# Métodos numéricos para sistemas de ecuaciones

# (Prácticas)

**Damián Ginestar Peiró**


UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

**UNIVERSIDAD POLITÉCNICA DE VALENCIA**

# Índice general

**2. Métodos directos para la resolución de ecuaciones 3**

2.1. Introducción . . . . . 3

2.2. Descomposición LU . . . . . 5

2.3. Permutación y reordenamiento . . . . . 8

2.3.1. Reordenamientos . . . . . 11

2.4. Descomposición QR . . . . . 13

2.5. Regresión lineal múltiple . . . . . 15

2.6. Proceso de Markov . . . . . 17

2.7. Datos faltantes . . . . . 20

2.8. Ejercicios . . . . . 24

# Práctica 2

# Métodos directos para la resolución de ecuaciones

## 2.1. Introducción

Ya hemo visto que el operador general para la resolución de sistemas que utiliza Matlab es \así escribiendo

```
x=A\b
```

se obtiene la solución del sistema

$$Ax = b\ .$$

Haciendo

```
A = pascal(3);
u = [3; 1; 4];
x = A\u
```

obtenemos

```
x =
      10
     -12
       5
```

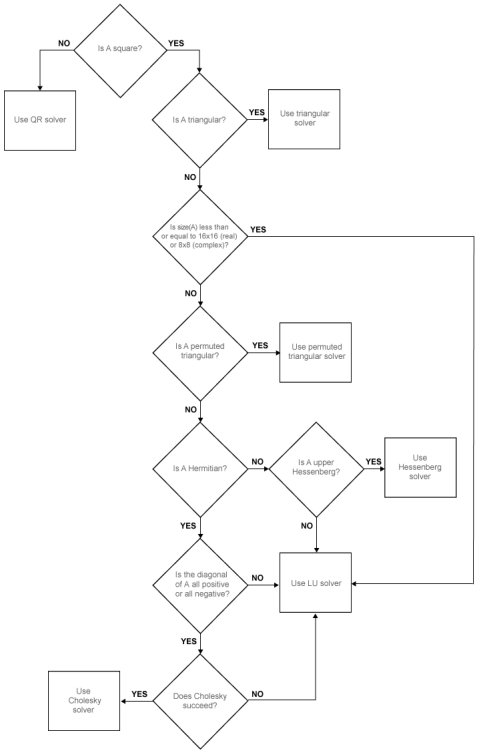
También se puede utilizar para resolver $n$ sistemas simultáneos. Así

```
B = magic(3);
X = A\B
```

proporciona

```
X =
      19    -3    -1
     -17     4    13
       6     0    -6
```

Antes de resolver el sistema, al utilizar el operador se hacem distintas comprobaciones para aplicar el método más eficiente. De este modo, se comprueba, por ejemplo, si la matriz es cuadrada o rectangular, si es triangular superior o inferior, si es simétrica y definida positiva. Asimismo se tiene en cuenta si el almacenamiento de la matriz es denso o disperso. (consultar la hoja `https://es.mathworks.com/help/matlab/ref/mldivide.html` para ver con detalle el árbol de decisión del operador \).



## 2.2. Descomposición LU

Uno de los métodos más utilizados para la resolución de sistemas de ecuaciones es el método de Gauss con pivotación parcial.

Un algoritmo básico de Matlab que realiza la factorización LU es el siguiente:

```
% triangularizacion de la matriz A
% se hacen ceros en las n-1 primeras columnas

for k=1:n-1
% se hacen ceros en la columna k
 for i=k+1:n
   m=A(i,k)/A(k,k);
% a la fila i se resta la fila k multiplicada por m
  for j=k+1:n
    A(i,j)=A(i,j)-m*A(k,j);
  end
% se transforma del mismo modo el termino independiente
  b(i)=b(i)-m*b(k);
 end
end
```

En una segunda fase se calculan las incógnitas mediante el método de sustitución regresiva:

```
% se calcula x(n) de la ultima ecuacion
x(n)=b(n)/A(n,n);
% se calcula x(k) de la ecuacion k
for k=n-1:-1:1
  s=0;
  for i=k+1:n
    s=s+A(k,i)*x(i);
  end
  x(k)=(b(k)-s)/A(k,k);
end
```

El algoritmo básico puede modificarse si la matriz $A$ es simétrica, haciendo que el número de operaciones se reduzca aproximadamente a la mitad:

```
% triangularizacion de la matriz A (simetrica)
% se hacen ceros en las n-1 primeras columnas

for k=1:n-1
% se hacen ceros en la columna k
  for i=k+1:n
    m=A(k,i)/A(k,k);
```

```
% se tiene en cuenta la simetria
% a la fila i se resta la fila k multiplicada por m
% solo se opera por encima de la diagonal
    for j=i:n
       A(i,j)=A(i,j)-m*A(k,j);
    end
% se transforma del mismo modo el termino independiente
    b(i)=b(i)-m*b(k);
  end
end
```

A continuación, mostramos dos funciones sencillas de Matlab que permiten la resolución de un sistema de ecuaciones lineales mediante este método. La primera función, `solvelu`, efecúa la triangularización de la matriz del sistema utilizando permutación parcial de las filas. La segunda función, `trisuper`, resuelve un sistema de ecuaciones cuya matriz de coeficientes es triangular superior mediante el método de sustitución regresiva, teniendo en cuenta las posibles permutaciones de filas efectuadas en el algoritmo de pivotación parcial.

```
function [An,bn,v]=solvelu(A,b)
%  [An,bn,v]=solvelu(A,b)
% Esta funcion realiza la trangularizacion
% de una matriz A y el termino independiente b
% para la resolucion de un sistema Ax=b.
%
% Entradas:
%      A matriz del sistema n x n
%      b termino independiente
% Salidas:
%      An matriz triangularizada
%      bn termino independiente
%      v  vector de punteros para la pivotacion
%         parcial
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,n]=size(A);
if (m~=n)
   error('La matriz A no es cuadrada');
end
An=A;
bn=b;
v=1:n;
for k=1:n-1
   M=k;
   for i=k+1:n
      if (abs(An(v(i),k))>abs(An(v(M),k)))
 M=i;
      end
```

```
   end
   if (k~=M)
      it=v(k);
       v(k)=v(M);
       v(M)=it;
    end
    if (abs(An(v(k),k))<=1.e-8)
    error('matriz singular');
    end
    for i=k+1:n
      alfa=An(v(i),k)/An(v(k),k);
       for j=k+1:n
  An(v(i),j)=An(v(i),j)-alfa*An(v(k),j);
       end
 bn(v(i))=bn(v(i))-alfa*bn(v(k));
     end
   end
```

y

```
function x=trisuper(An,bn,v)
%   x=trisuper(An,bn,v)
% Esta funcion resulelve
% un sistema triangular
% por el metodo de sustitucion
% regresiva. Se usa despues de la
% funcion solvelu
% Entradas:
%   An matriz triangular superior como sale de solvelu
%   bn termino independiente como sale de solvelu
%   v vector de puneteros de la pivotacion parcial
% Salida:
%   x solucion del sistema
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,n]=size(An);
if (m~=n)
   error('La matriz A no es cuadrada');
end
for i=n:-1:1
   x(i)=bn(v(i));
   for j=n:-1:i+1
      x(i)=x(i)-An(v(i),j)*x(j);
   end
   x(i)=x(i)/An(v(i),i);
end
```

Dada una matriz $S$, la función implementada en Matlab para calcular su descomposición $LU$ es la función `lu( )`. De este modo, la instrucción

```
[L,U,P]=lu(S)
```

devuelve tres matrices `L`, `U` y `P` de forma que se cumple `P*S = L*U`.

El método utilizado para calcular estas matrices es el método de Gauss con pivotación parcial. La matriz de permutaciones, `P`, se calcula teniendo en cuenta sólo el proceso de la pivotación parcial y no se preocupa de optimizar la dispersión de los factores `L` y `U`.

Si se utiliza la sintaxis[1]

```
[L,U,P,Q]=lu(S)
```

se obtienen una matriz triangular inferior, `L`, una matriz triangular superior, `U`, una matriz de permutación, `P`, y una matriz de reordenamiento de columnas, `Q`, de forma que se cumple `P*S*Q=L*U`. La matrix `P` se elige mediante un método de pivotación parcial que utiliza un cierto umbral y además se determinan `P` y `Q` para controlar el relleno de los factores `L` y `U` en lo posible.

Se puede aumentar el control sobre la pivotación parcial utilizando la sintaxis

```
[L,U,P,Q]=lu(S,thresh)
```

de este modo, se fija el umbral utilizado en la pivotación parcial mediante el valor `thresh`, que es un número del intervalo $[0, 1]$. Así, se producirá el intercambio de filas cuando el módulo del elemento de la diagonal es menor que `thresh` veces el módulo de cualquiera de los elementos de la columna por debajo de la diagonal.

## 2.3. Permutación y reordenamiento

La permutación y reordenamiento de las filas y las columnas de una matriz `S`, puede hacerse de dos modos:

- Una matriz de permutaciones, `P`, actúa sobre las filas de una matriz `S` como `P*S` y sobre las columnas como `S*P'`.

[1] Hay que tener en cuenta que esta posibilidad del la función `lu ( )` se tiene en la versión 6.5 o superiores de Matlab

- Un vector de permutaciones, `p`, que contiene una permutación de los índices $1, \ldots, n$, actúa sobre las filas de `S` como `S(p,:)`, y sobre las columnas como `S(:,p)`.

Veamos algunos ejemplos. Si escribimos las instrucciones

```
p=[1 3 4 2]
I=eye(4,4)
P=I(p,:)
```

obtenemos la matriz de permutación

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} .$$

Si introducimos las instrucciones

```
S=diag(-2*ones(4,1))+diag(ones(3,1),-1)+diag(ones(3,1),1)
```

obtenemos la matriz

$$S = \begin{pmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{pmatrix} .$$

Para aplicar las permutaciones de filas definidas por el vector `p` o la matriz `P`, se introducen las instrucciones `S(p,:)` o, equivalentemente, `P*S`, obteniendo la matriz

$$PS = \begin{pmatrix} -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \\ 1 & -2 & 1 & 0 \end{pmatrix} .$$

También se pueden permutar las columnas de `S` mediante las instrucciones `S(:,p)` o `S*P'`, obteniendo

$$SP^T = \begin{pmatrix} -2 & 0 & 0 & 1 \\ 1 & 1 & 0 & -2 \\ 0 & -2 & 1 & 1 \\ 0 & 1 & -2 & 0 \end{pmatrix} .$$

En resumen, dado un vector `p` de reordenamiento se puede obtener la matriz `P` de permutación de filas o la matriz `P'` de reordenamiento de columnas mediante las instrucciones

```
I=eye(n)
P=I(p,:)
P'=I(:,p)
```

y dadas la matrices `P` y `P'` se obtiene el vector de reordenamiento de la forma

```
p=(1:n)*P'
p=(P*(1:n)')'
```

Como `P` es una matriz de permutaciones, su inversa es `R=P'`. Con el vector de reordenamiento se calcula la inversa haciendo `r(p)=1:n`. Por ejemplo, si hacemos

```
p=[1 3 4 2]
r(p)=1:4
```

obtenemos

```
r =
     1     4     2     3
```

Un modo alternativo de definir una función que devuelva una matriz elemental que intercambie la fila $i$ y la fila $j$ de una matriz de orden $n$ es el siguiente:

```
function p=pij(n,i,j)
p=eye(n);
% partimos de la identidad de orden n.
p(i,i)=0;% modificamos los elementos necesarios.
p(j,j)=0;
p(i,j)=1; p(j,i)=1;
```

De igual forma, se pueden definir funciones que multipliquen una fila por un número o le sumen a una fila una combinación lineal de otras filas.

Este tipo de matrices elementales se pueden utilizar, por ejemplo, para tranformar imágenes. Supongamos que se tiene una matriz `foto` (una matriz de enteros $300 \times 286$). Haciendo:

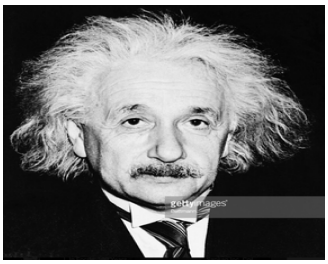


Figura 2.1: Foto de Einstein.

```
load('foto.dat')
colormap(gray(256));
image(foto);
axis off
```

obtenemos la imagen de la Figura 2.1

Mediante las siguientes instrucciones:

```
size(foto);
foto1=foto;
%
for n=1:150
P=pij(300,n,301-n);
foto1=P*foto1;
end
%
figure
colormap(gray(256));
image(foto1)
axis off
```

conseguimos invertir la foto, como muestra la Figura 2.2,

### 2.3.1. Reordenamientos

Hay distintos reordenamientos que permiten mejorar el relleno producido al calcular la descomposición LU de una matriz.

Uno de estos reordenamientos es el obtenido a partir del algoritmo de Cuthill-McKee inverso, que transforma la matriz inicial en una matriz que

Figura 2.2: Foto de Einstein invertida.

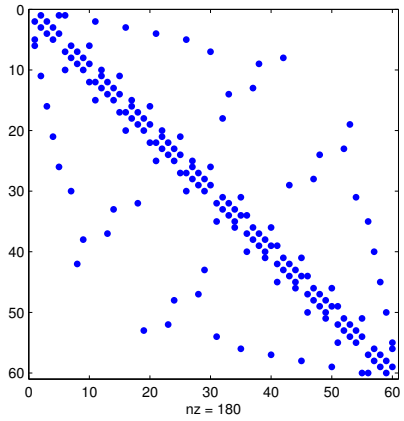



Figura 2.3: Patrón de elementos no nulos de la matriz `bucky`.

tiene sus elementos cerca de su diagonal principal. Este algoritmo está implementado en la función `symrcm( )`. Podemos pues ver su efecto sobre la matriz `bucky`. Si hacemos

```
A=bucky;
spy(A)
```

obtenemos el patrón mostrado en la Figura 2.3.

Si reordenamos la matriz y mostramos sus elementos no nulos

```
p=symrcm(A);
spy(A(p,p))
```

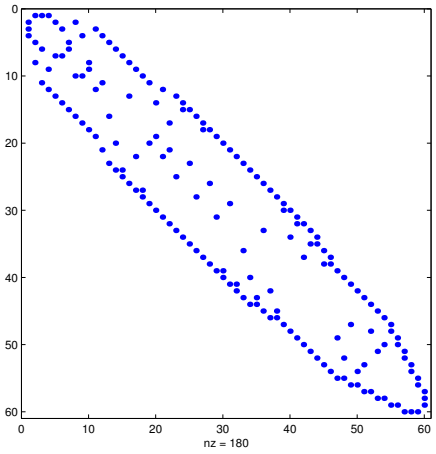



Figura 2.4: Patrón de elementos no nulos de la matriz `bucky` reordenada.

obtenemos el patrón de la Figura 2.4.

Este tipo de estructura en banda suele ser mejor para evitar el relleno en los factores $L$ y $U$ de la matriz.

Otros reordenamientos están implementados, por ejemplo, en las funciones `colamd()` y `symamd()`.

## 2.4. Descomposición QR

La función [Q,R] = qr(A) de Matlab realiza la factorización de la matriz $A$ como producto de una matriz ortogonal $Q$ por otra triangular superior $R$. Esta factorización es útil tanto para matrices cuadradas como rectangulares. Por ejemplo, si introducimos la matriz

```
A =  [ 1     2     3
       4     5     6
       7     8     9
      10    11    12 ]
```

que es una matriz de rango 2, y hacemos

```
[Q,R] = qr(A)
```

obtenemos el resultado

```
Q =

    -0.0776    -0.8331     0.5444     0.0605
    -0.3105    -0.4512    -0.7709     0.3251
    -0.5433    -0.0694    -0.0913    -0.8317
    -0.7762     0.3124     0.3178     0.4461

R =

   -12.8841   -14.5916   -16.2992
          0    -1.0413    -2.0826
          0          0     0.0000
          0          0          0
```

Si se utiliza la llamada

```
[Q,R,P] = qr(A)
```

se obtiene una matriz triangular `R`, una matriz unitaria `Q`, y una matriz de permutación `P`, tal que `A*P = Q*R`. La matriz de permutación es tal que las columnas de `A` se permutan de forma que los elementos `abs(diag(R))` están en orden decreciente.

Para obtener la descomposición QR mímima se hace

```
[Q1,R1] = qr(A,0)
```

obteniendo

```
Q1 =

-0.0776   -0.8331    0.5336
-0.3105   -0.4512   -0.8036
-0.5433   -0.0694    0.0065
-0.7762    0.3124    0.2636

R1 =

-12.8841  -14.5916  -16.2992
      0   -1.0413    -2.0826
      0         0    -0.0000
```

La siguiente función calcula la factorización QR de una matriz que no tiene rango completo y nos devuelve su rango y la permutación de columnas necesario para tener los vectores linealmente independientes.

```
function [Q R p rank] = qrrd(A,tol)
    % Calcula la factorizacion QR con permutacion de columnas
    [Q R p] = qr(A,0);

    % Busca en la diagonal de R los elementos menores
    % que una cierta tolerancia

    rrank = find(abs(diag(R))<tol);
    rank = rrank(1)-1;

    % Si se tiene rango cero da un error
    if rrank==0
        error('␣Factorizacion␣QR␣de␣una␣matriz␣de␣rango␣0')
    end

    % Reduce   Q y R
    Q = Q(:,1:rank);
    R = R(1:rank,1:rank);
```

Una función sencilla que implementa el método de Householder es la siguiente:

```
function [U,R] = householder(A)
[m, n] = size(A);
R = A;
for k = 1:n,
x = R(k:m,k);
e = zeros(length(x),1); e(1) = 1;
u = sign(x(1))*norm(x)*e + x;
u = u./norm(u);
R(k:m, k:n) = R(k:m, k:n) -2*u*u'*R(k:m, k:n);
U(k:m,k) = u;
end
```

## 2.5. Regresión lineal múltiple

Se parte de un modelo lineal para un conjunto de $N$ observaciones de una variable $y$ que depende linealmente de otras $p-1$ variables, $x_1, \ldots, x_{p-1}$. Asumimos que el modelo es de la forma

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_{p-1} x_{ip-1} + \varepsilon_i,$$

con $i = 1, \ldots, N$. Se pretende estimar el valor de las constantes del modelo $\beta_0, \ldots, \beta_{p-1}$. Para ello, construimos una matriz $X$ cuya primera columna es una columna de unos y el resto de columnas corresponden a las distintas observaciones de las variables $x_1, \ldots, x_{p-1}$, con lo que el modelo lineal queda

$$Y = X\beta + \varepsilon,$$

o sea,

$$\begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p-1} \\ \vdots & & & \vdots \\ 1 & x_{N1} & \cdots & x_{Np-1} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_{p-1} \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}$$

Consideremos el modelo lineal reescrito como

$$\epsilon = Y - X\beta.$$

Se eligen los $\beta$-s tales que minimicen

$$\begin{aligned} \|\epsilon\|_2^2 &= \sum_{i=1}^{N} \varepsilon_i^2 = \epsilon^T \epsilon \\ &= (Y - X\beta)^T (Y - X\beta) = Y^T Y - 2Y^T X\beta + \beta^T X^T X\beta \end{aligned}$$

Si $X$ es una matriz de rango completo, la solución de este problema viene dada por el sistema

$$X^T X\beta = X^T y,$$

que constituyen las *ecuaciones normales* del problema de regresión.

Otra posibilidad para abordar el problema del cálculo de las $\beta$-s consiste en utilizar la descomposición $QR$ de la matriz $X$,

$$X = QR,$$

y escribir

$$\begin{aligned} \epsilon^T \epsilon &= Y^T Y - 2Y^T QR\beta + \beta^T R^T Q^T QR\beta \\ &= Y^T Y - Y^T QQ^T Y + \left(Y^T QQ^T Y - 2Y^T QR\beta + \beta^T R^T Q^T QR\beta\right) \\ &= Y^T \left(I - QQ^T\right) Y + \left(Q^T Y - R\beta\right)^T \left(Q^T Y - R\beta\right). \end{aligned}$$

El error se minimiza si

$$R\beta = Q^T Y.$$

Veamos el siguiente ejemplo.En un estudio sobre la poblaci´on de un parásito se hizo un recuento de parásitos en 15 localizaciones con diversas condiciones ambientales. Los datos obtenidos son los siguientes:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Temperatura | 15 | 16 | 24 | 13 | 21 | 16 | 22 | 18 | 20 | 16 | 28 | 27 | 13 | 22 | 23 |
| Humedad | 70 | 65 | 71 | 64 | 84 | 86 | 72 | 84 | 71 | 75 | 84 | 79 | 80 | 76 | 88 |
| Recuento | 156 | 157 | 177 | 145 | 197 | 184 | 172 | 187 | 157 | 169 | 200 | 193 | 167 | 170 | 192 |

Se quiere ajustar un modelo del recuento en función de la temperatura y la humedad

$$R = \beta_0 + \beta_1 T + \beta_2 H + \varepsilon,$$

mediante las instrucciones:

```
paras=[
156     70      15
157     65      16
177     71      24
145     64      13
197     84      21
184     86      16
172     72      22
187     84      18
157     71      20
169     75      16
200     84      28
193     79      27
167     80      13
170     76      22
192     88      23];

Y=paras(:,1);
X=[ones(15,1),paras(:,2:3)];
[Q,R]=qr(X,0);
bet=R\Q'*Y
```

obtenemos el modelo

$$R = 25.7115 + 1.5424T + 1.5818H + \varepsilon.$$

## 2.6. Proceso de Markov

Un proceso de Markov puede modelarse como una cadena de eventos que sólo dependen del estado anterior y cuya evolución viene determinada por un conjunto determinado de transiciones.

$$w_{k+1} = Aw_k = A^k w_0.$$

Algunos de estos procesos tienen un estado estacionario

$$Aw = w$$

donde $w$ es un autovector de $A$ con autovalor 1.

Consideremos el siguiente ejemplo: Si se derrama una gota de tinta en un recipiente con agua, la tinta tiende a difundirse (disolverse) homogéneamente por todo el volumen. Este proceso puede modelarse con una ecuación en derivadas parciales, la ecuación de difusión, que es la misma que la ecuación del calor.

Si se considera el problema en una dimensión (piénsese en la concentración de tinta en la superficie, proyectada en el eje OX, por ejemplo), es posible construir un modelo discreto bastante simple basado en las cadenas de Markov. Una manera de hacerlo es así:

- Se discretiza la anchura del recipiente en, digamos 300 elementos.
- Se postula que la concentración de tinta cumple la ley siguiente: una partícula tiene probabilidad 0.8 de quedarse en su lugar, 0.1 de ir hacia la derecha y 0.1 de ir hacia la izquierda.
- En las dos casillas de los bordes, la probabilidad de quedarse es 0.8 y la de ir a la contigua es 0.2.

Con estos tres principios, se tiene un proceso de Markov.

La matriz de transición es sencilla de construir: 0.8 en la diagonal principal y 0.1 en las dos diagonales contiguas, excepto en los lugares $A(2,1)$ y $A(299,300)$ que es 0.2, esto se implementa como:

```
A = 0.8*eye(300);
A = A + 0.1*diag(ones(299,1),1) + ...
    0.1*diag(ones(299,1),-1);
A(2,1)=0.2;
A(299,300)=0.2;
```

La evolución del sistema (que es muy lenta, pues la ley que hemos impuesto hace que la tinta tarde mucho en disolverse) puede observarse como se indica a continuación. Partimos de una mancha de tinta que llena solo la primera casilla:

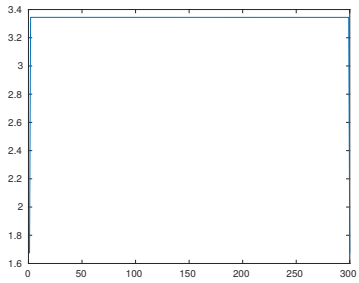



Figura 2.5: Estado estacionario de la distribución de tinta.

```
v = zeros(300,1);
v(1) = 1000;
for k=1:100:10000
plot(A^k*v);
axis([0 300 0 80]);
pause(0.1);
end
```

se observa cómo la tinta se distribuye poco a poco homogéneamente. Al cabo de un millón de pasos, la cosa queda más clara: Haciendo `bar(A^1000000*v)`, obtenemos la Figura 2.5.

Puede comenzarse con una distribución diferente, por ejemplo, con varias manchas de tinta en diversos lugares:

```
w = zeros(300,1);
w(3)=1000;
w(100)=5000;
w(200)=7000;
w(295)=1000;

for k=1:1000:100000
  plot(A^k*w);
  axis([0 300 0 80]);
  pause(0.1);
end
```

obteniendo el mismo estado estacionario para la distribución de tinta. Esta manera de entender la difusión es similar a la que utilizó Einstein en su trabajo de 1905 para describir el movimiento Browniano.

El estado estacionario se puede obtener resolviendo el problema de auto-

valores asociado a la matriz $A$. Obtener el autovector resolviendo el sistema

$$Ax = x \Longrightarrow (A - I)x = 0,$$

nos conduce a problemas. ¡Compruébalo!.

## 2.7. Datos faltantes

Se dispone de una serie de datos donde faltan algunas observaciones y se trata de obtener una estimación de los mismos.

Para formular el problema como un problema de mínimos cuadrados introducimos alguna notación. Sea $x$ una señal de longitud N. Supongamos que se conocen $K$ muestras de $x$ donde $K < N$. La señal conocida de $K$ puntos, $y$, puede escribirse como

$$y = Sx$$

donde $S$ es una matriz de selección (o muestreo) de tamaño $K \times N$.

Por ejemplo, si sólo el primer segundo y último elemento de una señal de 5 puntos x se observan,

$$x = (-1.4, -1.2, NaN, NaN, -9.8)$$

entonces la matriz $S$ viene dada por

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

La matriz $S$ es la matriz identidad a la que se han eliminado las filas correspondientes a las muestras que faltan. Obsérvese que $Sx$ elimina las muestras que no se conocen de la señal $x$,

$$Sx = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{pmatrix} = \begin{pmatrix} x(0) \\ x(1) \\ x(4) \end{pmatrix} = y$$

El vector $y$ está formado por las muestras conocidas de $x$. Por lo tanto, el vector $y$ es más corto que $x$, $(K < N)$.

El problema puede plantearse como sigue: Dada la señal, $y$, y la matriz, $S$ encontrar $x$ tal que

$$y = Sx.$$

Por supuesto, hay infinitas soluciones. A continuación, se muestra cómo obtener una solución suave por mínimos cuadrados.

Observad que $S^T y$ tiene el efecto de poner o cero las componentes que faltan

$$S^T y = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y(0) \\ y(1) \\ y(2) \end{pmatrix} = \begin{pmatrix} y(0) \\ y(1) \\ 0 \\ 0 \\ y(2) \end{pmatrix}$$

Definamos $S_c$ como la matriz complementaria de $S$. La matrix $S_c$ está formada por las filas de la matriz identidad que no están en $S$. Siguiendo con el ejemplo de 5 puntos,

$$Sc = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Una estimación $\hat{x}$ se puede representar por

$$\hat{x} = S^T y + S_c^T v \ ,$$

donde $y$ son los datos disponibles y $v$ las muestras que se han de determinar. Por ejemplo

$$S^T y + S_c^T v = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y(0) \\ y(1) \\ y(2) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v(0) \\ v(1) \end{pmatrix} = \begin{pmatrix} y(0) \\ y(1) \\ v(0) \\ v(1) \\ y(2) \end{pmatrix}$$

El problema consiste en estimar el vector $v$, que es de longitud $N - K$.

Asumamos que la señal original, $x$, es suave. Es razonable buscar un $v$ que optimice la suavidad de $\hat{x}$, o sea, que minimice la energía de la derivada segunda de $\hat{x}$. Así, $v$ se puede obtener minimizando $\|D\hat{x}\|_2^2$ donde $D$ es una aproximación del operador segunda derivada

$$D = \begin{pmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & & \dots \\ & & & 1 & -2 & 1 \end{pmatrix}$$

Hemos de obtener $v$ resolviendo el problema

$$\min_v \left\| D\left(S^T y + S_c^T v\right)\right\|_2^2 = \min_v \left\| DS^T y + DS_c^T v\right\|_2^2$$

cuya solución viene dada por

$$v = -\left(S_c D^T D S_c^T\right)^{-1} S_c D^T D S^T y$$

Una vez se ha obtenido $v$ basta insertarlo en el vector inicicial.

Veamos un ejemplo. Cargamos el fichero `datos_faltantes.dat`

```
%%
clear
close all
load datos_faltantes.dat;
y=datos_faltantes;
N=length(y)
n=1:N;
y(1:10)
```

obtenemos

```
ans =

-0.0144
NaN
-0.0126
NaN
-0.0108
-0.0099
NaN
NaN
-0.0065
NaN
```

observamos que hay valores del vector que no están disponibles, etiquetados con `NaN`.

Para dibujar los datos

```
%%
figure(1)
clf
plot(n,y,'k','LineWidth',3)
title('Datos')
```

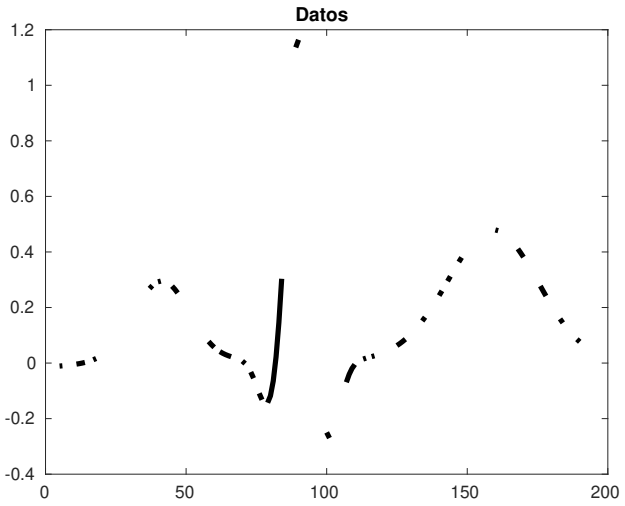



Figura 2.6: Serie de datos con datos faltantes.

obteniendo la Figura 2.6

Definimos las matrices $D$, $S$ y $S_c$

```
%% Matriz D
e = ones(N, 1);
D = spdiags([e -2*e e], -1:1, N, N);

%% matrices S y Sc

k = isfinite(y);      % k :  vector logico

S = speye(N);
S(~k, :) = [];        % S : matriz muestreo

Sc = speye(N);        % Sc : matriz complementaria de S
Sc(k, :) = [];

L = sum(~k)           % L : numero de valores faltantes

%%
% Estimacion de los valores faltantes .

v = -(Sc * (D' * D) * Sc') \ ( Sc * D' * D * S' * y(k));

%% LLenado de los valores

x = zeros(N,1);
```

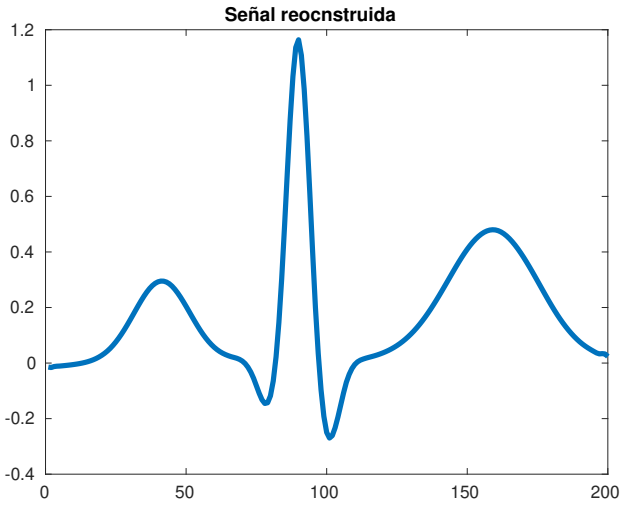



Figura 2.7: Serie de datos reconstruida.

```
x(k) = y(k);
x(~k) = v;
%%

figure(1)
clf
plot(n,x,'LineWidth',3 )
title('Senal reconstruida')
```

obteniendo la figura 2.7

## 2.8. Ejercicios

1. Modifica la función `solvelu( )` para que calcule el determinante de una matriz. Modifica la función `trisuper( )` para que resuelva $n$ sistemas de ecuaciones simultáneos con la misma matriz de coeficientes. Utiliza la función modificada para obtener la inversa de una matriz, resolviendo $n$ sistemas de ecuaciones.

2. La matriz de Frank de orden $n$ se define por

$$a_{ij} = \begin{cases} 0 & \text{si } j < i-2 \\ n+1-i & \text{si } j = i-1 \\ n+1-j & \text{si } j \geq i \end{cases}$$

y en Matlab se puede obtener mediante la instrucción

```
A= gallery('frank',n)}
```

Se sabe que el determinate de la matriz de Frank vale 1. Obtén el valor del determinante de la matriz de Frank usando la descomposición LU de la matriz utilizando pivotación y sin utilizarla cuando $n = 10, 20, 100$.

3. Recuerda que una matriz simétrica y definida positiva admite factorización de Cholesky, esto es, si $A$ es simétrica y definida positiva, existe una matriz triangular inferior $L$ tal que $A = L^T L$. la función de Matlab `chol( )` realiza la factorización de Cholesky de una matriz. Esto es, la instrucción

```
R=chol(A)
```

devuelve una matriz triangular superior, $R$, tal que $R^T R = A$. Se pide modificar la función `trisuper( )` para construir una nueva función que permita resolver un sistema de ecuaciones asociado a una matriz triangular inferior. Usar esta nueva función, la función `trisuper( )` y la función `chol( )`, para resolver el siguiente sistema

$$\begin{pmatrix} 4 & -1 & 1 \\ -1 & 4.25 & 2.75 \\ 1 & 2.75 & 3.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 15.75 \\ 17 \end{pmatrix}$$

Utiliza esta metodología para resolver un sistema asociado a una matriz simétrica y definida positiva de la colección Matrix Market

```
http://math.nist.gov/MatrixMarket/
```

4. Como ya vimos en una práctica anterior, los vectores almacenados en los ficheros `V.dat`, `I.dat` y `J.dat` definen una matriz dispersa en formato coordenando. Construye la matriz dispersa $A$ de Matlab a partir de estos vectores y calcula su descomposición LU. Calcula los elementos no nulos de la matriz L obtenida y compáralos con los elementos no

nulos de la matriz inicial. Las funciones: symamd, colamd, symmmd, colmmd, symrcm, dmperm, implementan distintos reordenamientos para las matrices. Compara el funcionamiento de los reordenamientos sobre la matriz $A$. Para ello, bastará obtener el número de elementos no nulos de las matrices triangular inferior y triangular superior de la descomposición LU de la matriz reordenada. Resuelve el sistema

$$Ax = b$$

donde $b$ es un vector de números aleatorios del intervalo $[0, 1]$. Utiliza para ello el mejor de los reordenamientos.

5. Recordad que si se usa un mallado uniforme de la forma $x_{i+1} = x_i + \Delta x,\ i = 0, 1, \ldots, n_x + 1,\ y_{j+1} = y_j + \Delta y,\ j = 0, 1, \ldots, n_y + 1$, una aproximación del laplaciano es

$$\nabla^2 (x_i, y_j) \approx \frac{1}{\Delta x^2} (u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + \frac{1}{\Delta y^2} (u_{i,j-1} - 2u_{i,j} + u_{i,j+1}) \ .$$

Se tiene el problema

$$\nabla^2 u = 0 \ , \quad (x, y) \in R \ ,$$

siendo $R = \{(x, y) :\ 0 \leq x \leq 4,\ 0 \leq y \leq 4\}$ con las condiciones de contorno

$$u(x, 0) = 20,\ u(x, 4) = 180, \quad 0 < x < 4,$$

y

$$u(0, y) = 8,\ u(4, y) = 0, \quad 0 < y < 4.$$

Resuelve el problema utilizando $n_x = n_y = 50, 100, 150, 200$ nodos. Haz una gráfica del tiempo que tarda en resolover el problema el matlab en función del número de nodos si se usa el almacenamiento denso para la matriz del problema y si se usa el almacenamiento disperso.

6. Dada la matriz de orden $n$ dada por

$$a_{ij} = \begin{cases} i(n - j + 1) & \text{si } i \leq j \\ a_{ji} & \text{si } i > j. \end{cases}$$

y la columna de términos independientes $b = (1, 2, \ldots, n)$. Resuelve el sistema

$$Ax = b \ ,$$

tomando $n = 10, 10^2, 10^3, 10^4$ y utilizando la descomposición $LU$ de $A$ y la descomposición $QR$ de $A$. Haz una gráfica logarítmica comparando los tiempos necesarios para resolver los sistemas.

7. Los datos contenidos en las columnas $(x, y)$ de la matriz `Mat1.dat` se sabe que aproximadamente se distribuyen según una ecuación de la forma

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 + a_4 x_i^4 + a_5 x_i^5 \ . \tag{2.1}$$

se pueden obtener los coeficientes $a_0, \ldots a_5$, resolviendo el sistema sobredeterminado

$$Ba = y \ ,$$

definido por le ecuación (2.1) utilizando la factorización $QR$ de la matriz $B$ o bien resolviendo las ecuaciones normales

$$B^T B a = B^T y \ .$$

Dibuja la nube de puntos y el modelo obtenido en el ajuste. Compara los resultados que proporcionan ambos métodos.

8. La instrucción `A=gallery('smoke',1000)` genera una matriz compleja $1000 \times 1000$. Dado el sistema

$$Ax = b$$

donde $b$ es el vector de $\mathbb{R}^{1000\times 1}$ cuyos elementos son todos $b_i = 1 + i$, escribe este sistema como un sistema equivalente donde las matrices involucradas sean reales. Compara el tiempo necesario para resolver el sistema si se utiliza aritmética real o compleja. Realiza los cálculos almacenado $A$ en formato denso y en formato disperso y comenta los resultados obtenidos.

9. La instrucción `A =gallery('wathen',nx,ny)` genera una matriz dispersa, $N \times N$, donde $N = 3 * nx * ny + 2 * nx + 2 * ny + 1$. La matriz es simétrica y definida positiva. Genera matrices utilizando $nx = ny = 15, 30, 60, 150$ y 200. Considera la matriz original y la versión reordenada que se obtiene utilizando las funciones `symrcm` y `symamd`. Mide el tiempo necesario en los tres casos para obtener la factorización de Cholesky de estas matrices. Determina así mismo el tiempo necesario para resolver un sistema

$$Ax = b$$

donde $b$ es el vector de unos, usando la factorización de Cholesky obtenida. Repite estos cálculos para una matriz simétrica y definida positiva de la colección de matrices de la Universidad de Florida (`http://www.cise.ufl.edu/research/sparse/matrices/`)

10. La matriz $A$ que está en el formato coordenado en el fichero `standford.dat` es un trozo de una matriz de Google, que es una matriz estocástica por filas (los elementos de cada columna suman 1). La matriz $A$ es rectangular y su rango no es completo. Calcula su rango y qué vectores son linealmente independientes usando la función `qrrd()` de las notas.

    Utiliza los algoritmos de Gram-Schmidt y el método de Gram-Schmidt modificado para calcular la matriz Q de los vectores columna linealmente independientes de $A$. Comprobar la calidad de la descomposición viendo lo que dista en ambos casos la matriz $Q^TQ$ de la matriz identidad.

11. Dada una cierta señal $y(nT), n = 0, \ldots N-1$ que presenta una cierta cantidad de ruido, podemos obtener una señal filtrada como la solución de un problema de la forma

$$\min_x \left( \|x - y\|_2^2 + \lambda \|Dx\|_2^2 \right)$$

    donde $D$ es una matriz tridiagonal de la forma

$$D = \begin{pmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & & \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & 1 & -2 & 1 \end{pmatrix}$$

    que es una aproximación de la derivada segunda y $\lambda$ es una constante. La solución del problema de minimización es

$$\left( I + \lambda D^T D \right) x = y.$$

    A partir del fichero `senalruido.dat` que continene una señal con ruido, obtener distintas señales filtradas variando el valor del parámetro $\lambda$. Compáralas en un gráfico.

12. Se definen los polinomios de Legendre ortonormales en $[-\frac{1}{2}, \frac{1}{2}]$, como los polinomios $P_n(t)$, que satisfacen

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} P_n(t) P_m(t)\, dt = \delta_{n,m},$$

    donde $\delta_{n,m}$ es la delta de Kronecker. Estos polinomios satisfacen:

$$P_0(t) = 1, \quad P_1(t) = 2\sqrt{3}t,$$
$$P_{n+1}(t) = \frac{2}{n+1}\sqrt{2n+3}\sqrt{2n+1}P_n(t) - \frac{n}{n+1}\frac{\sqrt{2n+3}}{\sqrt{2n-1}}P_{n-1}(t).$$

Los polinomios discretos de Legendre se pueden obtener a partir de la descomposición $QR$ de la matriz de vandermonde

$$A = \begin{pmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^n \\ 1 & t_2 & t_2^2 & \cdots & t_2^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^n \end{pmatrix},$$

cuyas columnas son los monomios $t^0, t^1, \ldots, t^n$ evaluados en los puntos $t_1, t_2, \ldots, t_m$, que definen una partición del intervalo $[-\frac{1}{2}, \frac{1}{2}]$. Comparar, mediante un gráfico, los 10 primeros polinomios de Legendre ortonormales continuos y discretos cuando $m = 10, 100, 500$.

13. El fichero `consumo_coches.csv` contiene datos del consumo de distintos vehículos (mpg milles per gallon) en función de otras variables: cylinders, displacement, horsepower, weight, year, origin. Se pretende ajustar, mediante el método de mínimos cuadrados, un modelo lineal para el consumo de la forma

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p.$$

Recordad que, en forma matricial, este modelo se puede expresar como

$$Y = X\beta,$$

y la solución para los coeficientes se escribe como

$$\beta = R^{-1}Q^T Y,$$

siendo $X = QR$ la descomposición $QR$ de $X$. Estudia el error asociado a este modelo.

14. Dados los vectores,

$$\vec{v}_1 = (1, 1, 1, -1), \quad \vec{v}_2 = (2, -1, -1, 1),$$
$$\vec{v}_3 = (0, 3, 3, -3), \quad \vec{v}_4 = (-1, 2, 2, 1).$$

Obtén un sistema de vectores ortonormal utilizando el algoritmo de Gram-Schmidt y utilizanod el algoritmo de Gram-Schmidt modificado. ¿Cuál es la dimensión del espacio generado por los vectores?.

15. Considera el sistema

$$(A + \varepsilon B)\, x = b,$$

donde $A$ y $B$ son matrices regulares $n \times n$, $b$ es un vector $n$ dimensional y $\varepsilon$ un número pequeño. La solución del sistema se puede escribir como

$$x(\varepsilon) = (A + \varepsilon B)^{-1} b. \tag{2.2}$$

Si $\varepsilon$ es pequeño, desarrollando en serie,

$$x(\varepsilon) = x_0 + \varepsilon x_1 + \varepsilon^2 x_2 + \varepsilon^3 x_3 + \cdots ,$$

y, por tatnto, se ha de cumplir

$$(A + \varepsilon B)\left(x_0 + \varepsilon x_1 + \varepsilon^2 x_2 + \varepsilon^3 x_3 + \cdots\right) = b,$$

con lo que se ha de cumplir

$$Ax_0 = b, \quad Ax_1 + Bx_0 = 0, \quad Ax_2 + Bx_1 = 0, \quad \cdots . \tag{2.3}$$

Considera la matrix $A$ obtenida con la instrucción:

```
A=gallery('dorr',500,0.05)
```

y la matriz $B$ obtenida con `B=randn(500)` y estudia el error que se comete al aproximar la solución del problema (2.2) resolviendo 5 sistemas de la forma (2.3) cuando $\varepsilon = 0.1, 0.01$ y $0.001$. Notar que para resolver de modo eficiente los sistemas de (2.3) es conveniente obtener la descomposición $LU$ de la matriz $A$.

16. La ecuación de la difusión dependiente del tiempo

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} ,$$

se puede utilizar como un filtro para eliminar el ruido de una imagen. Se pide utilizar esta ecuación discretizada mediante un método implícito de primer orden para mejorar la imagen `cerebro_ruido.jpg`, aplicando distintos pasos $\Delta t = 0.01, 0.1, 1$ y distintos números de pasos de tiempo. Hay que tener en cuenta que para las imágenes se supone que $\Delta x = \Delta y = 1$ y que las condiciones de contorno son $\vec{n}\vec{\nabla} = 0$ en la frontera de la imagen, donde $\vec{n}$ es el vector unitario normal a la superficie que define a la frontera.

Para leer una imagen hay que usar la función `imread()` de Matlab y para ver el resultado `imshow()`. Además, para hacer cálculos hay que hacer la conversion a de tipos de uint8 a double.

Ejemplo de uso de imágenes en Matlab:

```
Imagenr = imread('cerebro_ruido.jpg');
I2 = double(Imagenr);
I3=255-I2
I4 = uint8(I3);
imshow(I4)
```