

Sistemas de Ecuaciones Lineales

Laboratorio 4

El objetivo de este laboratorio es utilizar eficientemente métodos directos e iterativos para la solución de sistemas de ecuaciones lineales $\mathbf{Ax}=\mathbf{b}$.

1. En el siguiente ejercicio, se comprobará que $\mathbf{A} \setminus \mathbf{b}$ es equivalente a calcular la descomposición LU de \mathbf{A} con pivote parcial. Si \mathbf{A} es estrictamente diagonal dominante no será necesario realizar permutaciones de filas al calcular su descomposición LU con pivote parcial, es decir, $P = I$

1. Haga un programa *function* que genere una matriz tridiagonal de orden n de la forma

$$\mathbf{A} = \begin{pmatrix} a & b & & 0 \\ & \ddots & \ddots & \\ c & & \ddots & b \\ 0 & & c & a \end{pmatrix}. \quad (1)$$

Los datos de entrada deben ser los valores de n , a , b y c , y la salida la matriz \mathbf{A} .

2. Mediante el comando MATLAB **rank** (devuelve el rango de una matriz), determine si la matriz tridiagonal y simétrica \mathbf{A} correspondiente a $n = 10$, $a = 4$ y $b = c = 1$ es invertible. Note que con estos valores de a, b, c la matriz \mathbf{A} es estrictamente diagonal dominante.
3. Resuelva el sistema $\mathbf{Ax} = \mathbf{b}$ con la matriz tridiagonal anterior y un segundo miembro \mathbf{b} cualquiera, de los siguientes dos modos:

```
1. >> x=A\b
2. >> [L,U,P]=lu(A);
   >> x=U\ (L\b)
```

Compruebe que $P = I$ y $A = LU$.

4. Compare las soluciones obtenidas y calcule los residuos $\mathbf{r}=\mathbf{b}-\mathbf{Ax}$ respectivos.
2. Haga un programa *function*, que genere una matriz \mathbf{A} con el comando **rand** de tamaño 10.
 1. Con el comando **rand** genere un vector $\mathbf{b} \in \mathbb{R}^{10}$.
 2. Mediante el comando **rank**, determine si \mathbf{A} es invertible. Si no lo es, genere una nueva matriz con **rand** hasta obtener una matriz invertible.
 3. Resuelva el sistema $\mathbf{Ax} = \mathbf{b}$ con la matriz y el vector generados anteriormente, de los siguientes dos modos:

```
1. >> x=A\b
2. >> [L,U,P]=lu(A);
   >> x=U\ (L\ (P*b))
```

Compruebe que $PA = LU$.

4. Compare las soluciones obtenidas y calcule los residuos $\mathbf{r}=\mathbf{b}-\mathbf{Ax}$ respectivos.

2. En muchas aplicaciones es necesario resolver varios sistemas de ecuaciones $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$, con la misma matriz $\mathbf{A} \in \mathbb{R}^{n \times n}$ y distintos segundos miembros \mathbf{b}_i , $i = 1, \dots, m$. Para hacer esto en MATLAB resulta conveniente generar la matriz de segundos miembros

$$\mathbf{B} = \left[\begin{array}{c|c|c} \mathbf{b}_1 & \cdots & \mathbf{b}_m \end{array} \right] \in \mathbb{R}^{n \times m}$$

y resolver el sistema matricial $\mathbf{A}\mathbf{X} = \mathbf{B}$, cuya solución

$$\mathbf{X} = \left[\begin{array}{c|c|c} \mathbf{x}_1 & \cdots & \mathbf{x}_m \end{array} \right] \in \mathbb{R}^{n \times m}$$

es la matriz de vectores solución \mathbf{x}_i , $i = 1, \dots, m$, de los sistemas anteriores.

El siguiente programa MATLAB tiene por objeto verificar esto experimentalmente:

```
A=rand(50);
%comprobar si es invertible
while rank(A)~=50
    A=rand(50);
end
B=rand(50,100);

t0=cputime;
X=A\B;
t1=cputime-t0

t0=cputime;
for i=1:100
    Y(:,i)=A\B(:,i);
end
t2=cputime-t0

dif=norm(X-Y,inf)
```

1. Escriba y ejecute el programa anterior.
2. Analice lo que hace este programa.
3. Justifique la diferencia de tiempos de ejecución que se observa.
4. Utilice la sentencia MATLAB

```
>> X=A\B;
```

con una matriz \mathbf{B} adecuada para calcular \mathbf{A}^{-1} . Compare el resultado obtenido con el del comando MATLAB **inv**.

3. El objeto de este ejercicio es demostrar que **no es conveniente** resolver un sistema de ecuaciones mediante la inversa de la matriz del mismo.

Considere el siguiente programa MATLAB:

```
nn=[100:100:500];  
t1=[];  
t2=[];  
for n=nn  
    A=rand(n);  
    y=ones(n,1);  
  
    t0=cputime;  
    x=inv(A)*y;  
    t1=[t1 cputime-t0];  
  
    t0=cputime;  
    x=A\y;  
    t2=[t2 cputime-t0];  
end  
plot(nn,t1,'*-',nn,t2,'o-')
```

1. Analice lo que hace este programa, ejecútelo e indique qué alternativa es la más conveniente.

4. 1. Considere el siguiente programa que resuelve mediante el método de **Jacobi** un sistema de ecuaciones $\mathbf{Ax} = \mathbf{b}$ expresado en (1) con error menor que una tolerancia dada \mathbf{tol} . Parta de un dato inicial \mathbf{x}^0 nulo y utilice el criterio de detención estudiado en las clases teóricas. El programa también debe detenerse si se alcanza un número máximo de iteraciones \mathbf{maxit} sin que se satisfaga el criterio de convergencia.

Testee el método con la matriz del Ej. 1 con una tolerancia $\mathbf{tol} = 10^{-6}$. Verifique que la solución obtenida aproxima a la solución exacta con la tolerancia del error estipulada. Determine la cantidad de iteraciones realizadas.

2. Repita lo anterior con el método de **Gauss-Seidel**. Indique cuál de los dos métodos requiere menos iteraciones.

Observación:

Los códigos presentados a continuación son para el caso específico de la matriz \mathbf{A} del ejercicio 1.

No sirven para cualquier matriz.

```
% Deben definir A (matriz diagonal dominante) y b del ejercicio 1
% deben ingresar una variable tol=
% deben ingresar el numero maximo de iteraciones maxit=
n=length(A);
N=diag(diag(A)); % Jacobi
%N=tril(A);      % Gauss-Seidel comentar N de Jacobi
P=N-A;

x=zeros(n,1);
corr=1;
errest=1;
iter=1;

while errest>tol & iter<maxit
    iter=iter+1;
    x0=x;
    corr0=corr;
    x=N\(P*x0+b);
    corr=norm(x-x0,inf);
    normest=corr/corr0;
    if normest>=1
        error('norma de la matriz de iteracion > 1')
    end
    errest=normest/(1-normest)*corr;
end
disp('numero de iteraciones')
iter
```