SOR	8	0.49436	$1.5789\ x\ 10^{-4}$
Gradiente Conjugado	282	0.28821	$9.1977\ x\ 10^{-4}$

A partir de éstos datos se genera siguiente la gráfica:



Se aplicará el método directo de Gauss através del algoritmo *Gauss (ver Anexo)* en instrucciones para Octave, para comparar los resultados obtenidos anteriormente.

Matriz	250x250	500x500	1000x1000
Iteraciones	250	500	1000
Tiempo (s)	0.13538	0.64148	6.1066

Como se sabe, el número de iteraciones es igual al tamaño de la matriz, por ende el número de iteraciones es directamente proporcional al crecimiento de tamaño de la matriz.

Conclusion:

Se observa que el número de iteraciones en los métodos de Jacobi, Gauss-Seidel y SOR se mantienen constantes. Por otro lado el método de Gradiente Conjugado aumenta su número de iteraciones conforme al crecimiento de tamaño de la matriz. Éste no presenta una buena eficiencia, ya que como se ha marcado anteriormente, presenta un número de condición elebado, el cual aumenta a medida que el tamaño de la matriz crece.

Comparando con el método directo de Gauss, éste tiene la ventaja que encuentra la solución exacta al sistema, el problema radica en la cantidad de iteraciones que debe hacer, a medida que la matriz aumenta el tiempo en realizar los calculos aumenta drásticamente.

Se concluye que Gauss-Seidel es el método más eficiente para los casos postulados.

Ejercicio 8

Problema:

Resuelva los siguientes sistemas lineales con el método de Gauss-Seidel y analice lo que sucede en cada caso. Luego intente resolverlos mediante el método directo de eliminacion de Gauss. Es necesario aplicar alguna estrategia de pivoteo? Si lo fuera, justifique por qué.

$$\left\{\begin{array}{c} 3x+y+z=5 \\ x+3y-z=3 \\ 3x+y-5z=-1 \end{array}\right\} \quad (1)$$

$$\left\{\begin{array}{c} 3x+y+z=5 \\ 3x+y-5z=-1 \\ x+3y-z=3 \end{array}\right\} \quad (2)$$

Desarrollo y resultados:

Se expresan los sistemas (1) y (2) en forma matricial, respectivamente:

$$A_1=\begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & -1 \\ 3 & 1 & -5 \end{bmatrix} \qquad A_2=\begin{bmatrix} 3 & 1 & 1 \\ 3 & 1 & -5 \\ 1 & 1 & -1 \end{bmatrix}$$

Se observa que la matriz A_1 es *estrictamente diagonal dominante*, por ende el método de Gauss no necesitará hacerse con pivoteo. En cambio, la matriz A_2 no cumple con esta condición, por lo cual se necesitará aplicar con pivoteo.

Se operando sobre el primer sistema:

Se carga la matriz A_1 y el vector de términos independientes. Luego

llamamos a la función *Gauss-Seidel* de Octave y obtenemos los resultados.

```
 1 >> A=[3 1 1; 1 3 -1; 3 1 -5];
 2 >> b=[5; 3; -1];
 3 >> [x,it,r_h,t] = Gauss_Seidel (A, b, b*0, length(A), 1e-5);
 4 x =
 5    0.95556
 6    1.03704
 7    0.98074
 8 it =  3
 9 r_h =
10      1.73333   0.40000   0.11556
11 t =  0.0010009
```

Ahora se llama a la funcion *Gauss* de Octave para resolver el mismo sistema.

```
1 >> [x,t] = Gauss (A,b)
2 x =
3    1.0000
4    1.0000
5    1.0000
6
7 t =  0.0010009
```

Ahora, se siguen los mismos pasos operando sobre el segundo sistema

```
>> A=[3 1 1; 3 1 -5; 1 3 -1];
>> b=[5; -1; 3];
>> [x,it,r_h,t] = Gauss_Seidel (A, b, b*0, length(A), 1e-5)
x =
    169.96
  -2395.33
  -7019.04

it =  3
r_h =

  9.6667e+001  1.7878e+003  3.3211e+004

t =  0.0010009
```

Se calcula el *radio espectral* de la matriz asociada al método

```
>> [Tg] = CalcularTG (A);
>> max(abs(eig(Tg)))
ans =  18.577
```

$$\rho(Tg)=18.577$$

Se resuelve el mismo sistema con el método de Gauss sin pivoteo, llamando a la función *Gauss*

```
>> [x,t] = Gauss (A,b)
warning: division by zero
warning: called from
    Gauss at line 5 column 6
warning: division by zero
warning: called from
    Sust_AT at line 7 column 9
    Gauss at line 9 column 4
x =

   NaN
   NaN
   NaN

t =  0.0010018
```

Se observa que Octave arroja un aviso de precaución debido a que se ha querido realizar una división por cero, por lo tanto el método no puede aplicarse sin pivoteo.

Se realiza la llamada a la función de Gauss con pivoteo *Gauss_CP*

```
>> [x,t] = Gauss_CP (A, b)
x =

   1.0000
   1.0000
   1.0000

t =   9.9993e-004
```

Conclusion:

Se observa que en ambos sistemas el método directo de Gauss arroja la solución exacta como es de esperarse, aunque en el segundo caso se necesita hacer con pivoteo, ya que el algoritmo intenta realizar una división por cero.

En el primer sistema Gauss-Seidel arroja una solución aproximada, con un error muy pequeño. Mientras que en el segundo sistema los resultados difieren mucho de los esperados, esto es debido a que el *radio espectral* es mayor que 1, por lo que el método diverge.

ANEXO

Implementaciones de Octave

Jacobi

```
function [x,it,r_h,t] = Jaccobi (A, b, x0, maxit, tol)
  tic();
  n=length(A(1,:));
  x=x0;
  it=0;
  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,'inf'); #norma infinito - RESIDUO
    error=norm(x0-x,'inf')/norm(x,'inf'); #error, criterio de parada
    if(error<tol)
      break;
    endif
    x0=x;
    it=it+1;
  endwhile
  t=toc();
endfunction
```

Gauss-Seidel

```
function [x,it,r_h,t] = Gauss_Seidel (A, b, x0, maxit, tol)
  tic();
  n=length(A(1,:));
  x=x0;
  it=0;
  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,'inf'); #norma infinito - RESIDUO
    error=norm(x0-x,'inf')/norm(x,'inf'); #error, criterio de parada
    if(error<tol)
      break;
    endif
    x0=x; #actualizo x0 es x^(k-1)
    it=it+1;
  endwhile
  t=toc();
endfunction
```

SOR

```
function [x,it,r_h,t] = SOR (A, b, x0, w, maxit, tol)
  tic();
  n=length(A(1,:));
  x=x0;
  it=0;
  while(it<maxit)
    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
    r_h(it+1)=norm(A*x-b,'inf'); #norma infinito - RESIDUO
    error=norm(x0-x,'inf')/norm(x,'inf'); #error, criterio de parada
    if(error<tol)
      break;
    endif
    x0=x; #actualizo x0 es x^(k-1)
    it=it+1;
  endwhile
  t=toc();
endfunction
```

Gauss

```
function [x,t] = Gauss (A,b)
  tic();
  n=length(A(1,:)); #Cantidad columnas de A
  for i=1:n-1
    m=A(i+1:n,i)/A(i,i);  #factor para hacer ceros hacia abajo
    b(i+1:n)=b(i+1:n)-m*b(i);
    A(i+1:n,i+1:n)=A(i+1:n,i+1:n)-m*A(i,i+1:n);
  endfor
  x=Sust_AT(A,b);
  t=toc();

endfunction
```

Sustitucion hacia Atras

```
function [x] = Sust_AT (A, b)
  x=0*b;    #creo el vector solucion del tamaño de b
  n=length(b); #tamaño de b
  x(n)=b(n)/A(n,n); #despejo el ultimo termino de la matriz y obtengo el resultado
  for i=n-1:-1:1 #recorro la matriz de final a principio
    x(i)=(b(i)-A(i,i+1:n)*x(i+1:n))/A(i,i);
  endfor
endfunction
```

Gauss_CP

```
function [x,t] = Gauss_CP (A, b)
  tic();
  x=0*b;
  n=length(A);
  indx=1:1:n;
  val=0;
  p=0;
  epsil=1e-20;
  for i=1:n-1
    [val,p]=max(abs(A(indx(i:n),i)));
    if (val<epsil)
      disp("No hay solucion");
      return;
    endif
    p=p+(i-1);
    if(indx(p)!=indx(i))
      aux=indx(i);
      indx(i)=indx(p);
      indx(p)=aux;
    endif
    m=A(indx(i+1:n),i)/A(indx(i),i);
    A(indx(i+1:n),i)=0;
    b(indx(i+1:n))=b(indx(i+1:n))-m*b(indx(i));
    A(indx(i+1:n),i+1:n)=A(indx(i+1:n),i+1:n)-m*A(indx(i),i+1:n);
  endfor
  x=Sust_AT(A(indx,:),b(indx));
  t=toc();
endfunction
```

Gradiente Conjugado

```
function [x,it,r_h,t] = GradienteConjugado (A, b, x0, maxit, tol)
  tic();
  r=b-A*x0;
  p=r;
  Ro=r'*p;
  x=x0;
  it=1;
  while(it<maxit)
    a=A*p;
    m=p'*a;
    alfa=Ro/m;
    x0=x;
    x=x+alfa*p;
    err = norm(x-x0,'inf')/norm(x,'inf');
    r_h(it)=norm(r,'inf');
    if(err<tol)
      break;
    endif
    r=r-alfa*a;
    Ro_old=Ro;
    Ro=r'*r;
    gama=Ro/Ro_old;
    p=r+gama*p;
    it=it+1;
  endwhile
  t=toc();
endfunction
```

CalcularTG

```
function [Tg] = CalcularTG (A)
  D=diag(diag(A));
  U=triu(A,1);
  L=tril(A,-1);
  Tg=-inv(D+L)*U;
endfunction
```

CalcularTJ

```
function [Tj] = CalcularTJ (A)
  D=diag(diag(A));
  U=triu(A)-D;
  L=tril(A)-D;
  Tj=inv(D)*(L+U);
endfunction
```

CalcularTS

```
function [Ts] = CalcularTS (A,w)
  D=diag(diag(A));
  U=triu(A)-D;
  L=tril(A)-D;
  Ts=inv(D-w*L)*((1-w)*D+w*U);
endfunction
```