MÉTODOS ITERATIVOS

Cálculo Numérico

2017

Juan Ignacio Ferrer

Juan Ignacio Ferrer
DNI: 35.749.374

Introducción:

En este ejercicio propuesto por la cátedra, bajo el marco de métodos iterativos para resolución de SEALs, se tiene por objetivo comparar la eficiencia, medida en número de iteraciones y tiempos de ejecución, de los distintos métodos cubiertos en clase, para luego definir cuáles de ellos son más conveniente para distintas situaciones. Además, se realizará una comparación entre métodos iterativos y directos, como es el de Gauss, para matrices de un tamaño considerable, y se obtendrán conclusiones a partir de los resultados obtenidos.

A continuación, un marco teórico para los métodos iterativos:

Métodos Iterativos para resolución de SEALs

La resolución de un Sistema de Ecuaciones Algebraicas Lineales (SEAL):

$$A\boldsymbol{x} = b$$

donde:

- $\boldsymbol{x}$ es un vector conteniendo n incógnitas
- A es una matriz cuadrada de n x n con los coeficientes del sistema
- **b** un vector de términos independientes

puede realizarse por distintos métodos. Estos suelen clasificarse en: Métodos Directos y Métodos Iterativos.

Un método iterativo es un método que **progresivamente** va calculando **aproximaciones** a la solución de un problema. En un método iterativo se repite un mismo proceso de mejora sobre una solución aproximada: se espera que lo obtenido sea una *solución más aproximada* que la inicial. El proceso se repite sobre esta nueva solución hasta que el resultado más reciente satisfaga ciertos requisitos.

A diferencia de los métodos directos, en los cuales se debe terminar el proceso para obtener una respuesta, en los métodos iterativos se puede suspender el proceso al término de una iteración y se obtiene una aproximación a la solución. Se hace hincapié en esto ya que es un elemento en contra de los métodos iterativos: sólo calculan *aproximaciones*.

Las ventajas de estos métodos es que se pueden utilizar cuando no se conoce un método para obtener la solución en forma exacta. También se utilizan cuando el método para determinar la solución exacta requiere mucho tiempo de cálculo, cuando una respuesta aproximada es adecuada, y cuando el número de iteraciones es relativamente reducido.

Método Iterativo General:

Un método iterativo cuenta con los siguientes pasos:

1- Inicia con una solución aproximada (semilla o seed)
2- Ejecuta una serie de cálculos para obtener o construir una mejor aproximación partiendo de la aproximación semilla. La fórmula que permite construir la aproximación usando otra se conoce como **ecuación de recurrencia**.
3- Se repite el paso anterior, pero usando como semilla la aproximación obtenida.

En clase han sido cubiertos ciertos métodos iterativos clásicos, así como uno basado en técnicas de optimización (métodos de iteración por subespacios). Éstos son:

- Método de Jacobi
- Método de Gauss-Seidel
- Método de SOR
- Método del Gradiente Conjugado

Método de Jacobi

El método de Jacobi es el método iterativo más simple y se aplica a sólo a sistemas cuadrados, es decir, a sistemas con tantas incógnitas como ecuaciones.

La matriz *A* puede descomponerse de manera aditiva:

$A = L_s + D_s + U_s$

donde:

- D_s es la matriz diagonal con la diagonal de *A* ($d_{ii} = a_{ii}$)
- L_s es la matriz triangular inferior con los términos de A excluyendo la diagonal ($l_{ij} = a_{ij}$ para $i > j$)
- U_s es la matriz triangular superior con los términos de A excluyendo la diagonal ($u_{ij} = a_{ij}$ para $i > j$)

La iteración de Jacobi puede escribirse:

$$\boldsymbol{M x^{(k)} = N x^{(k-1)} + b}$$

con

$$\boldsymbol{M = D_s}$$

$$\boldsymbol{N = -(L_s + U_s)}$$

En la iteración *k,* la componente *i* de **x** se calcula:

$$x_i^{(k)} = \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k-1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right] / a_{ii}$$

Convergencia del método:

Uno de los principales problemas de los métodos iterativos es la garantía de que el método va a converger, es decir, va a producir una sucesión de aproximaciones cada vez efectivamente más próximas a la solución. En el caso del método de Jacobi no existe una condición exacta para la convergencia. Sin embargo, se cuenta con una condición que garantiza la convergencia, pero en caso de no cumplirse puede o no haberla, y es la siguiente:

Si una matriz **A** es **diagonalmente dominante**, entonces **el método de Jacobi converge para cualquier vector inicial**.

Recordar: una matriz se dice diagonalmente dominante si $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$

En ciertas ocasiones al aplicar Jacobi la matriz no es diagonalmente dominante y por tanto no existirá garantía de convergencia. Sin embargo, en algunos casos será posible reordenar las incógnitas de otra manera de forma que la nueva matriz de coeficientes sea diagonalmente dominante. Esto se puede detectar revisando todos los posibles ordenamientos de las incógnitas y evaluar la matriz resultante. Sin embargo, esto conlleva un gran número de pruebas, ya que el número posible de ordenamientos en n variables es $(n-1)!$, de modo que es conveniente realizarlo sólo cuando n es reducido.

Método de Gauss-Seidel

El método de Gauss-Seidel es muy semejante al método de Jacobi. Mientras que en el de Jacobi se utiliza el valor de las incógnitas para determinar una nueva aproximación, en el de Gauss-Seidel se va utilizando los valores de las incógnitas recién calculados en la misma iteración, y no en la siguiente.

Dada *A* descompuesta en forma aditiva:

$$A = L_s + D_s + U_s$$

la fórmula de iteración Gauss-Seidel:

$$\boldsymbol{M x^{(k)} = N x^{(k-1)} + b}$$

con

$$\boldsymbol{M = D_s + L_s}$$

$$\boldsymbol{N = -U_s}$$

En la iteración *k,* la componente *i* de **x** se calcula:

$$x_i{}^{(k)} = \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j{}^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j{}^{(k-1)} \right] / \, a_{ii}$$

Convergencia del método:

Del mismo modo que para el método de Jacobi, existen condiciones suficientes (pero no necesarias) que garantizan la convergencia del método para cualquier vector inicial:

- Si una matriz es **diagonalmente dominante,** entonces el método de Gauss-Seidel converge.
- Si una matriz es **simétrica**, **definida positiva**, entonces el método de Gauss-Seidel converge.

Recordar: una matriz **A** se dice definida positiva si:

$$\boldsymbol{x^T A x > 0 \; \forall x \in \mathbb{R}^n, x \neq 0}$$

Además, una matriz es definida positiva si y solo si todos sus autovalores son positivos.

Método de Sobre-relajaciones Sucesivas (SOR)

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

Dada *A* descompuesta en forma aditiva:

$$A = L_s + D_s + U_s$$

la fórmula de iteración SOR:

$$\boldsymbol{M_\omega x^{(k)} = N_\omega x^{(k-1)} + \omega b}$$

con

$$\boldsymbol{M = D_s - \omega L_s}$$

$$\boldsymbol{N = (1-\omega) D_s + \omega U_s}$$

donde ω es un parámetro propio del método.

En la iteración *k,* la componente *i* de **x** se calcula:

$$x_i^{(k)} = \omega \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right] / a_{ii} + (1-\omega) x_i^{(k-1)}$$

Características del método:

- Si $\omega = 1$, el método de SOR es igual al de Gauss-Seidel.
- Si $\rho\left(\boldsymbol{M_\omega}^{-1} \boldsymbol{N_\omega}\right) < 1$, SOR converge.
- Si $a_{ii} \neq 0 \; \forall \boldsymbol{i}$, entonces $\rho\left(\boldsymbol{M_\omega}^{-1} \boldsymbol{N_\omega}\right) \leq |\omega - 1|$, y luego $0 < \omega < 2$.
- Si $0 < \omega < 1 \rightarrow$ método de sub-relajación.
- Si $1 < \omega < 2 \rightarrow$ método de sobre-relajación.
- Para una matriz se puede hallar un ω óptimo que minimice el radio espectral.

Recordar que ρ es el radio espectral de una matriz **A**, definido como:

$$\rho(\boldsymbol{A}) = \max\{ |\lambda| \mid \det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0 \}$$

donde λ son los autovalores de **A**, soluciones de la ecuación:

$$\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$$

Entonces, $\rho(\boldsymbol{A})$ es el modulo del mayor autovalor, o sea, el radio del menor círculo del plano complejo, que contiene a todos los autovalores.

Método del Gradiente Conjugado

Los métodos iterativos mencionados anteriormente son sencillos y fáciles de aplicar. Sin embargo, su convergencia, en general, es lenta y no son muy utilizados en la práctica.

Por otro lado, los métodos directos resultan muy costosos para resolver grandes sistemas de ecuaciones.

Para estos casos se utilizan otros tipos de métodos iterativos que pueden ser designados como métodos de iteración por subespacios, o métodos de Krylov.

Entre ellos se encuentra el Método del Gradiente Conjugado. Este método, que puede ser mirado como un método directo, ya que en un número finito de pasos conduce a la solución exacta, en la práctica se utiliza como un método iterativo ya que realiza una cantidad -relativamente- pequeña de iteraciones hasta lograr la convergencia deseada.

Sin embargo, el Método del Gradiente Conjugado permite resolver matrices simétricas y definidas positivas.

Para llevar a cabo este método, se define una *forma cuadrática* asociada a este sistema:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^{\boldsymbol{T}}\boldsymbol{x}$$

donde ϕ es un escalar.

Se puede demostrar que el vector **x** que minimiza ϕ es, a su vez, solución de $A\boldsymbol{x} = b$.

Entonces, en lugar de buscar el vector solución de ese sistema, se busca el **vector que minimiza la forma cuadrática**.

Hay métodos que buscan un mínimo de ϕ con iteraciones

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)}$$

Dado $x^{(k)}$ en la iteración *k*, los algoritmos buscan una dirección de descenso $\boldsymbol{p}^{(k)}$ y un paso $\alpha^{(k)}$ a lo largo de esa dirección, para hallar la nueva iteración.

En la iteración *k* se define el residuo como:

$$\boldsymbol{r}^{(k)} = (b - A\boldsymbol{x}^{(k)})$$

de modo que se busca que el residuo tienda a cero.

Por último, se procederá a detallar la obtención de la dirección de descenso $\boldsymbol{p}$ y el paso α en dicha dirección:

En la iteración *k* se calcula la nueva iteración:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(0)} + \alpha^{(0)}\boldsymbol{p}^{(0)} + \alpha^{(1)}\boldsymbol{p}^{(1)} + \cdots + \alpha^{(k)}\boldsymbol{p}^{(k)}$$

La dirección de descenso $\boldsymbol{p}^{(k)}$ se busca tal que sea *conjugada* a todas las anteriores.

Recordar que dos direcciones se dicen conjugadas si el doble producto para dos direcciones diferentes es cero. Si se trata de la misma dirección, sería un número positivo debido a que la matriz es definida positiva.

La nueva dirección se calcula con la fórmula:

$$\boldsymbol{p}^{(k+1)} = \boldsymbol{r}^{(k+1)} + \gamma^{(k)}\boldsymbol{p}^{(k)}$$

Introduciendo esta ecuación en la condición de conjugadas:

$$(\boldsymbol{p}^{(k+1)})^{T}\boldsymbol{A}\boldsymbol{p}^{(k)} = 0$$

podemos despejar $\gamma^{(k)}$ que resulta:

$$\gamma^{(k)} = -\frac{\boldsymbol{r}^{(k+1)^{T}}\boldsymbol{A}\,\boldsymbol{p}^{(k)}}{\boldsymbol{p}^{(k)^{T}}\boldsymbol{A}\,\boldsymbol{p}^{(k)}}$$

Para calcular el paso de avance α , se escribe ϕ como función del paso:

$$\phi\left(\boldsymbol{x}^{(k+1)}\right) = \phi\left(\boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)}\right) = \tilde{\phi}(\alpha^{(k)})$$

El paso $\alpha^{(k)}$ se calcula tal que minimice $\tilde{\phi}(\alpha^{(k)})$:

$$\frac{\mathrm{d}\tilde{\phi}(\alpha^{(k)})}{\mathrm{d}\alpha^{(k)}} = \boldsymbol{p}^{(k)^T}\frac{\partial\phi}{\partial\boldsymbol{x}^{(k)}} = \boldsymbol{p}^{(k)^T}\left[\boldsymbol{A}\left(\boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)}\right) - \boldsymbol{b}\right] = 0$$

$$\left(\boldsymbol{p}^{(k)^T}\boldsymbol{A}\,\boldsymbol{p}^{(k)}\right)\alpha^{(k)} = \boldsymbol{p}^{(k)^T}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(\boldsymbol{k})}\right) = \boldsymbol{p}^{(\boldsymbol{k})^T}\boldsymbol{r}^{(\boldsymbol{k})}$$

siendo $\boldsymbol{r}^{(k)}$ el residuo en la iteración *k*.

De aquí, el valor del paso se calcula:

$$\alpha^{(k)} = \frac{\boldsymbol{p}^{(k)^T}\boldsymbol{r}^{(k)}}{\boldsymbol{p}^{(k)^T}\boldsymbol{A}\,\boldsymbol{p}^{(k)}}$$

A continuación, el enunciado del 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. 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, tasa 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 ud. que es el más conveniente. ¿Cómo justifica que los métodos iterativos logren o no convergencia? ¿Qué expectativa puede tener sobre la precisión de los resultados obtenidos? Las entradas de la matriz de coeficientes del sistema lineal son,

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

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

Procedemos, en primer lugar, a crear un algoritmo que genere una matriz $N\,x\,N$ con las características planteadas, recibiendo como parámetro un número N.

A continuación, el algoritmo:

```
function [A,b] = generarMatriz(n)
 A=zeros(n)+
 diag(2:2:2*n+
 diag(0.5:0.5:0.5*(n-2),2+
 diag(0.25:0.25:0.25*(n-4),4+
 diag(1.5:0.5:0.5*n, -2)+
 diag(1.25:0.25:0.25*n, -4);
 b=pi*ones(n,1);
endfunction
```

Caracterización de las matrices:

Para los N dados, las matrices presentarán las siguientes características:

No son simétricas, ya que $A^T \neq A$.

Son definidas positivas. Esto se demuestra calculando la matriz de autovalores mediante la función eig(A) de Octave, y buscando el mínimo autovalor en la diagonal. **Si el mínimo autovalor es positivo, todos lo serán**. Para esto, se crea una función menorAutovalor de la siguiente manera:

```
function x = menorAutovalor(A)
  [V,lambda] = eig(A);
  x = lambda(1,1);
  for(i=2 : length(lambda))
    if(x > lambda(i,i))
      x = lambda(i,i);
    endif
  endfor
endfunction
```

De este modo, verificamos que el mínimo autovalor es positivo para las tres matrices del enunciado, y obtenemos:

```
>> [A1,b1] = generarMatriz(250);
>> x1 = menorAutovalor(A1)
x1 =  1.8097
>>
>> [A2,b2] = generarMatriz(500);
>> x2 = menorAutovalor(A2)
x2 =  1.8097
>>
>>
>> [A3,b3] = generarMatriz(1000);
>> x3 = menorAutovalor(A3)
x3 =  1.8097
```

En todos los casos obtenemos un autovalor positivo, por lo que se concluye que las matrices son definidas positivas.

Son diagonalmente dominantes. Esto es así ya que, dada la forma en la que se genera la matriz, para cualquier fila *i* sucede que:

$$|\alpha_{ii}| = |2i| = 2i$$

$$\sum_{j \neq i} |a_{ij}| = |2 \text{ x } 0.5i| + |2 \text{ x } 0.25i| = 1.5i$$

Y entonces:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

por lo que las matrices serán diagonalmente dominantes.

Debido a esto, por teorema, las **matrices serán no singulares**, por lo que poseerán solución única. Además, se puede garantizar la convergencia de los métodos de Jacobi y Gauss-Seidel.

Juan Ignacio Ferrer
DNI: 35.749.374

Algoritmos y resultados:

El algoritmo **jacobi** recibe como parámetros la matriz *A*, el vector de términos independientes *b*, una aproximación inicial x_0 que actuará como semilla, un número máximo de iteraciones *maxit* y la tolerancia en el residuo *tol*, y devuelve la solución *x*, la cantidad de iteraciones *it* y la convergencia del método *r*, además del tiempo que le tomó al algoritmo realizar los cálculos.

Para todas las aplicaciones de los algoritmos se fijará la tolerancia en el residuo en $\mathbf{10^{-5}}$, el número máximo de iteraciones será de **100** y la aproximación inicial será el vector x_0 de tamaño N lleno de ceros.

A continuación, el algoritmo:

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

Aplicando *jacobi* a las matrices generadas por N = 250, 500, 1000 obtenemos los siguientes resultados:

N = 250

```
>> [x,r,it,t] = jacobi(A1,b1,x0,maxit,tol);
>> t
t =  0.43760
>> it
it =  23
```

N = 500

```
>> [x,r,it,t] = jacobi(A2,b2,x0,maxit,tol);
>> t
t =  0.65747
>> it
it =  23
```

N = 1000

```
>> [x,r,it,t] = jacobi(A3,b3,x0,maxit,tol);
>> t
t =  1.3460
>> it
it =  23
```

En todos los casos se obtuvo los respectivos vectores solución **x** y de error **r** que no se muestran aquí debido a su gran tamaño, pero se compararán más adelante mediante gráficas semilogarítmicas.

Ahora, se mostrará el algoritmo **gauss_seidel**:

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

Aplicando *gauss_seidel* a las matrices generadas por N = 250, 500, 1000 obtenemos los siguientes resultados:

N = 250

```
>> [x,r,it,t] = gauss_seidel(A1,b1,x0,maxit,tol);
>> t
t =  0.090061
>> it
it =  6
```

N = 500

```
>> [x,r,it,t] = gauss_seidel(A2,b2,x0,maxit,tol);
>> t
t =  0.18987
>> it
it =  6
```

N = 1000

```
>> [x,r,it,t] = gauss_seidel(A3,b3,x0,maxit,tol);
>> t
t =  0.39026
>> it
it =  6
```

A simple vista se observa una convergencia más rápida que con el método de Jacobi, y en menos iteraciones.

A continuación, el algoritmo para el método SOR y los resultados que éste arrojó:

Aclaración: para elegir ω, se tiene en cuenta que:

- Si $\omega = 1$, el método es el mismo que Gauss-Seidel. Si $\omega \in (0,1)$ se tendrá un método de sub-relajación que será útil cuando Gauss-Seidel no converja.
- Si $\omega \in (1,2)$ se tendrá un modelo de sobre-relajación que será útil para acelerar la convergencia de Gauss-Seidel.

Se sabe que en este caso Gauss-Seidel converge, de modo que se elegirá $\boldsymbol{\omega = 1.25}$, un valor que, en principio, optimizará la convergencia.

```
function [x,r,it,t] = sor(A,b,x0,maxit,tol,w)
  tic();
  n=length(b);
  x=x0;
  it=0;
  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(it+1)=norm(x0-x,'inf')/norm(x,'inf');
    if r(it+1)<tol break; endif
    x0=x;
    it=it+1;
  endwhile
  t=toc();
endfunction
```

N = 250

```
>> [x,r,it,t] = sor(A1,b1,x0,maxit,tol,w);
>> t
t =  0.17813
>> it
it =  10
```

N = 500

```
>> [x,r,it,t] = sor(A2,b2,x0,maxit,tol,w);
>> t
t =  0.36324
>> it
it =  10
```

N = 1000

```
>> [x,r,it,t] = sor(A3,b3,x0,maxit,tol,w);
>> t
t =  0.75983
>> it
it =  10
```

A continuación, se muestra el algoritmo del método del Gradiente Conjugado, y los resultados que éste retorna:

```
function[x,it,r_h,t]=gradienteConjugado(A,b,x0,maxit,tol)
  tic();
  x=x0;
  r=b-A*x;
  producto_r=(r'*r);
  v=r;
  for it=1:maxit
    producto_Av=(A*v);
    t=producto_r/(v'*producto_Av);
    x0=x;
    x=x0-t*v;
    r=r-t*producto_Av;
    r_h(it)=norm(r,'inf')/norm(b,'inf');
    if(r_h(it))<tol
      break;
    endif
    producto_r2=producto_r;
    producto_r=(r'*r);
    s=producto_r/producto_r2;
    v=r+s*v;
  endfor
  t=toc();
endfunction
```

N = 250

```
>> [x,it,r,t] = gradienteConjugado(A1,b1,x0,maxit,tol);
>> t
t =  0.11335
>> it
it =  100
```

N = 500

```
>> [x,it,r,t] = gradienteConjugado(A2,b2,x0,maxit,tol);
>> t
t =  0.019015
>> it
it =  100
```

N = 1000

```
>> [x,it,r,t] = gradienteConjugado(A3,b3,x0,maxit,tol);
>> t
t =  0.098568
>> it
it =  100
```

Por último, realizaremos la resolución de los SEALs mediante el método directo de eliminación gausseana, para ver si realmente hay una diferencia considerable:

Al ser matrices *muy grandes y diagonalmente dominantes*, es más conveniente no realizar pivoteo en este caso, debido a que crecerá demasiado el costo de los cálculos para el intercambio de filas, siendo éste no necesario.

El algoritmo de dicho método es el siguiente:

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

Y los resultados:

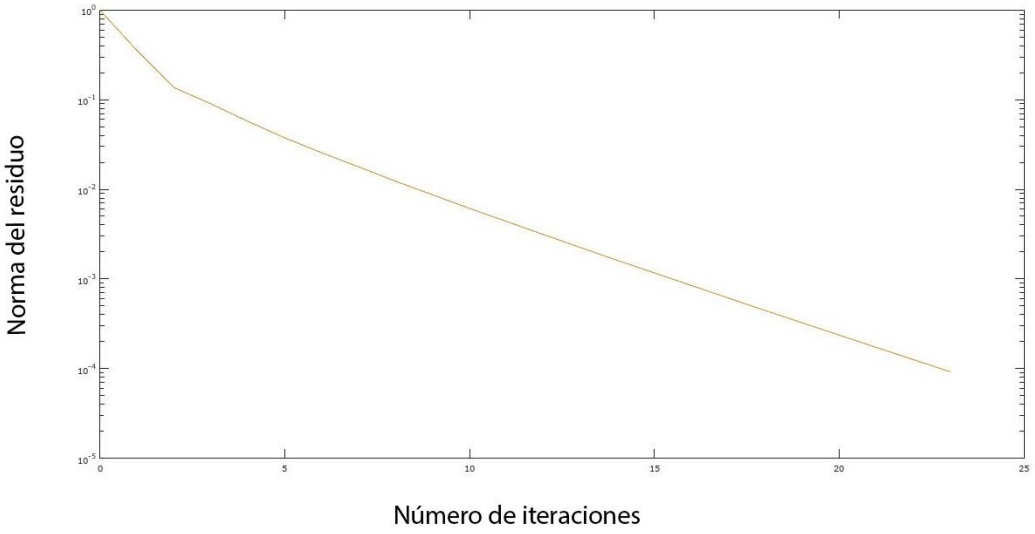
```
>> [x,t] = gauss(A1,b1);
>> t
t =  0.051035
>> [x,t] = gauss(A2,b2);
>> t
t =  0.59372
>> [x,t] = gauss(A3,b3);
>> t
t =  5.8500
```

Gráficas:

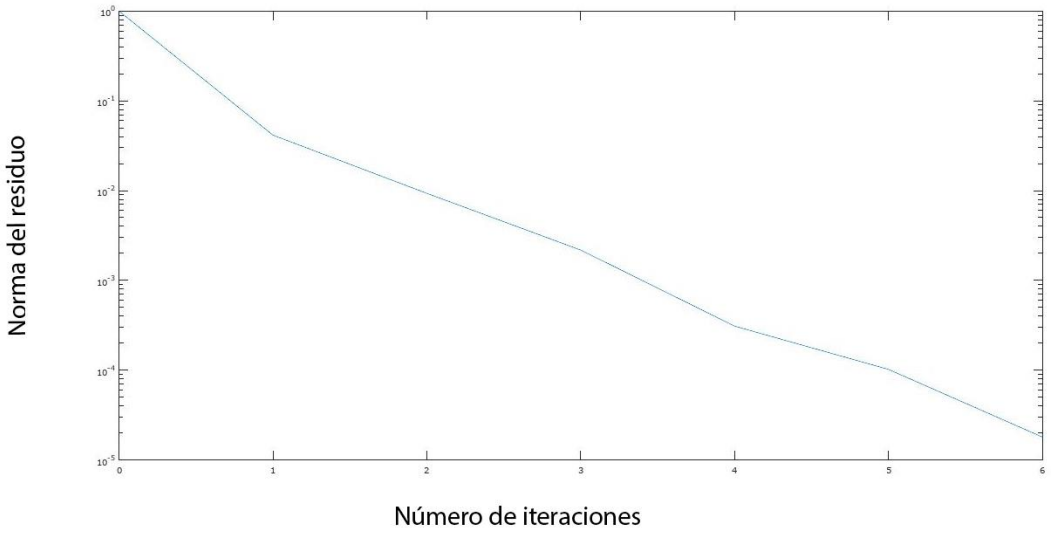
Se procede a realizar una gráfica por cada método, para evaluar la convergencia comparando *norma del residuo vs. número de iteraciones*. Las gráficas poseerán un eje vertical logarítmico y un eje horizontal lineal, por lo que se empleará la función de graficación *semilogy* de Octave para todos los casos.

A continuación, las gráficas:

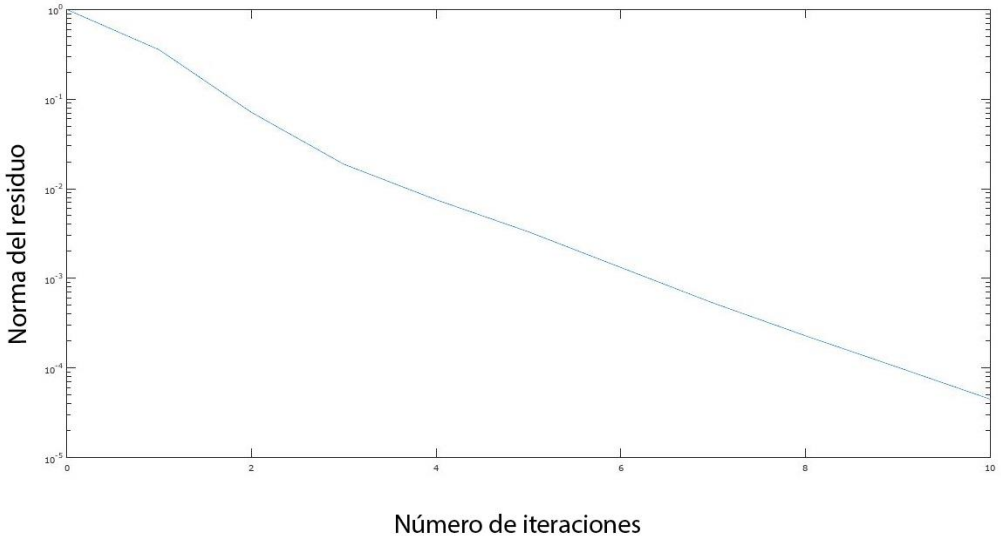
Gráfica 1: Norma del residuo vs. Número de iteraciones
Método de Jacobi



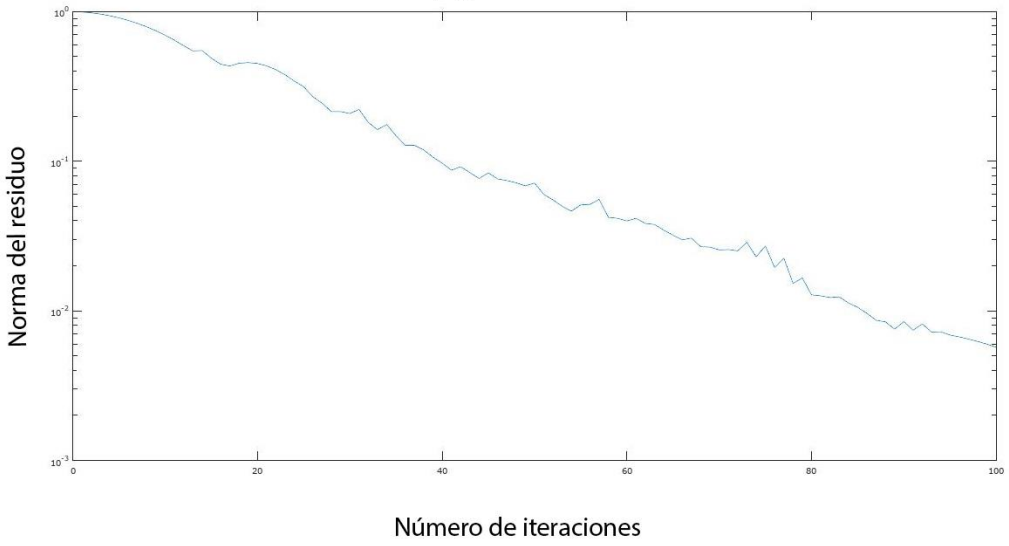
Gráfica 2: Norma del residuo vs Número de iteraciones
Método de Gauss-Seidel



Gráfica 3: Norma del residuo vs. Número de iteraciones
Método de Sobre-relajaciones Sucesivas (SOR)

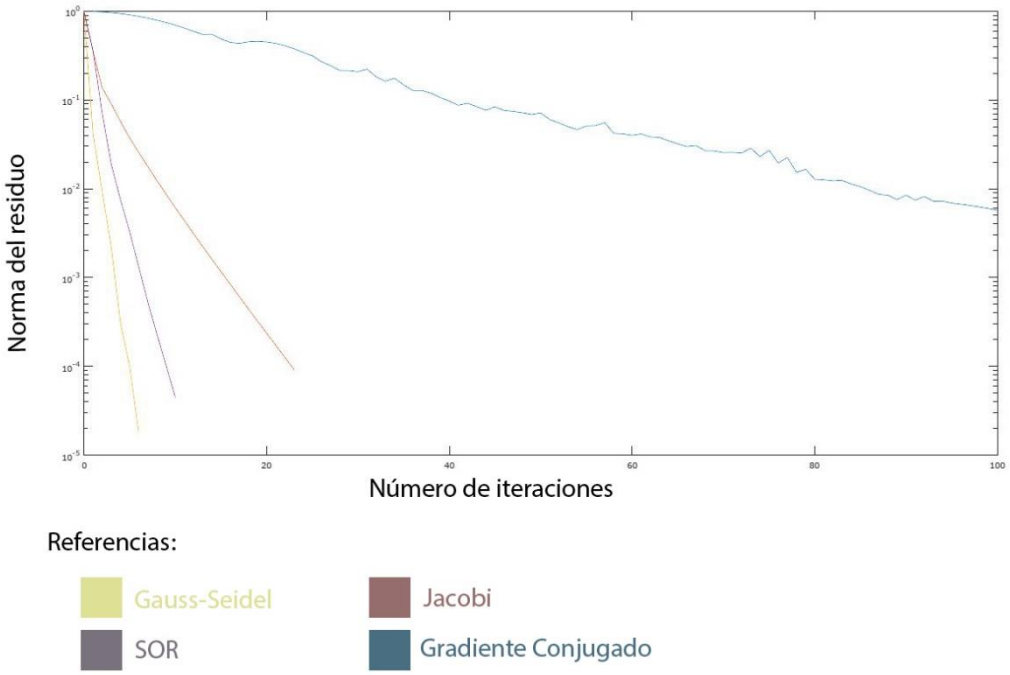


Gráfica 4: Norma del residuo vs. Número de iteraciones
Método del Gradiente Conjugado



Finalmente, se integran todas las gráficas en una sola para realizar una comparación más clara de las convergencias en cada caso. En Octave, para realizar múltiples curvas en el mismo gráfico, hago uso del comando *hold on*. La gráfica, entonces, es la siguiente:

Gráfica 5: Norma del residuo vs. Número de iteraciones
Convergencia de Métodos Iterativos



Conclusión:

Luego de haber implementado todos los métodos iterativos mencionados, realizamos las siguientes comparaciones:

Matriz N = 250		
	t (s)	it
Jacobi	0,43	23
Gauss-Seidel	0,090061	6
SOR	0,17813	10
Gradiente Conjugado	0,11335	100

Matriz N = 500		
	t (s)	it
Jacobi	0,65747	23
Gauss-Seidel	0,18987	6
SOR	0,36324	10
Gradiente Conjugado	0,019015	100

Matriz N = 1000		
	t (s)	it
Jacobi	1,346	23
Gauss-Seidel	0,39026	6
SOR	0,75983	10
Gradiente Conjugado	0,098568	100

Dentro de los métodos convergentes, podemos ver que el de **Gauss-Seidel fue el mejor**, ya que le tomó el menor tiempo arribar a una solución aproximadamente aceptable, y lo hizo con el menor número de iteraciones. Con respecto a este método, se había predicho la convergencia al comprobar que las matrices eran diagonalmente dominantes.

En segundo lugar, situamos al método SOR, al cual le tomó un tiempo superior que al método de Gauss Seidel, y lo hizo con 10 iteraciones, habiendo elegido un $\omega = 1.25$.

Por último, se posiciona el método de Jacobi, el cual convergió más lento con 23 iteraciones. La convergencia de éste estaba asegurada por las matrices diagonalmente dominantes.

Con respecto al método del Gradiente Conjugado, observamos que en todos los casos llegó al máximo de iteraciones sin acercarse al error tolerado de 10^{-5}. Esto podía pasar debido a que no se podía asegurar la convergencia del mismo, ya que las matrices no son simétricas.

¡Sin embargo! Podemos observar que le tomó, relativamente, mucho menos tiempo realizar muchas más iteraciones. Entonces, de haber sido simétricas las matrices, además de definidas positivas, es muy probable que este método hubiera sido el mejor.

Finalmente, si se compara el mejor método iterativo (en este caso, Gauss-Seidel) contra el método de Gauss, observamos que los tiempos de arribo a una solución son muy distintos:

Si bien Gauss encuentra la solución exacta (más allá del error de redondeo de la computadora) y Gauss-Seidel una solución aproximada, para aquellos casos donde sea tolerable dicho error, Gauss-Seidel es muy superior. Esto es así, especialmente, en matrices de gran tamaño: para el caso de N = 1000, al método directo le tomó 5.45974 segundos más que al método iterativo arribar a la solución, un tiempo muy grande. Si esta diferencia se traduce a matrices con N del orden del millón o superior, puede marcar una diferencia abismal.

En conclusión, siempre que sea aceptable una solución aproximada a un SEAL y la matriz sea diagonalmente dominante, es recomendable resolverla por métodos iterativos, en especial Gauss-Seidel. En el caso de que la matriz sea simétrica definida positiva, el método más eficaz es el del Gradiente Conjugado.

Introducción:

Ahora, lo que se pide es resolver dos SEALs iguales, salvo por un cambio en el orden de las filas, mediante los métodos de Gauss-Seidel (iterativo) y el de Gauss (directo) y analizar los resultados.

Ya se ha introducido un marco teórico, en este trabajo, sobre los métodos que serán empleados, además de la explicitación de los algoritmos correspondientes a estos.

A continuación, el enunciado:

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.

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

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

Se realiza la caracterización de ambas matrices:

Matriz del SEAL (1):

- **No es simétrica**, ya que $A^T \neq A$.
- **No es definida positiva**. Esto es así ya que, al aplicar la función *menorAutovalor*, detallada anteriormente en este TP, obtenemos:

```
>> auto = menorAutovalor(A)
auto = -5.2749
```

  Al haber al menos un autovalor negativo, la matriz no es definida positiva.
- **Es diagonalmente dominante**. Se cumple esto ya que, evaluando todas las filas:
  Fila 1:

$$|\alpha_{ii}| = |3| = 3$$

$$\sum_{j \neq i} |a_{ij}| = |1| + |1| = 2$$

  Fila 2:

$$|\alpha_{ii}| = |3| = 3$$

$$\sum_{j \neq i} |a_{ij}| = |1| + |-1| = 2$$

Fila 3:

$$|\alpha_{ii}| = |-5| = 5$$

$$\sum_{j \neq i} |a_{ij}| = |3| + |1| = 4$$

De modo que para todas las filas se cumple que:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

Al ser diagonalmente dominante, la matriz es no singular y cuenta con única solución.

Matriz del SEAL (2)

- **No es simétrica**, ya que $A^T \neq A$.
- **No es definida positiva**. Esto es así ya que, al aplicar la función *menorAutovalor* obtenemos:

```
>> auto = menorAutovalor(B)
auto = -0.37421 - 3.55884i
```

Al haber al menos un autovalor negativo, la matriz no es definida positiva.

- **No es diagonalmente dominante**. Si evaluamos la fila 2:

$$|\alpha_{ii}| = |1| = 1$$

$$\sum_{j \neq i} |a_{ij}| = |3| + |-5| = 8$$

Entonces, para esta fila:

$$|a_{ii}| < \sum_{j \neq i} |a_{ij}|$$

y la matriz no es diagonalmente dominante.

Entonces, aplicando el algoritmo *gauss_seidel* (mostrado anteriormente en este trabajo) al SEAL (1) con un máximo de iteraciones de 100 y una tolerancia de 10^{-5}, se obtiene:

```
>> [x,r,it,t] = gauss_seidel(A,b1,x0,maxit,tol);
>> x
x =

   1.00001
   0.99999
   1.00001

>> it
it =  8
>> t
t =  0.0019989
```

El algoritmo converge en 8 iteraciones a la solución **aproximada**.

Realizamos el mismo cálculo mediante el algoritmo *gauss*, también explicitado anteriormente en el trabajo:

```
>> [x, t] = gauss(A,b1)
x =

   1.0000
   1.0000
   1.0000

t =  0.0010018
```

El algoritmo halla la solución única, **exacta**, aún más rápido que el algoritmo iterativo. Esto es así ya que el SEAL es pequeño. Como concluimos anteriormente, *gauss* es pésimo para hallar soluciones de matrices muy grandes.

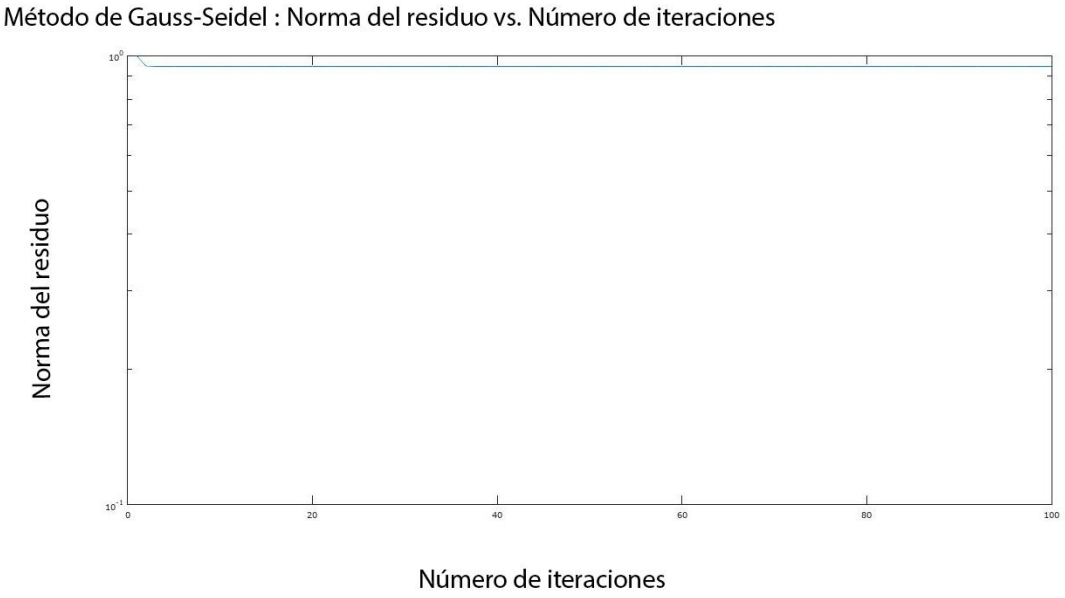
Ahora, ejecutamos el algoritmo *gauss_seidel* para el SEAL (2), obteniendo:

```
>> [x,r,it,t] = gauss_seidel(B,b2,x0,maxit,tol);
>> x
x =

   2.0812e+125
  -2.9516e+126
  -8.6468e+126

>> it
it =  100
>> t
t =  0.019478
```

Vemos que, si bien el SEAL (2) es igual al SEAL (1) con un intercambio de renglones, la solución es radicalmente distinta. Para ilustrar mejor la no-convergencia del método, se presenta una gráfica semilogarítmica de *norma del residuo vs número de iteraciones*:



El método nunca tiende a la solución aproximada, indistintamente de las iteraciones que realice.

Intentamos hallar la solución exacta del SEAL (2) mediante el método *gauss*:

```
x =

   NaN
   NaN
   NaN

t =   0.0015011
```

Entonces, si intentamos resolver este SEAL mediante el método directo sin pivoteo parcial, no arribamos a una solución debido a que en **el procedimiento se producen divisiones por cero**.

De este modo, deberíamos aplicar **pivoteo parcial** para obtener una matriz distinta que pueda ser resuelta por el método. En este caso, aplicando pivoteo parcial transformaríamos la matriz del SEAL (1) en la matriz del SEAL (2), el cual fue resuelto anteriormente y se arribó a una solución única.

Así, aplicando *gauss_p* (algoritmo correspondiente al método de Gauss realizando pivoteo parcial) obtengo:

```
>> x = gauss_p(A,b1)
x =

   1.0000
   1.0000
   1.0000

>> x = gauss_p(B,b2)
x =

   1.0000
   1.0000
   1.0000
```

Conclusión:

Este ejercicio mostró que dos SEALs iguales con renglones cambiados pueden presentar diferencias radicales a la hora de resolverlos, y la importancia de realizar pivoteo parcial en ciertos casos.

En el primer caso, tanto el método iterativo de Gauss-Seidel y como el método directo de Gauss arribaron a una solución única. El método de Gauss realizó el procedimiento más rápido que Gauss-Seidel debido a que el SEAL es pequeño, pero su eficacia decae para aquellos SEALs muy grandes.

En el segundo caso, ambos métodos fallaron en arribar a la solución. Por un lado, Gauss-Seidel nunca converge, indistintamente del límite de iteraciones que se le impongan. Por otro, Gauss muestra que no existe solución ya que se producen divisiones por cero en el procedimiento. Sin embargo, al realizar pivoteo parcial mediante el algoritmo de *gauss_p*, al realizar intercambio de renglones, el algoritmo obtiene el mismo SEAL que en el caso anterior, el cual fue resuelto con éxito.