CALCULO NUMÉRICO

Trabajo práctico Nº 3

26 DE ABRIL DE 2017

PROFESOR: -JUAN JOSÉ GÓMEZ BARROSO
Alumno: Yoris, Luciana

INTRODUCCION

Los métodos iterativos son usados para resolver sistemas de ecuaciones de dimensiones grandes con alto porcentaje de elementos ceros. Comienzan con una aproximación inicial $x^{(0)}$ a la solución real x y generan una sucesión de vectores $\{x^k\}$ que converge a x. Este tipo de métodos conlleva un proceso que convierte el sistema $Ax = b$ en otro equivalente $x = Tx + c$ para alguna matriz T y un vector c.

Luego de seleccionar el vector inicial $x^{(0)}$ la sucesión de vectores de la solución aproximada se genera calculando $x^{(k)} = Tx^{(k-1)} + c\ para\ cada\ k = 1,2,3 \ldots$

Los diferentes métodos iterativos a usar para resolver los ejercicios son:

Método de Jacobi:

Se genera el sistema equivalente $x^{(k)} = T_j x^{(k-1)} + c_j$ mediante los pasos seguidos en teoría donde se deduce que $T_j = D^{-1}(\mathrm{L} + \mathrm{U})\ y\ c_j = D^{-1}\mathrm{b}$ con $D = diagonal\ de\ A,\ L = triangular\ inferior\ de\ A$ y $U = triangular\ superior\ de\ A$. Tanto L como U tienen los elementos se du diagonal en cero.

Método de Gauss Seidel:

Es una mejora del método de Jacobi. Para calcular los términos $x^{(k)}$ no se usa $x^{(k-1)}$ sino que seemplen los $x^{(k)}$ ya calculados que serán mejor aproximación a la solución, esto permite una convergencia más rápida.

El sistema equivalente usado para el método iterativo de Gauss Seidel es:

$x^{(k)} = T_g x^{(k-1)} + c_g$ con $T_g = -(D + L)^{-1}U\ y\ c_{g=}(D + L)^{-1}b$

Método de Relajación:

Consiste en tomar el método de Gauss y mejorarlo utilizando un término w para asegurar una convergencia más rápida.

El sistema equivalente esta dado por $x^{(k)} = T_w x^{(k-1)} + c_w$ donde

$$T_w = (D - wL)^{-1} * [(1 - w)D + wU]\ y\ c_w = w(D - wL)^{-1}$$

Se usará el método SOR (succesive Over-Relaxation) en el que el valor de w esta entre 1<w<2

Método de Gradiente Conjugado:

Si bien como método directo es inferior a la eliminación gaussiana con pivoteo, es muy útil para resolver sistemas esparcidos de gran tamaño. El hecho de que pueda ser mirado como un método directo se debe a que en un numero de pasos finitos conduce a la solución exacta, al ser usado como método iterativo realiza una cantidad relativamente pequeña de iteraciones hasta lograr la convergencia deseada.

Ejercicio 7:

Resuelva el siguiente sistema de ecuaciones lineales con todos los métodos presentados en esta guía. Utilice una tolerancia en el residuo 10^{-5} para la convergencía. En cada caso, estime el costo del método en función del tiempo reloj de ejecución (comandos tic y toc de Octave), el número de iteraciones, taza de 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 iterativos logren o no convergencia? ¿Qué expectativas puede tener sobre la precisión de los resultados obtenidos? Las entradas de la matriz de coeficientes del sistema lineal son,

$$a_{ij}\begin{cases} 2i & si \quad j=i \\ 0.5i & si \quad j=i+2 \text{ ó } j=i-2 \\ 0.25i & si \quad j=i+4 \text{ ó } j=i-4 \\ 0 & en\ otra\ posición \end{cases} \qquad (1)$$

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

En primer lugar, se genera la matriz especificada con la función creada en octave $GenSistema(n)\ donde\ n = tamaño\ de\ la\ matriz$:

```
1  function [A,b] = GenSistema (n)
2
3  b=pi*ones(n,1);
4
5   v1=0.5:0.5:0.5*(n-2);
6   v2=0.25:0.25:0.25*(n-4);
7   v3=1.5:0.5:0.5*n;
8   v4=1.25:0.25:0.25*n;
9
10  A=zeros(n)+diag(2:2:2*n);
11  A=A+diag(v1,2)+diag(v2,4)+diag(v3,-2)+diag(v4,-4);
12
13 endfunction
```

Se analizan las características de la matriz:

La matriz es definida positiva:

Se sabe que una matriz es definida positiva si todos sus autovalores son positivos. Se halla el mínimo autovalor para los tamaños de matriz dada mediante las funciones de octave $min(eig(A))$.

```
>> [A1 b1]=GenSistema(250)
>> [A2 b2]=GenSistema(500)
>> [A3 b3]=GenSistema(1000)
>> min(eig(A1))
ans =  1.8097
>> min(eig(A2))
ans =  1.8097
>> min(eig(A3))
ans =  1.8097
```

Como todos son positivos la matriz especificada con sus diferentes tamaños es definida positiva.

La matriz es estrictamente diagonal dominante:

Se sabe que una matriz es estrictamente diagonal dominante si y solo si para cada fila se cumple que el valor absoluto del elemento de la diagonal es estrictamente mayor a la sumatoria de los valores absolutos de los elementos restantes de la fila, es decir:

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

De (1) se nota que para cada fila i de la matriz A el elemento de la diagonal tiene un valor de $2i$, además A tiene como mucho 4 elementos no nulos en la fila donde se encuentra el elemento a_{ii} de los cuales, a lo sumo dos de ellos son la mitad del elemento de la diagonal y los otros dos son 0.25 de dicho elemento.

Se escribe:

$a_{ii} = 2i \; con \; i = 1, \dots, n$.

$De \; esta \; igualdad \; se \; deduce \; que \; i = \frac{a_{ii}}{2} \; (2)$

Luego, el valor máximo que puede tomar la suma de los elementos $|a_{ij}| \; para \; i \neq j$ es:

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

$remplazando \; por \; el \; valor \; de \; i \; obtenido \; en \; (2)$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} |a_{ij}| = |1.5i| = \left|(1.5) * \frac{a_{ii}}{2}\right| = |(0.75) * a_{ii}| \;\; y \; se \; cumple \; que \; \sum_{\substack{i=1 \\ i\neq j}}^{n} |a_{ij}| < |a_{ii}|$$

Por lo que A es estrictamente diagonal dominante, y por teorema se sabe que al cumplirse esto la matriz es no singular. Toda matriz no singular tiene solución única.

La matriz no es tridiagonal:

No cumple con la definición, si bien tiene los elementos de la diagonal distintos de cero, los que están adyacentes por encima y por debajo de esta no lo son.

La matriz no es simétrica:

Para ser simétrica debe cumplir con la definición, es decir se debe satisfacer que $a_{ij} = a_{ji} \;\; \forall i \forall j$

$Para \;\; a_{ij} \; donde \;\; i = 3 \;\; y \; j = 1 \;\; tenemos \; que \; j = i - 2, entonces \; a_{ij} = 0.5i = 1.5$

$Para \;\; a_{ji} \; donde \;\; i = 1 \;\; y \; j = 3 \; tenemos \; que \; j = i + 2, entonces \;\; a_{ij} = 0.5i = 0.5$

No cumple con la definición ya que $a_{ij} \neq a_{ji}$.

Análisis de convergencia:

Por teorema se sabe que, si la matriz A es estrictamente diagonal dominante, entonces con cualquier estimación inicial $x^{(0)}$ a la solución, tanto el método de Jacobi como el de Gauss Seidel dan sucesiones $\{x^{(k)}\}$ que convergen a la única solución del sistema $A * x = b$.

Por el teorema de Ostrowki-Reich, al ser A definida positiva y $1 < w < 2$ se puede asegurar que el método SOR converge para cualquier elección del vector inicial aproximado $x^{(0)}$.

Si además A fuera una matriz definida, tridiagonal se podría elegir un w óptimo para el método de SOR dado por $w = \frac{2}{1+\sqrt{1-[\rho(T_j)]^2}}$ $con\ \rho = Radio\ espectral\ y\ T_j = matriz\ de\ jacobi$ mencionada anteriormente , pero se sabe que A no es tridiagonal.

Para los métodos de Jacobi, Gauss Seidel y SOR se hallan los radios espectrales (ρ) de las matrices relacionadas a cada método iterativo, para ellos se generan en octave los algoritmos que calculan las matrices $T_J, T_g\ y\ T_w$. Se sabe que el radio espectral es el máximo autovalor de dichas matrices por lo tanto se usan las funciones de octave $max\left(abs\left(eig(T)\right)\right)$ para calcular el radio de cada matriz obteniendo la siguiente tabla:

Tamaño de Matriz / Radio espectral	n=250	n=500	n=1000
$\rho(T_j)$	0.74954	0.74988	0.74997
$\rho(T_g)$	0.22943	0.22978	0.23731
$\rho(T_w)$ con w=1.02	0.54961	0.55031	0.55478
$\rho(T_w)$ con w=1.5	0.57817	0.57831	0.57911
$\rho(T_w)$ con w= 1.8	0.83668	0.83669	0.83670
$\rho(T_w)$ con w=1.99	0.99199	0.99199	0.99199

La rapidez de convergencia de un procedimiento iterativo depende del radio espectral de la matriz relacionada con el método, la convergencia se da más rápido en aquel método cuya matriz asociada tenga un radio espectral mínimo.

Al observar la tabla se nota que la matriz relacionada al método de gauss es aquella que tiene un radio espectral mínimo por lo que Gauss Seidel converge más rápido que el resto de los procesos iterativos.

Debería suceder que el método SOR converja más rápido que el método de Gauss, sin embargo, no se ha podido encontrar un w optimo que minimice el radio espectral de la matriz. De ahora en más se empleará w=1.02 para realizar cálculos ya que de todos los evaluados es el que hace que la convergencia sea más rápida.

A medida que el tamaño de la matriz aumenta el radio espectral asociado a cada matriz aumenta muy poco, por lo que debería esperarse que, para cada método, la convergencia debería ser similar y el número de iteraciones debería permanecer constante al aumentar el tamaño.

Se analiza el número de condicionamiento k para el método de Gradiente conjugado, se sabe que si el número de condicionamiento es bajo el método es eficiente. K esta dado por: $k(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$ cuyo valor λ hace referencia a los autovalores de la matriz.

Se obtiene k en octave aplicando $abs(max(eig(A)))/abs(min(eig(A)))$

Tamaño de matriz	n=250	n=500	n=1000
Numero de condicionamiento k	454.61	929.17	1885.0

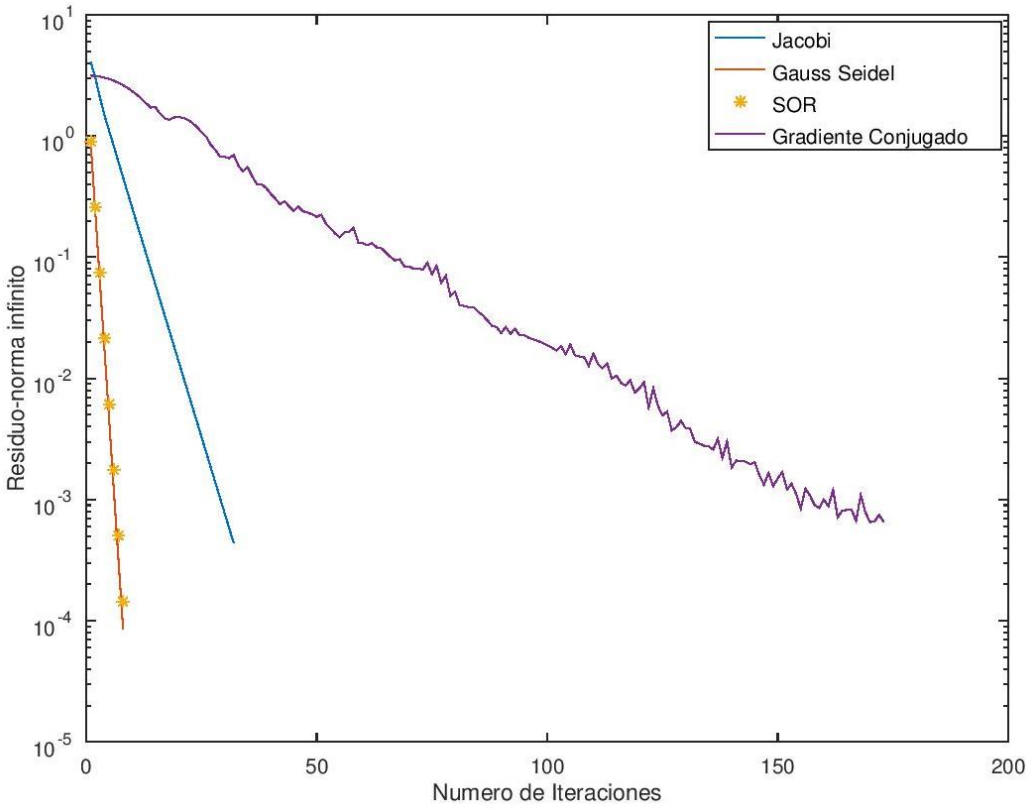
La matriz no está bien condicionada vemos que k es grande y aumenta conforme el tamaño de la matriz aumenta.

Se aplican los algoritmos $Jaccobi(A, b, x0, maxit, tol)$, $GaussSeidel(A, b, x0, maxit, tol)$, $SOR\ (A, b, x0, w, maxit, tol)$ y $GradienteConjugado(A, b, x0, Maxit, Tol)$ de cada método generados en octave y luego se grafica mediante $GraficaMetodos\ (A, b, w)$. En los parámetros se pasan $A\ y\ b$ generados, un vector nulo $x0 = 0 * b$, el valor máximo $(maxit)$ eligiendo el tamaño de la matriz como máximo valor $maxit = length(A)$ y $tol = 1e - 5$ que es el valor de tolerancia indicado.

Para n=250

Método	**Iteraciones**	**Tiempo(seg)**
Jacobi	32	0.68074
Gauss Seidel	8	0.17128
SOR	8	0.18821
Gradiente Conjugado	173	0.035031

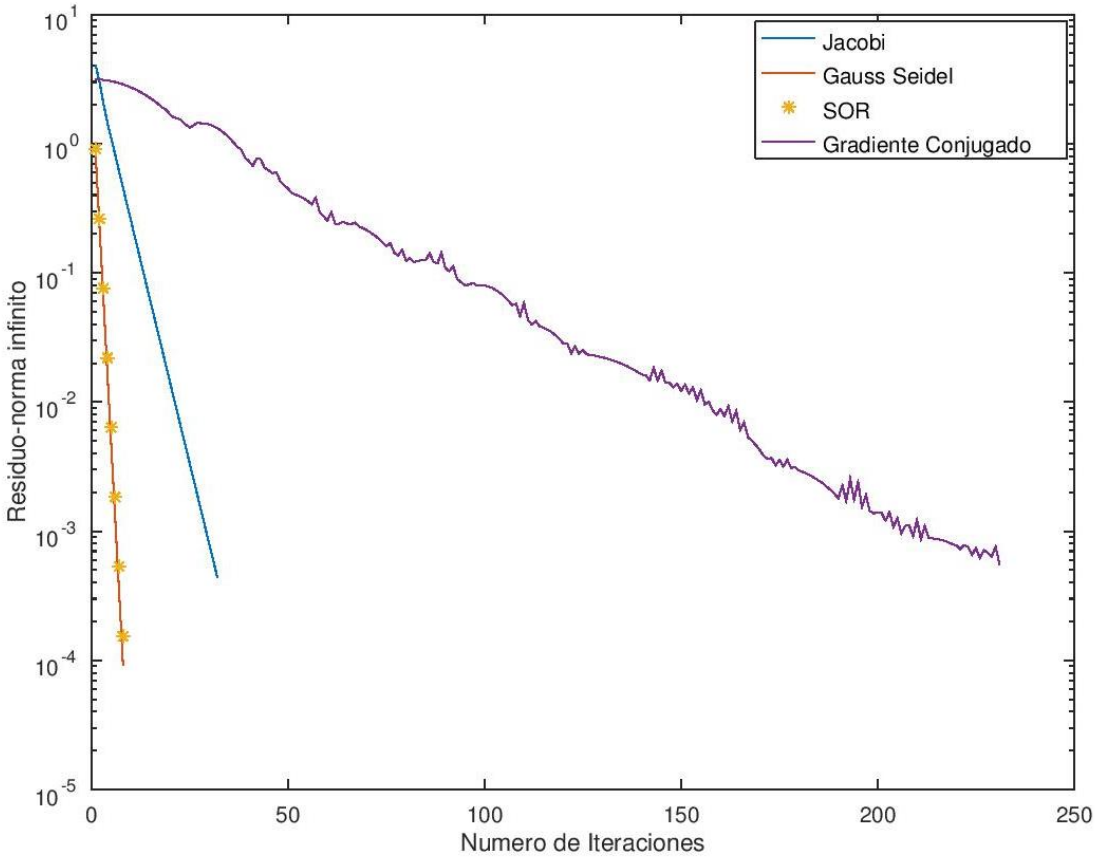
Grafica obtenida:



Para n=500

Método	Iteraciones	Tiempo(seg)
Jacobi	32	1.4425
Gauss Seidel	8	0.36141
SOR	8	0.40047
Gradiente Conjugado	231	0.099098

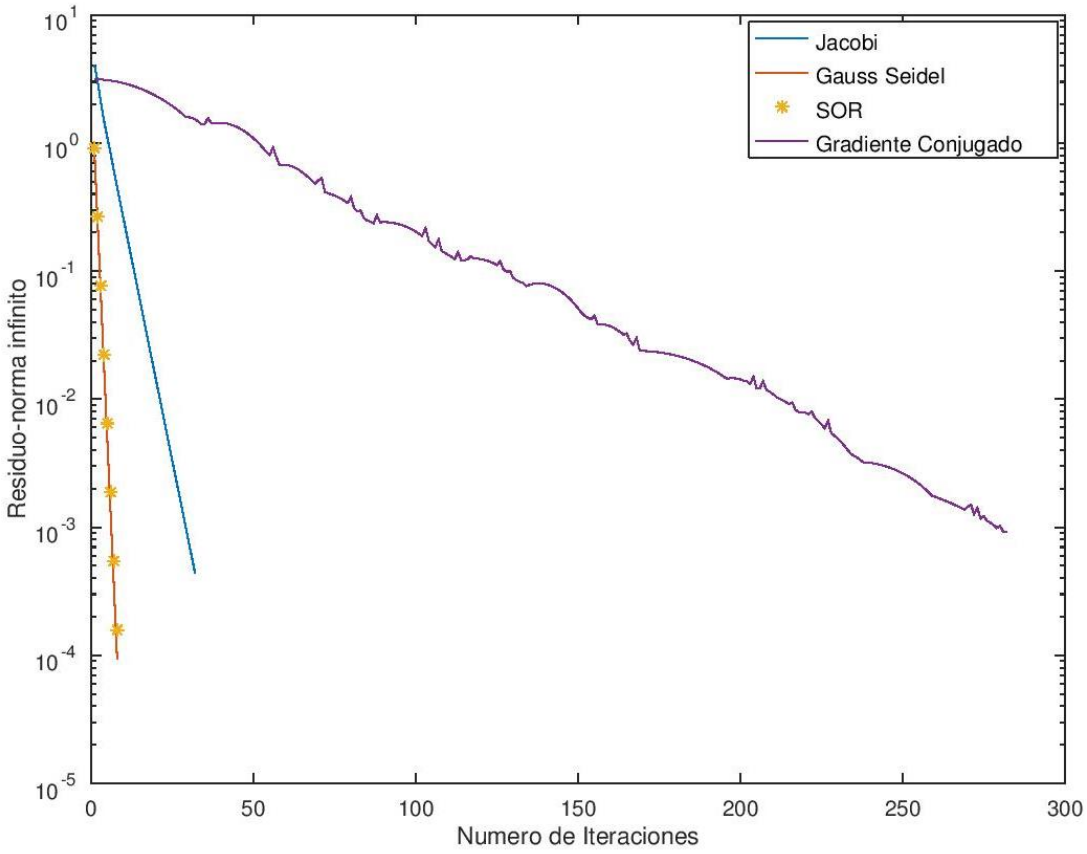
Grafica obtenida:



Para n=1000

Método	Iteraciones	Tiempo (seg)
Jacobi	32	3.3396
Gauss Seidel	8	0.83742
SOR	8	0.91794
Gradiente Conjugado	282	0.49152

Grafica Obtenida:



Como se nota, el número de iteraciones en Jacobi, Gauss Seidel y SOR permanece constante, no sucede lo mismo con el método de Gradiente conjugado, se observa que este tiene una mayor cantidad de iteraciones, y podría decirse que no es eficiente porque la matriz A presenta un numero de condición grande y este aumenta cuando el tamaño de A lo hace, podría precondicionarse la matriz para así generar un número de condicionamiento más chico y optimizar el método de Gradiente Conjugado, esto sería de utilidad porque si bien hace muchas iteraciones el tiempo que tarda es menor al resto de los métodos. Como no se ha podido hallar un w óptimo para SOR, el método iterativo de Gauss Seidel sigue siendo el más recomendable, tiene mejor convergencia, pocas iteraciones y el tiempo en que converge sigue siendo menor a Jacobi y a SOR en este caso.

Ahora, se aplica el método directo de Gauss, se sabe que el número de iteraciones es igual al tamaño de la matriz, por lo que al aumentar su tamaño se realizan mayor cantidad de iteraciones.

Tamaño de matriz (n)	250	500	1000
Iteraciones	250	500	1000
Tiempo (seg)	0.094131	1.0150	8.6397

La ventaja que tiene el método directo es que se halla la solución exacta, pero el problema que se presenta es el tiempo, al ver la tabla se observa un aumento considerable de tiempo en la solución de la matriz de dimensión 1000.

No es necesario realizar una estrategia de pivoteo ya que la matriz es estrictamente diagonal dominante, lo que indica que los valores mínimos no están en la diagonal y tampoco hay ceros en ella.

Por último, se hace referencia a los métodos de corte utilizados, los algoritmos pueden terminar o bien porque el número de iteraciones llega a un máximo determinado, en este caso nuestro método diverge, o porque se cumple que $\frac{\|x^k - x^{k-1}\|_\infty}{\|x^k\|_\infty} < tol$,es decir, la norma infinito de la solución actual calculada con respecto a la solución calculada en la iteración anterior, lo que se conoce como error relativo, debe ser menor a la tolerancia que se ha indicado, en este caso el método converge.

Ejercicio 8:

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 eliminación de Gauss. ¿Es necesario aplicar alguna estrategia de pivoteo? Si lo fuera, justifique por qué.

$$\begin{cases} 3x + y + z = 5 \\ x + 3y - z = 3 \\ 3x + y - 5z = -1 \end{cases} \quad (1)$$

$$\begin{cases} 3x + y + z = 5 \\ 3x + y - 5z = -1 \\ x + 3y - z = 3 \end{cases} \quad (2)$$

Para el sistema (1), se observa que no es necesario aplicar pivoteo ya que su matriz correspondiente $A_1 = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & -1 \\ 3 & 1 & -5 \end{bmatrix}$ es estrictamente diagonal dominante. Sin embargo en el sistema de ecuaciones (2) cuya matriz representativa es $A_2 = \begin{bmatrix} 3 & 1 & 1 \\ 3 & 1 & -5 \\ 1 & 3 & -1 \end{bmatrix}$, no sucede esto por lo que requiere que se aplique pivoteo parcial.

Para la matriz (1):

```
>> A=[3 1 1;1 3 -1;3 1 -5] % Se carga la matriz
A =
    3    1    1
    1    3   -1
    3    1   -5
>> b=[5 3 -1]' % Se carga el vector de términos independientes
b =
    5
    3
   -1
```

Resolviendo el sistema (1) con el método iterativo Gauss Seidel :

```
>> [x,it,r_h,T] = GaussSeidel(A,b,b*0,length(A),1e-5) % Se llama al método

>> x % Se pide ver el vector solución
x =

    1.0889
    1.0667
    1.0667

>> it % Se pide ver la cantidad de iteraciones
it =  3

>> r_h % Se pide ver el vector Residuo
r_h =

    1.73333   0.40000

>> T % Se pide ver el tiempo
T =  0.0010011
```

Resolviendo el sistema (1) con el método directo de eliminación de Gauss:

```
>> [x,r_h,T] = Gauss (A,b) % Se llama al método

>> x % Se pide ver el vector solución
x =

    1.0000
    1.0000
    1.0000

>> T % Se pide ver el tiempo
T =  0.018986
>> r_h % Se pide ver el vector residuo
r_h =

  0.0000e+000
  2.2204e-016
  0.0000e+000
```

Como podemos ver el método directo de gauss nos da la solucione exacta mientras que el método iterativo nos da una solución aproximada, sin embargo, el tiempo en el método iterativo es menor al del método directo.

Para el sistema (2), primero:

```
>> A=[3  1 1;3 1 -5;1 3 -1] % Se carga la matriz
A =

    3    1    1
    3    1   -5
    1    3   -1

>> b=[5 -1 3]' % Se carga el vector de términos independientes
b =

    5
   -1
    3
```

Se resuelve el sistema (2) con el método iterativo Gauss-Seidel :

```
>> [x,it,r_h,T] = GaussSeidel(A,b,b*0,length(A),1e-5) % Se llama al método

>> x % Se pide ver el vector solución
x =

    10.111
  -128.000
  -376.889

>> it % Se pide ver el número de iteraciones
it =  3

>> r_h % Se pide ver el vector residuo
r_h =

     96.667   1787.778

>> T % Se pide ver el tiempo
T =  0.0020020
```

Resolviendo el sistema (2) con el método directo de eliminación de Gauss sin Pivoteo:

```
1
 2 >> [x,r_h,T] = Gauss (A,b)% Se llama al método
 3
 4 warning: division by zero
 5 warning: called from
 6     Gauss at line 30 column 6
 7 warning: division by zero
 8 warning: called from
 9     sust_at_sindex at line 30 column 9
10     Gauss at line 36 column 5
11
12 >> x % Se pide ver el vector solución
13 x =
14
15    NaN
16    NaN
17    NaN
18
19 >> T % Pedimos ver el tiempo
20 T =  0.0040040
```

Se observa que no se puede aplicar gauss sin pivoteo, el warning nos está diciendo que hay presente una división con denominador cero.

Resolviendo el sistema (2) con el método directo de eliminación de Gauss con Pivoteo:

```
1  >> [x,r_h,T] = GaussConPivoteo(A, b) %Llamamos al metodo
 2
 3 >>x % Se pide ver el vector solución
 4 x =
 5
 6    1.0000
 7    1.0000
 8    1.0000
 9
10 >>r_h % Se pide ver el vector Residuo
11 r_h =
12
13   0.0000e+000
14   0.0000e+000
15   -2.2204e-016
16
17 >>T % Se pide ver el tiempo
18
19 T =  0.0020020
```

Gauss con pivoteo nos da la solución exacta y el vector de error es muy pequeño, el tiempo en el que lo hace es corto.

En el método iterativo de Gauss-Seidel para (2) calculado anteriormente se obtienen valores inesperados, por lo que se nota que algo no anda bien, se comprueba si el método realmente converge, para ellos se calcula el radio espectral utilizando el algoritmo $CalculaTg$ y luego hallando $max(abs(eig(Tg)))$

```
>> Tg=CalculaTg(A)
Tg =

    0.00000   -0.33333  -0.33333
    0.00000    1.00000   6.00000
    0.00000    2.66667  17.66667

>> max(abs(eig(Tg)))
ans =  18.577
```

El radio $\rho(T_g) > 1$, y por teorema se puede asegurar que el método diverge y la cantidad de iteraciones que realizo se debe a que el método de corte era con $it < (tamaño\ de\ A)$.

Si se pone un numero de corte mayor:

```
>> [x,it,r_h,T] = GaussSeidel(A,b,b*0,1000,1e-5)
>>x
x =

    NaN
    NaN
    NaN
>>it
it =  1000
```

El método corto cuando llego a las 1000 iteraciones.

En base a los resultados obtenidos en este ejercicio, se puede concluir que sería mejor opción utilizar los métodos de resolución directos ya que al ser matrices pequeñas se puede llegar al resultado exacto sin preocuparse por el tiempo ni la cantidad de iteraciones realizadas, siempre teniendo en cuenta que si la matriz tiene elementos ceros o muy pequeños en la diagonal se deberá aplicar el método directo de gauss con pivoteo. En el caso de que se quiera aplicar el método de Gauss Seidel iterativo, es de suma importancia evaluar el radio espectral, este nos va a determinar si el método converge o diverge, si no lo tenemos en cuenta puede pasar que los resultados sean muy diferentes a los que realmente buscamos y esto nos generará inconvenientes.

A continuación de indexaran los métodos usados para la resolución de ambos ejercicios.

Método iterativo de Jacobi:

```
function  [x,it,r_h,T] = Jaccobi(A,b,x0,maxit,tol)
  tic();
  n=length(A);
  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)=norm(A*x-b,'inf');
    error=norm(x-x0,'inf')/norm(x,'inf');
      if error<tol
      break;
      endif
    x0=x;
    it= it+1;
    endwhile
  T=toc();
  r_h(it)=norm(A*x-b,'inf');
endfunction
```

Método iterativo de Gauss Seidel:

```
function [x,it,r_h,T] = GaussSeidel(A,b,x0,maxit,tol)
   tic();

   n=length(A);
   it=1;
   x=x0;
   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)=norm(A*x-b,'inf');
    error=norm(x-x0,'inf')/norm(x,'inf');
      if(error<tol)
      break;
      endif
    it=it+1;
    x0=x;
   endwhile
    r_h(it)=norm(A*x-b,'inf');
    T=toc();
endfunction
```

Método iterativo SOR:

```
function [x,it,r_h,T] = SOR (A,b,x0,w,maxit,tol)
  tic();
  it=1;
  n=length(A(1,:));
  x=x0;
    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)=norm(A*x-b,'inf');
     error=(norm(x0-x,'inf')/norm(x,'inf'));
       if(error<tol)
         break
       endif
     x0=x;
     it=it+1;
    endwhile
  r_h(it)=norm(A*x-b,'inf');
  T=toc();
endfunction
```

Método de 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;
    error = norm(x-x0,'inf')/norm(x,'inf'); %error relativo
    r_h(it)=norm(r,'inf');
    if(error<Tol)
      break;
    endif
    r=r-alfa*a;
    ro_old=ro;
    ro= r'*r;
    gamma=ro/ro_old;
    p=r+gamma*p;
    it=it+1;
  endwhile
T=toc();
endfunction
```

Método directo de Gauss sin pivoteo:

```
function [x,r_h,T] = Gauss (A,b)

  tic();
  n=length(b);
   for i=1:n-1
     m=A(i+1:n,i)/A(i,i);
     A(i+1:n,i)=0;
     A(i+1:n,i+1:n)=A(i+1:n,i+1:n) - m*A(i,i+1:n);
     b(i+1:n)=b(i+1:n) - m*b(i);
   endfor
  x=sust_at_sindex(A,b);
  r_h=A*x-b;
  T=toc();

endfunction
```

Método directo de Gauss con pivoteo:

```
function [x,r_h,T] = GaussConPivoteo(A, b)
  tic();
  x = b*0;
  n = length(b);
  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))); % Se obteniene val max y la posicion
p
    if(val<epsil)
      disp("no hay solucion");
      return;
    endif
    p=p+(i-1); #posicion relativa al vector indx

    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_sindex(A(indx, :), b(indx));
  r_h=b-A*x
  T=toc();
endfunction
```

Función sustitución atrás usada en Gauss con pivoteo:

```
function [x] = sust_at_sindex(A, b)
  x=0*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 Graficar los métodos:

```
function [] = GraficaMetodos (A,b,w)
 n=length(A(1,:));

  [xJ,itJ,r_hJ,TJ] = Jaccobi(A,b,b*0,n,1e-5);
  [xG,itG,r_hG,TG] = GaussSeidel(A,b,b*0,n,1e-5);
  [xS,itS,r_hS,TS] = SOR(A,b,b*0,w,n,1e-5);
  [xGC,itGC,r_hGC,TGC] = GradienteConjugado(A,b,b*0,n,1e-5);

  semilogy(1:itJ,r_hJ);
  hold on;
  semilogy(1:itG,r_hG);
  semilogy(1:itS,r_hS,"*");
  semilogy(1:itGC,r_hGC);
  legend("Jacobi","Gauss Seidel","SOR","Gradiente Conjugado");

  xlabel("Numero de Iteraciones");
  ylabel("Residuo-norma infinito");
  hold off;

endfunction
```

Función para calcular la matriz Tj:

```
function [Tj] = CalculaTj (A)

  D=diag(diag(A));
  L=tril(A)-D;
  U=triu(A)-D;
  Tj=inv(D)*(L+U);

endfunction
```

Funcion para calcular la matriz Tg:

```
function [Tg] = CalculaTg (A)

  D=diag(diag(A));
  L=tril(A,-1);
  U=triu(A,1);
  Tg=-inv(D+L)*(U);

endfunction
```

Función para calcular Tw:

```
function [Tw] = CalculaTw (A,w)

  D=diag(diag(A));
  L=tril(A)-D;
  U=triu(A)-D;
  Tw=inv(D-w*L)*((1-w)*D+w*U);

endfunction
```

Especificaciones de la máquina que se usó para los cálculos:

Ver información básica acerca del equipo

Edición de Windows

Windows 10 Pro

© 2016 Microsoft Corporation. Todos los derechos reservados.

Sistema

Procesador:	Intel(R) Pentium(R) CPU P6100 @ 2.00GHz 2.00 GHz
Memoria instalada (RAM):	3.00 GB (2.73 GB utilizable)
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil:	La entrada táctil o manuscrita no está disponible para esta pantalla