Programa de Análisis Numérico II

Tema 1. Métodos directos para la resolución de sistemas de ecuaciones lineales.

- Método de Gauss.
  gauss1.m, eliminación Gaussiana sin pivoteo
  gauss2.m, eliminación Gaussiana con pivoteo básico
  gauss3.m, eliminación Gaussiana con pivoteo parcial de columna
  gauss4.m, eliminación Gaussiana con pivoteo escalado de columna
  gauss5.m, eliminación Gaussiana con pivoteo total
- Método de Doolittle.
  doolittle.m, factorización de Doolittle
  doolittle2.m, factorización de Doolitle con pivoteo de columna.
- Método de Crout.
  crout.m, crout2.m
- Método de Cholesky.
  cholesky.m, cholesky2.m

Tema 2. Métodos iterativos de resolución de sistemas de ecuaciones lineales

- Introducción a los métodos iterativos en general.
- Métodos de Jacobi, Gauss-Seidel y relajación.
  jacobi.m, gs.m y sor.m
- Estimación del error y refinamiento iterativo.
  refit.m y refit2.m

Tema 3. Aproximación de los valores característicos.

- Álgebra lineal y valores característicos.
- Método de la potencia.
  potencias.m, potencias_aitken.m, potencias_msimetrica.m,
  potencias_inversas.m, potencias_inversas_aitken.m
- Método de Householder.
  householder.m
- Algoritmo QR.
  qr2.m

Tema 4. Problemas de contorno para ecuaciones diferenciales.

- El método de diferencias finitas.
  Diffin.m
- Métodos variacionales.
  RRL.m, RRTC.m

Tema 5. Ecuaciones en derivadas parciales parabólicas.

- Ecuaciones en diferencias: consistencia, convergencia y estabilidad.
- Ecuación de difusión unidimensional: métodos explícito, mplícito y de Crank-Nicolson.
  forwdiff.m, backdiff.m, CrankNicolson.m
- Ecuación de onda unidimensional. Métodos explícito e implícito.
  onda.m
- Métodos en diferencias finitas.
  poisson.m

Bibliografía.

Análisis numérico. Richard L. Burden, J. Douglas Faires.
Métodos numéricos con Matlab. John H. Mathews, Kurtis D. Fink.
Análisis numérico. David Kincaid, Ward Cheney.

Métodos directos para la resolución de sistemas de ecuaciones lineales

1. Métodos de Gauss.
2. Algoritmo de Doolittle.
3. Algoritmo de Crout.
4. Algoritmo de Cholesky.

1. Metodos de Gauss.

Para aplicar el algoritmo básico de eliminación Gaussiana al sistema lineal:

$$\begin{aligned} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n &= b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n &= b_2 \\ &\ldots \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + \ldots + a_{nn} \cdot x_n &= b_n \end{aligned}$$

formamos la matriz aumentada $\tilde{A} = [A, \mathbf{b}]$, donde A denota la matriz formado por los coeficientes. Los elementos de la columna n+1 de $\tilde{A}$ son los valores de **b**, vector de los términos independientes, es decir, $a_{(i,n+1)} = b_i$ para i = 1, 2, ..., n.

Siempre y cuando $a_{ii} \neq 0$ se realiza la operación $E_j - \frac{a_{ji}}{a_{ii}} \cdot E_i \rightarrow E_j$ siguiendo un procedimiento secuencial para i = 2, 3, ..., n-1 y j = i+1, i+2, ..., n, se anularán todos los valores i = 1, 2, ..., n-1. La matriz resultante tendrá la forma siguiente:

$$\tilde{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} & b_1 \\ 0 & a_{22} & \ldots & a_{2n} & b_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & a_{nn} & b_n \end{bmatrix}$$

La matriz anterior representa un sistema lineal triangular con el mismo conjunto de soluciones que el sistema inicial, por tanto, se puede realizar la sustitución hacia atrás comenzando por la n-ésima ecuación para x_n :

$$x_n = \frac{a_{n,n+1} - a_{n-1,n} \cdot x_n}{a_{n-1,n-1}}$$

y continuando con este proceso llegamo a:

$$x_i = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij} \cdot x_j}{a_{ii}} \quad para\ i = n-1, n-2, \ldots, 1$$

Codificación en Matlab

```
% gauss.m - eliminación gaussiana.
% No tiene en cuenta que el elemento pivote sea 0.
%
% Entradas:
%   A, matriz de los coeficientes
%   b, vector de los términos independientes
%
% Salidas:
%   x, vector solución del sistema
%   men, mensaje de éxito o fracaso
%
function [x, men]=gauss1(A, b)
Aum = [A b];
[m n] = size(Aum);
for p = 1:m-1
    for fila = p+1:m
        if Aum(p, p) == 0
            x = [];
            men = 'El elemento pivote es 0, el procedimiento no puede
continuar';
            return
        end
        mult = Aum(fila, p) / Aum(p, p);
        Aum(fila, p) = 0;
        for col = p+1:n
            Aum(fila, col) = Aum(fila, col) - mult*Aum(p, col);
        end
    end
end
x = sustreg(Aum(1:m, 1:n-1), Aum(:, n));
men = 'Procedimiento completado con éxito';
```

Ejemplo 1. El sistema de ecuaciones:

$$
\begin{aligned}
E_1 &: x_1 + x_2 + 3x_4 = 4 \\
E_2 &: 2x_1 + x_2 - x_3 + x_4 = 1 \\
E_3 &: 3x_1 - x_2 - x_3 + 2x_4 = -3 \\
E_4 &: -x_1 + 2x_2 + 3x_3 - x_4 = 4
\end{aligned}
$$

tiene por solución:

```
>> A = [1 1 0 3; 2 1 -1 1; 3 -1 -1 2; -1 2 3 -1]

A =

     1     1     0     3
     2     1    -1     1
     3    -1    -1     2
    -1     2     3    -1

>> b = [4 1 -3 4]'

b =

     4
     1
    -3
     4
```

```
>> x=gauss1(A, b)

x =

    -1
     2
     0
     1
```

Ejemplo 2. El sistema lineal Ax = b dado por

$$
\begin{aligned}
&E_1: 10x_1 - x_2 + 2x_3 = 6,\\
&E_2: -x_1 + 11x_2 - x_3 + 3x_4 = 25,\\
&E_3: 2x_1 - x_2 + 10x_3 - x_4 = -11,\\
&E_4: 3x_2 - x_3 + 8x_4 = 15.
\end{aligned}
$$

tiene por solución a x = (1, 2, -1, 1)'.

```
>> A=[10 -1 2 0; -1 11 -1 3; 2 -1 10 -1; 0 3 -1 8]

A =

    10    -1     2     0
    -1    11    -1     3
     2    -1    10    -1
     0     3    -1     8

>> b=[6 25 -11 15]'

b =

     6
    25
   -11
    15

>> gauss1(A,b)

ans =

    1.0000
    2.0000
   -1.0000
    1.0000
```

El algoritmo que acabamos de desarrollar fallará si alguno de los pivotes $a_{1,1}^{(1)}, a_{2,2}^{(2)}, \ldots, a_{n-1,n-1}^{(n-1)}$ es cero, o la sustitución hacia atrás no se puede llevar a cabo porque $a_{n,n}^{(n)} = 0$. Un procedimiento mejorado debería incluir el intercambio de ecuaciones en el sistema cuando el elemento pivote es cero.

Ejemplo 3. Si se intenta resolver el sistema de ecuaciones:

$$
\begin{aligned}
&E_1: x_1 - x_2 + 2x_3 - x_4 = -8,\\
&E_2: 2x_1 - 2x_2 + 3x_3 - 3x_4 = -20,\\
&E_3: x_1 + x_2 + x_3 = -2,\\
&E_4: x_1 - x_{2} + 4x_3 + 3x_4 = 4
\end{aligned}
$$

con el método de eliminación gaussiana surge un problema debido a que surge un elemento

pivote que es 0:

```
>> A = [1 -1 2 -1; 2 -2 3 -3; 1 1 1 0; 1 -1 4 3]

A =

     1    -1     2    -1
     2    -2     3    -3
     1     1     1     0
     1    -1     4     3

>> b = [-8 -20 -2 4]'

b =

    -8
   -20
    -2
     4

>> [x,men]=gauss1(A, b)

x =

     []


men =

El elemento pivote es 0, el procedimiento no puede continuar
```

Algoritmo de eliminación gaussiana con intercambio de filas.

```
% gauss2.m - eliminación gaussiana.
% Realiza intercambio de filas para eliminar el problema que surge
% cuando un pivote es 0
% La estrategia de pivoteo que emplea es trivial.
%
% Entradas:
%   A, matriz de los coeficientes
%   b, vector de los términos independientes
% Salidas:
%   x, vector solución del sistema
%   men, mensaje de éxito o fracaso
%
function [x, men]=gauss2(A, b)
Aum = [A b];
[m n] = size(Aum);

for p = 1:m-1
    for fila = p+1:m
        Aum=pivota(Aum, p);
        if Aum == []
            x = [];
            men = 'No existe solución única';
            return
        end
        mult = Aum(fila, p) / Aum(p, p);
        Aum(fila, p) = 0;
        for col = p+1:n
            Aum(fila, col) = Aum(fila, col) - mult*Aum(p, col);
        end
    end
end
x = sustreg(Aum(1:m, 1:n-1), Aum(:, n));
men = 'Procedimiento completado con éxito';

% Si el pivote es 0,
%   intercambia la fila p-ésima con alguna fila posterior para
%   obtener un elemento no nulo.
function A=pivota(A, p)
[m n] = size(A);

c = 0;
if A(p, p) == 0
    for fila = p+1:m
        if A(fila, p) ~= 0
            tmp = A(fila, :);
            A(fila, :) = A(p, :);
            A(p, :) = tmp;
            c = 1;
            break
        end
    end
    if c == 0
        A = [];
    end
end
```

```
>> [x,men]=gauss2(A, b)

x =

    -7
     3
     2
     2


men =

Procedimiento completado con éxito
```

Ejemplo 4. El proposito de este ejemplo es mostrar qué pasa si el algoritmo de eliminación Gaussiana falla. El sistema:

$$
\begin{aligned}
E_1 &: x_1 + x_2 + x_3 + x_4 = 7,\\
E_2 &: x_1 + x_2 + 2x_4 = 8,\\
E_3 &: 2x_1 + 2x_2 + 3x_3 = 10,\\
E_4 &: -x_1 - x_2 - 2x_3 + 2x_4 = 0.
\end{aligned}
$$

tiene infinita soluciones.

```
>> A = [1 1 1 1; 1 1 0 2; 2 2 3 0; -1 -1 -2 2]

A =

     1     1     1     1
     1     1     0     2
     2     2     3     0
    -1    -1    -2     2

>> b = [7 8 10 0]'

b =

     7
     8
    10
     0

>> [x,men]=gauss2(A, b)

x =

     []


men =

No existe solución única
```

Ejemplo 5. El sistema

$$
\begin{aligned}
E_1 &: x_1 + x_2 + x_3 + x_4 = 7,\\
E_2 &: x_1 + x_2 + 2x_4 = 5,\\
E_3 &: 2x_1 + 2x_2 + 3x_3 = 10,\\
E_4 &: -x_1 - x_2 - 2x_3 + 2x_4 = 0.
\end{aligned}
$$

no tiene solución.

```
>> A = [1 1 1 1; 1 1 0 2; 2 2 3 0; -1 -1 -2 2]

A =

     1     1     1     1
     1     1     0     2
     2     2     3     0
    -1    -1    -2     2

>> b = [7 5 10 0]'

b =

     7
     5
```

```
     10
      0

>> [x,men]=gauss2(A, b)

x =

     []


men =

No existe solución única
```

Estrategias de pivoteo.

En la práctica, es deseable realizar el intercambio de las filas que contienen pivotes cuando éstas son pequeñas en relación a los elementos que se encuentran por debajo de él en la misma columna para reducir los errores de redondeo.
Las estrategias de pivoteo se llevan a cabo, en general, seleccionando un nuevo elemento como pivote $a_{p,q}^{(k)}$ intercambiando los renglones k y p, e intercambiando las columnas k y q si es necesario. La estrategia más simple, pivoteo máximo de columna o pivoteo parcial, consiste en seleccionar el elemento en la misma columna que está abajo de la diagonal y que tiene el mayor valor absoluto.

$$|a_{p,k}^{(k)}| = max_{k \le i \le n} |a_{i,k}^{(k)}|$$

y se efectúa $(E_k) \Leftrightarrow (E_p)$ sin realizar intercambio de columnas.

Ejemplo.

$$E_1 : 0.003x_1 + 59.14x_2 = 59.17,$$
$$E_2 : 5.291x_1 - 6.130x_2 = 46.78.$$

tiene por solución $(10.00, 1.00)^t$

Pivoteo escalado de columna.

Este procedimiento consiste en definir un factor de escala s_i para cada fila:

$$s_i = max_{j=1,2,\ldots,n} |a_{ij}|$$

Si $s_i = 0$ para alguna i, implica que no existe solución única y el procedimiento se detiene.
El intercambio apropiado de reglones para obtener ceros en la primera columna queda determinado escogiendo el primer entero k con:

$$\frac{|a_{ki}|}{s_k} = max_{j=1,2,\ldots,n} \frac{|a_{j1}|}{s_j}$$

y realizando $(E_1) \Leftrightarrow (E_k)$. El efecto de escalar consiste en asegurar que el elemento mayor de cada renglón tenga una magnitud relativa de uno antes de que se empiecela comparación

para el intercambio de renglones. El escalamiento se hace sólo con propósitos de comparación, así que la división entre factores de escala no produce un error de redondeo.

Ejemplo .

$$E_1 : 30.00x_1 + 591400x_2 = 591700,$$
$$E_2 : 5.291x_1 + 6.130x_2 = 46.78.$$

Pivoteo máximo o total.

El pivoteo máximo en el k-ésimo paso busca todos los elementos:

$$a_{ij}, \text{ para } i = k, k+1, \ldots, n \text{ y } j = k, k+1, \ldots, n$$

para encontrar el elemento que tiene la magnitud más grande.

Ejercicio 1. Resuelva los siguientes sistemas lineales usando eliminación Gaussiana.

a)

$$E_1 : x_1 + 2x_2 + 3x_3 = 1,$$
$$E_2 : 2x_1 + 3x_2 + 4x_3 = -1,$$
$$E_3 : 3x_1 + 4x_2 + 6x_3 = 2.$$

```
>> A = [1 2 3; 2 3 4; 3 4 6]

A =

     1     2     3
     2     3     4
     3     4     6

>> b = [1 -1 2]'

b =

     1
    -1
     2

>> x = gauss1(A, b)

x =

     0
    -7
     5

>> A*x

ans =

     1
    -1
     2
```

b)

$$E_1: 2x_1 + 4x_2 - x_3 = -5,$$
$$E_2: x_1 + x_2 - 3x_3 = -9,$$
$$E_3: 4x_1 + x_2 + 2x_3 = 9.$$

```
>> B = [2 4 -1; 1 1 -3; 4 1 2]

B =

     2     4    -1
     1     1    -3
     4     1     2

>> b = [ -5 -9 9]'

b =

    -5
    -9
     9

>> x = gauss1(B, b)

x =

     1
    -1
     3

>> B*x

ans =

    -5
    -9
     9
```

c)

$$E_1: 0.1x_1 + 0.2x_2 + 0.4x_3 = 1.1,$$
$$E_2: 4x_1 + x_2 - x_3 = 6,$$
$$E_3: 2x_1 + 5x_2 + 2x_3 = 3.$$

```
>> C = [0.1 0.2 0.4; 4 1 -1; 2 5 2]

C =

    0.1000    0.2000    0.4000
    4.0000    1.0000   -1.0000
    2.0000    5.0000    2.0000

>> b = [ 1.1 6 3]'

b =

    1.1000
    6.0000
    3.0000

>> x = gauss1(C, b)

x =

    2.6271
   -1.6102
    2.8983

>> C*x
```

```
ans =

    1.1000
    6.0000
    3.0000
```

d)

$$E_1: 0.04x_1 + 0.01x_2 - 0.01x_3 = 0.06,$$
$$E_2: 0.2x_1 + 0.5x_2 - 0.2x_3 = 0.3,$$
$$E_3: x_1 + 2x_2 + 4x_3 = 11.$$

```
>> D = [0.04 0.01 -0.01; 0.2 0.5 -0.2; 1 2 4]

D =

    0.0400    0.0100   -0.0100
    0.2000    0.5000   -0.2000
    1.0000    2.0000    4.0000

>> b = [0.06 0.3 11]'

b =

    0.0600
    0.3000
   11.0000

>> x = gauss1(D, b)

x =

    1.8276
    0.6552
    1.9655

>> D*x

ans =

    0.0600
    0.3000
   11.0000
```

Ejercicio 2. Use el algoritmo de eliminación Gaussiana para resolver los siguientes sistemas lineales.

a)

$$E_1: 0.03x_1 + 58.9x_2 = 59.2,$$
$$E_2: 5.31x_1 - 6.10x_2 = 47.0.$$

```
>> A = [0.03 58.9; 5.31 -6.10]

A =

    0.0300   58.9000
    5.3100   -6.1000

>> b = [59.2 47.0]'

b =

   59.2000
```

```
   47.0000

>> x = gauss1(A, b)

x =

   10.0000
    1.0000

>> A*x

ans =

   59.2000
   47.0000
```

b)

$$E_1: 58.9x_1 + 0.03x_2 = 59.2,$$
$$E_2: -6.10x_1 + 5.31x_2 = 47.0.$$

```
>> B = [58.9 0.03; -6.10 5.31]

B =

   58.9000    0.0300
   -6.1000    5.3100

>> b = [59.2 47.0]'

b =

   59.2000
   47.0000

>> x = gauss1(B, b)

x =

    1.0000
   10.0000

>> B*x

ans =

   59.2000
   47.0000
```

c)

$$E_1: x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 = \frac{11}{16,}$$
$$E_2: 5x_1 + \frac{10}{3}x_2 + \frac{5}{2}x_3 = \frac{65}{6,}$$
$$E_3: \frac{100}{3}x_1 + 25x_2 + 20x_3 = \frac{235}{3.}$$

```
>> C = [1 1/2 1/3; 5 10/3 5/2; 100/3 25 20]

C =

    1.0000    0.5000    0.3333
    5.0000    3.3333    2.5000
   33.3333   25.0000   20.0000
```

```
>> b = [11/16 65/6 235/3]'

b =

    0.6875
   10.8333
   78.3333

>> x = gauss1(C, b)

x =

   -9.3125
   42.2500
  -33.3750

>> C*x

ans =

    0.6875
   10.8333
   78.3333
```

Ejercicio 3. Aplique los algoritmos de eliminación Gaussiana a los siguientes sistemas lineales.

a)

$$E_1: 3.3330x_1 + 15920x_2 - 10.333x_3 = 15913,$$
$$E_2: 2.2220x_1 + 16.710x_2 + 9.6120x_3 = 28.544,$$
$$E_3: 1.5611x_1 + 5.1791x_2 + 1.6852x_3 = 8.4254.$$

```
>> A = [3.3330 15920 -10.333; 2.2220 16.710 9.6120; 1.5611 5.1791 1.6852]

A =

  1.0e+004 *

    0.0003    1.5920   -0.0010
    0.0002    0.0017    0.0010
    0.0002    0.0005    0.0002

>> b = [15913 28.544 8.4254]'

b =

  1.0e+004 *

    1.5913
    0.0029
    0.0008

>> x = gauss1(A, b)

x =

    1.0000
    1.0000
    1.0000

>> A*x

ans =

  1.0e+004 *
```

```
1.59130000000000
0.00285440000000
0.00084254000000
```

b)

$$E_1: x_1+\frac{1}{2}x_2+\frac{1}{3}x_3=2,$$
$$E_2: \frac{1}{2}x_1+\frac{1}{3}x_2+\frac{1}{4}x_3=-1,$$
$$E_3; \frac{1}{3}x_1+\frac{1}{4}x_2+\frac{1}{5}x_3=0.$$

```
>> B = hilb(3)

B =

   1.00000000000000   0.50000000000000   0.33333333333333
   0.50000000000000   0.33333333333333   0.25000000000000
   0.33333333333333   0.25000000000000   0.20000000000000

>> b = [2 -1 0]'

b =

     2
    -1
     0

>> x2 = gauss2(B, b);
>> x3 = gauss3(B, b);
>> x4 = gauss4(B, b);
>> x5 = gauss5(B, b);
>> [(B*x1) (B*x2) (B*x3) (B*x4) (B*x5)]'

ans =

   2.00000000000000  -1.00000000000000   0.00000000000001
   2.00000000000000  -0.99999999999999                   0
   2.00000000000000  -0.99999999999999                   0
   2.00000000000001  -0.99999999999999                   0
   2.00000000000000  -1.00000000000000  -0.00000000000001
```

c)

$$E_1: x_1+\frac{1}{2}x_2+\frac{1}{3}x_3+\frac{1}{4}x_4+\frac{1}{5}x_5=1,$$
$$E_2: \frac{1}{2}x_1+\frac{1}{3}x_2+\frac{1}{4}x_3+\frac{1}{5}x_4+\frac{1}{6}x_5=1,$$
$$E_3: \frac{1}{3}x_1+\frac{1}{4}x_2+\frac{1}{5}x_3+\frac{1}{6}x_4+\frac{1}{7}x_5=1,$$
$$E_4: \frac{1}{4}x_1+\frac{1}{5}x_2+\frac{1}{6}x_3+\frac{1}{7}x_4+\frac{1}{8}x_5=1,$$
$$E_5: \frac{1}{5}x_1+\frac{1}{6}x_2+\frac{1}{7}x_3+\frac{1}{8}x_4+\frac{1}{9}x_5=1.$$

```
>> C = hilb(5)

C =

   1.00000000000000   0.50000000000000   0.33333333333333   0.25000000000000   0.20000000000000
```

```
   0.50000000000000   0.33333333333333   0.25000000000000   0.20000000000000   0.16666666666667
   0.33333333333333   0.25000000000000   0.20000000000000   0.16666666666667   0.14285714285714
   0.25000000000000   0.20000000000000   0.16666666666667   0.14285714285714   0.12500000000000
   0.20000000000000   0.16666666666667   0.14285714285714   0.12500000000000   0.11111111111111
```

```
>> b = [1 1 1 1 1]'

b =

     1
     1
     1
     1
     1
```

```
>> x1 = gauss1(C, b);
>> x2 = gauss2(C, b);
>> x3 = gauss3(C, b);
>> x4 = gauss4(C, b);
>> x5 = gauss5(C, b);
>> [b C*x1 C*x2 C*x3 C*x4 C*x5]'

ans =

   1.00000000000003   1.00000000000000   1.00000000000000   1.00000000000003   1.00000000000000
   1.00000000000000   1.00000000000000   1.00000000000001   0.99999999999999   0.99999999999999
   1.00000000000000   1.00000000000000   1.00000000000001   0.99999999999999   0.99999999999999
   0.99999999999999   1.00000000000000   1.00000000000000   1.00000000000000   1.00000000000000
   1.00000000000000   1.00000000000000   1.00000000000001   0.99999999999999   0.99999999999999
   1.00000000000003   1.00000000000000   1.00000000000000   1.00000000000003   1.00000000000001
```

d)

$$
\begin{aligned}
E_1 &: \pi x_1 - e x_2 + \sqrt{2} x_3 - \sqrt{3} x_4 = 1, \\
E_2 &: \pi^2 x_1 + e x_2 - e^2 x_3 + \frac{3}{7} x_4 = -1, \\
E_3 &: \sqrt{5} x_1 - \sqrt{6} x_2 + x_3 - 1.1 x_4 = 0, \\
E_4 &: \pi^3 x_1 + e^2 x_2 - \sqrt{7} x_3 + x_4 = \sqrt{2}
\end{aligned}
$$

```
>> D = [pi -exp(1) sqrt(2) -sqrt(3); pi^2 exp(1) -exp(2) (3/7); sqrt(5)
-sqrt(6) 1 -1.1; pi^3 exp(2) -sqrt(7) 1]

D =

   3.14159265358979  -2.71828182845905   1.41421356237310  -1.73205080756888
   9.86960440108936   2.71828182845905  -7.38905609893065   0.42857142857143
   2.23606797749979  -2.44948974278318   1.00000000000000  -1.10000000000000
  31.00627668029982   7.38905609893065  -2.64575131106459   1.00000000000000
```

```
>> b = [1 -1 0 sqrt(2)]'

b =

   1.00000000000000
  -1.00000000000000
                   0
   1.41421356237310
```

```
>> x1 = gauss1(D, b);
>> x2 = gauss2(D, b);
>> x3 = gauss3(D, b);
>> x4 = gauss4(D, b);
>> x5 = gauss5(D, b);

>> [b D*x1 D*x2 D*x3 D*x4 D*x5]'

ans =

   1.00000000000000  -1.00000000000000                   0   1.41421356237310
   1.00000000000000  -1.00000000000000                   0   1.41421356237309
   1.00000000000000  -1.00000000000000                   0   1.41421356237309
   1.00000000000000  -1.00000000000000  -0.00000000000000   1.41421356237309
   1.00000000000000  -1.00000000000001                   0   1.41421356237309
```

1.00000000000000 -1.00000000000000 0 1.41421356237310

Factorización directa de matrices.

Se dice que $A\ de\ n \times n$ es una matriz estrictamente dominante diagonal en el caso que satisfaga la siguiente condición:

$$|a_{ii}| > \sum_{j=1, i \neq j}^{n} |a_{ij}| \quad para\ i = 1, 2, \ldots, n$$

Si A es una matriz de $x \times x$ estrictamente dominante diagonal, entonces A es no singular. Además, se puede efectuar eliminación Gaussiana en cualquier sistema lineal de la forma $A \cdot x = b$ para obtener su solución única sin intercambios de renglones o columnas, y los cálculos son estables con respecto al crecimiento de los errores de redondeo.

Si el procedimiento de eliminación Gaussiana puede aplicarse al sistema $A \cdot x = b$ sin intercambio de renglones, entonces la matriz A puede factorizarse como el producto de una matriz triangular inferior L con una matriz triangular superior U, $A = L \cdot U$ donde:

$$L = \begin{bmatrix} l_{11} & 0 & 0 & \ldots & 0 \\ l_{21} & l_{22} & 0 & \ldots & 0 \\ l_{31} & l_{32} & l_{33} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ l_{n1} & l_{n2} & l_{n3} & \ldots & l_{nn} \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1n} \\ 0 & u_{22} & u_{23} & \ldots & u_{2n} \\ 0 & 0 & u_{33} & \ldots & u_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & u_{nn} \end{bmatrix}$$

Los elementos de A pueden utilizarse para determinar parcialmente los elementos de L y de U. Sin embargo, si el procedimiento nos debe llevar a una solución única, se necesitan n condiciones adicionales para los elementos de L y de U. En función de cómo se fijen estás condiciones surgen los métodos de Doolittle, Crout y Choleski.
El método de Doolittle exige que todos los elementos de la diagonal principal de L, sean 1, $l_{11} = l_{22} = \ldots = l_{nn} = 1$. El método de Crout impone la condición anterior a la matriz U, $u_{11} = u_{22} = \ldots = u_{nn} = 1$. Finalmente, el método de Choleski requiere que $l_{ii} = u_{ii}$ para cada valor de i = 1, 2, ..., n.

Ejemplo 4. La matriz estrictamente dominante diagonal de 4×4 se puede factorizar en la forma $A = L \cdot U$.

```
>> A=[6 2 1 -1; 2 4 1 0; 1 1 4 -1; -1 0 -1 1]

A =

     6     2     1    -1
     2     4     1     0
     1     1     4    -1
    -1     0    -1     1

>> [L, U, men]=doolittle(A)

L =

    1.0000         0         0         0
    0.3333    1.0000         0         0
```

```
    0.1667    0.2000    1.0000         0
   -0.1667    0.1000   -0.2432    1.0000

U =

    6.0000    2.0000    1.0000   -1.0000
         0    3.3333    0.6667    0.3333
         0         0    3.7000   -0.9000
         0         0         0    0.5811

men =

Procedimiento terminado con éxito
```

Ejemplo 5. La matriz positiva definida A se puede factorizar en la forma $A = L \cdot U$.

```
>> A=[4 -1 1; -1 4.25 2.75; 1 2.75 3.5]

A =

    4.0000   -1.0000    1.0000
   -1.0000    4.2500    2.7500
    1.0000    2.7500    3.5000

>> [L, U, mem]=choleski(A)

L =

    2.0000         0         0
   -0.5000    2.0000         0
    0.5000    1.5000    1.0000


U =

    2.0000   -0.5000    0.5000
         0    2.0000    1.5000
         0         0    1.0000


mem =

Procedimiento terminado con éxito
```

Si A es una matriz de $x \times x$ positiva definida, entonces A tiene una factorización de la forma $A = L \cdot L^t$, donde L es una matriz triangular inferior.

```
>> [L, U, men]=choleskipd(A)

L =

    2.0000         0         0
   -0.5000    2.0000         0
    0.5000    1.5000    1.0000


U =

    2.0000   -0.5000    0.5000
         0    2.0000    1.5000
         0         0    1.0000


men =

Procedimiento terminado con éxito

>> L*U
```

```
ans =

    4.0000   -1.0000    1.0000
   -1.0000    4.2500    2.7500
    1.0000    2.7500    3.5000
```

Para resolver el sistema lineal $A \cdot x = b$ incorporando alguno de los algoritmos de factorización anteriores es necesario que se pueda escribir en la forma $(L \cdot U) \cdot x = b$. La sustitución hacia adelante resuelve el sistema $L \cdot z = b$ y la sustitución hacia atrás resuelve el sistema $U \cdot x = z = L^{-1} \cdot b$.

Ejemplo 6. Resolución del sistema lineal mediante el algoritmo de factorización directa.

$$
\begin{aligned}
E_1 &: 1.00x_1 + 0.333x_2 + 1.50x_3 - 0.333x_4 = 3.00,\\
E_2 &: -2.01x_1 + 1.45x_2 + 0.500x_3 + 2.95x_4 = 5.40,\\
E_3 &: 4.32x_1 - 1.95x_2 \qquad\quad + 2.08x_4 = 0.139,\\
E_4 &: 5.11x_1 - 4.00x_2 + 3.33x_3 - 1.11x_4 = 3.77
\end{aligned}
$$

```
>> A=[1 0.333 1.5 -0.333; -2.01 1.45 0.5 2.95; 4.32 -1.95 0 2.08; 5.11
-4.0 3.33 -1.11]

A =

    1.0000    0.3330    1.5000   -0.3330
   -2.0100    1.4500    0.5000    2.9500
    4.3200   -1.9500         0    2.0800
    5.1100   -4.0000    3.3300   -1.1100

>> b=[3; 5.4; 0.13; 3.77]

b =

    3.0000
    5.4000
    0.1300
    3.7700

>> [L, U, b2, men]=doolittle2(A, b)

L =

    1.0000         0         0         0
    0.8454    1.0000         0         0
    0.1957    0.7794    1.0000         0
   -0.3933   -0.0862    0.5151    1.0000


U =

    5.1100   -4.0000    3.3300   -1.1100
         0    1.4316   -2.8152    3.0184
         0         0    3.0425   -2.4683
         0         0         0    4.0450


b2 =

    3.7700
    0.1300
    3.0000
    5.4000


men =
```

```
Procedimiento terminado con éxito

>> z=L\b2

z =

    3.7700
   -3.0572
    4.6450
    4.2268

>> x=U\z

x =

   -0.3238
    0.3306
    2.3744
    1.0449

>> A*x

ans =

    3.0000
    5.4000
    0.1300
    3.7700

>> [L, U, b2, men]=crout2(A, b)

L =

    5.1100         0         0         0
    4.3200    1.4316         0         0
    1.0000    1.1158    3.0425         0
   -2.0100   -0.1234    1.5672    4.0450


U =

    1.0000   -0.7828    0.6517   -0.2172
         0    1.0000   -1.9665    2.1084
         0         0    1.0000   -0.8113
         0         0         0    1.0000


b2 =

    3.7700
    0.1300
    3.0000
    5.4000


men =

Procedimiento terminado con éxito

>> z=L\b2

z =

    0.7378
   -2.1355
    1.5267
    1.0449

>> x=U\z

x =
```

```
   -0.3238
    0.3306
    2.3744
    1.0449

>> A*x

ans =

    3.0000
    5.4000
    0.1300
    3.7700

>> [L, U, b2, men]=choleski2(A, b)

L =

    2.2605         0         0         0
    1.9111    1.1965         0         0
    0.4424    0.9325    1.7443         0
   -0.8892   -0.1031    0.8985    2.0112


U =

    2.2605   -1.7695    1.4731   -0.4910
         0    1.1965   -2.3529    2.5227
         0         0    1.7443   -1.4151
         0         0         0    2.0112


b2 =

    3.7700
    0.1300
    3.0000
    5.4000


men =

Procedimiento terminado con éxito

>> z=L\b2

z =

    1.6677
   -2.5551
    2.6630
    2.1016

>> x=U\z

x =

   -0.3238
    0.3306
    2.3744
    1.0449

>> A*x

ans =

    3.0000
    5.4000
    0.1300
    3.7700
```

Ejemplo 7. Consideremos el sistema tridiagonal de ecuaciones:

$$
\begin{aligned}
E_1 &: 2x_1 - x_2 = 1,\\
E_2 &: -x_1 + 2x_2 - x_3 = 0,\\
E_3 &: -x_2 + 2x_3 - x_4 = 0,\\
E_4 &: -x_3 + 2x_4 = 1.
\end{aligned}
$$

```
>> A=[2 -1 0 0; -1 2 -1 0; 0 -1 2 -1; 0 0 -1 2]

A =

     2    -1     0     0
    -1     2    -1     0
     0    -1     2    -1
     0     0    -1     2

>> [L, U, men]=crouttrid(A)

L =

    2.0000         0         0         0
   -1.0000    1.5000         0         0
         0   -1.0000    1.3333         0
         0         0   -1.0000    1.2500


U =

    1.0000   -0.5000         0         0
         0    1.0000   -0.6667         0
         0         0    1.0000   -0.7500
         0         0         0    1.0000


men =

Procedimiento terminado con éxito

>> b=[1 0 0 1]'

b =

     1
     0
     0
     1

>> z=L\b

z =

    0.5000
    0.3333
    0.2500
    1.0000

>> x=U\z

x =

    1.0000
    1.0000
    1.0000
    1.0000
```

El algoritmo de reducción de Crout para sistemas lineales tridiagonales puede aplicarse siempre y cuando $a_{ii} \neq 0$ para cada i = 1, 2, ..., n.

Ejercicio. Factorizar las siguientes matrices en la descomposición LU usando el método de

Doolittle.

a)

$$\begin{bmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{bmatrix}$$

```
>> A=[2 -1 1; 3 3 9; 3 3 5]

A =

     2    -1     1
     3     3     9
     3     3     5

>> [L, U, men]=doolittle(A)

L =

    1.0000         0         0
    1.5000    1.0000         0
    1.5000    1.0000    1.0000


U =

    2.0000   -1.0000    1.0000
         0    4.5000    7.5000
         0         0   -4.0000


men =

Procedimiento terminado con éxito

>> L*U

ans =

     2    -1     1
     3     3     9
     3     3     5
```

b)

$$\begin{bmatrix} 2 & -1.5 & 3 \\ -1 & 0 & 2 \\ 4 & -4.5 & 5 \end{bmatrix}$$

```
>> B = [2 -1.5 3; -1 0 2; 4 -4.5 5]

B =

    2.0000   -1.5000    3.0000
   -1.0000         0    2.0000
    4.0000   -4.5000    5.0000

>> [L, U, men]=doolittle(B)

L =

    1.0000         0         0
   -0.5000    1.0000         0
    2.0000    2.0000    1.0000
```

```
U =

    2.0000   -1.5000    3.0000
         0   -0.7500    3.5000
         0         0   -8.0000


men =

Procedimiento terminado con éxito

>> L*U

ans =

    2.0000   -1.5000    3.0000
   -1.0000         0    2.0000
    4.0000   -4.5000    5.0000
```

c)

$$\begin{bmatrix} 1.012 & -2.132 & 3.104 \\ -2.137 & 4.096 & -7.013 \\ 3.104 & -7.013 & 0.014 \end{bmatrix}$$

```
>> C = [1.012 -2.132 3.104; -2.132 4.096 -7.013; 3.104 -7.013 00.14]

C =

    1.0120   -2.1320    3.1040
   -2.1320    4.0960   -7.0130
    3.1040   -7.0130    0.1400

>> [L, U, men]=doolittle(C)

L =

    1.0000         0         0
   -2.1067    1.0000         0
    3.0672    1.1978    1.0000


U =

    1.0120   -2.1320    3.1040
         0   -0.3955   -0.4737
         0         0   -8.8131


men =

Procedimiento terminado con éxito

>> L*U

ans =

    1.0120   -2.1320    3.1040
   -2.1320    4.0960   -7.0130
    3.1040   -7.0130    0.1400
```

d)

$$\begin{bmatrix} 3.107 & 2.101 & 0 \\ 0 & -1.213 & 2.101 \\ 0 & 0 & 2.179 \end{bmatrix}$$

```
>> D = [3.107 2.101 0; 0 -1.213 2.101; 0 0 2.179]

D =

    3.1070    2.1010         0
         0   -1.2130    2.1010
         0         0    2.1790

>> [L, U, men]=doolittle(D)

L =

     1     0     0
     0     1     0
     0     0     1


U =

    3.1070    2.1010         0
         0   -1.2130    2.1010
         0         0    2.1790


men =

Procedimiento terminado con éxito

>> L*U

ans =

    3.1070    2.1010         0
         0   -1.2130    2.1010
         0         0    2.1790
```

e)

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 1.5 & 0 & 0 \\ 0 & -3 & 0.5 & 0 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

```
>> E = [2 0 0 0; 1 1.5 0 0; 0 -3 0.5 0; 2 -2 1 1]

E =

    2.0000         0         0         0
    1.0000    1.5000         0         0
         0   -3.0000    0.5000         0
    2.0000   -2.0000    1.0000    1.0000

>> [L, U, men]=doolittle(E)

L =

    1.0000         0         0         0
    0.5000    1.0000         0         0
         0   -2.0000    1.0000         0
    1.0000   -1.3333    2.0000    1.0000
```

```
U =

    2.0000         0         0         0
         0    1.5000         0         0
         0         0    0.5000         0
         0         0         0    1.0000


men =

Procedimiento terminado con éxito

>> L*U

ans =

    2.0000         0         0         0
    1.0000    1.5000         0         0
         0   -3.0000    0.5000         0
    2.0000   -2.0000    1.0000    1.0000
```

f)

$$\begin{bmatrix} 2.1756 & 4.0231 & -2.1732 & 5.1967 \\ -4.0231 & 6.0000 & 0 & 1.1973 \\ -1.0000 & -5.2107 & 1.1111 & 0 \\ 6.0235 & 7.0000 & 0 & -4.1561 \end{bmatrix}$$

```
>> F = [2.1756 4.0231 -2.1732 5.1967; -4.0231 6.0 0 1.1973; -1.0 -5.2107
1.1111 0; 6.0235 7 0 -4.1561]

F =

    2.1756    4.0231   -2.1732    5.1967
   -4.0231    6.0000         0    1.1973
   -1.0000   -5.2107    1.1111         0
    6.0235    7.0000         0   -4.1561

>> [L, U, men]=doolittle(F)

L =

    1.0000         0         0         0
   -1.8492    1.0000         0         0
   -0.4596   -0.2501    1.0000         0
    2.7687   -0.3079   -5.3523    1.0000


U =

    2.1756    4.0231   -2.1732    5.1967
         0   13.4395   -4.0187   10.8070
         0         0   -0.8930    5.0917
         0         0         0   12.0361


men =

Procedimiento terminado con éxito

>> L*U

ans =

    2.1756    4.0231   -2.1732    5.1967
   -4.0231    6.0000         0    1.1973
   -1.0000   -5.2107    1.1111         0
    6.0235    7.0000         0   -4.1561
```

Ejercicio 2. Resolver los siguientes sistemas lineales.

a) Usando el método de Doolittle comparando las versión sin y con pivoteo máximo de columna.

$$E_1: 2x_1 - x_2 + x_3 = -1,$$
$$E_2: 3x_1 + 3x_2 + 9x_3 = 0,$$
$$E_3: 3x_1 + 3x_2 + 5x_3 = 4.$$

```
>> A = [2 -1 1; 3 3 9; 3 3 5]

A =

     2    -1     1
     3     3     9
     3     3     5

>> b = [ -1 0 4]'

b =

    -1
     0
     4

>> [L, U, men]=doolittle(A)

L =

    1.0000         0         0
    1.5000    1.0000         0
    1.5000    1.0000    1.0000

U =

    2.0000   -1.0000    1.0000
         0    4.5000    7.5000
         0         0   -4.0000

men =

Procedimiento terminado con éxito

>> z=L\b

z =

   -1.0000
    1.5000
    4.0000

>> x=U\z

x =

     1
     2
    -1

>> [L, U, b2, men]=doolittle2(A, b)

L =

    1.0000         0         0
    0.6667    1.0000         0
    1.0000         0    1.0000
```

```
U =

     3     3     9
     0    -3    -5
     0     0    -4

b2 =

     0
    -1
     4

men =

Procedimiento terminado con éxito

>> z=L\b2

z =

     0
    -1
     4

>> x=U\z

x =

     1
     2
    -1

>> A*x

ans =

    -1
     0
     4
```

b) Usando el método de Doolittle comparando las versión sin y con pivote máximo de columna.

$$E_1: 2x_1 = 3,$$
$$E_2: x_1 + 1.5x_2 = 4.5,$$
$$E_3: -3x_2 + 0.5x_3 = -6.6,$$
$$E_4: 2x_1 - 2x_2 + x_3 + x_4 = 0.8$$

```
>> B=[2 0 0 0; 1 1.5 0 0; 0 -3 0.5 0; 2 -2 1 1]

B =

    2.0000         0         0         0
    1.0000    1.5000         0         0
         0   -3.0000    0.5000         0
    2.0000   -2.0000    1.0000    1.0000

>> b=[3 4.5 -6.6 .8]'

b =

    3.0000
    4.5000
   -6.6000
    0.8000

>> [L, U, men]=doolittle(B)
```

```
L =

    1.0000         0         0         0
    0.5000    1.0000         0         0
         0   -2.0000    1.0000         0
    1.0000   -1.3333    2.0000    1.0000

U =

    2.0000         0         0         0
         0    1.5000         0         0
         0         0    0.5000         0
         0         0         0    1.0000

men =

Procedimiento terminado con éxito

>> z=L\b

z =

    3.0000
    3.0000
   -0.6000
    3.0000

>> x=U\z

x =

    1.5000
    2.0000
   -1.2000
    3.0000

>> [L, U, b2, men]=doolittle2(B, b)

L =

    1.0000         0         0         0
         0    1.0000         0         0
    1.0000    0.6667    1.0000         0
    0.5000   -0.5000    0.3750    1.0000

U =

    2.0000         0         0         0
         0   -3.0000    0.5000         0
         0         0    0.6667    1.0000
         0         0         0   -0.3750

b2 =

    3.0000
   -6.6000
    0.8000
    4.5000

men =

Procedimiento terminado con éxito

>> z=L\b2

z =

    3.0000
   -6.6000
    2.2000
   -1.1250

>> x=U\z
```

```
x =

    1.5000
    2.0000
   -1.2000
    3.0000

>> B*x

ans =

    3.0000
    4.5000
   -6.6000
    0.8000
```

c) Utilizando el método de Crout sin y con pivoteo máximo de columna

$$E_1: 1.012x_1 - 2.132x_2 + 3.104x_3 = 1.984,$$
$$E_2: -2.132x_1 + 4.096x_2 - 7.013x_3 = -5.049,$$
$$E_3: 3.104x_1 - 7.013x_2 + 0.014\,x_3 = -3.895.$$

```
>> C = [1.012 -2.132 3.104; -2.132 4.096 -7.013; 3.104 -7.013 0.014]

C =

    1.0120   -2.1320    3.1040
   -2.1320    4.0960   -7.0130
    3.1040   -7.0130    0.0140

>> b = [1.984 -5.049 -3.895]'

b =

    1.9840
   -5.0490
   -3.8950

>> [L, U, men]=crout(C)

L =

    1.0120         0         0
   -2.1320   -0.3955         0
    3.1040   -0.4737   -8.9391

U =

    1.0000   -2.1067    3.0672
         0    1.0000    1.1978
         0         0    1.0000

men =

Procedimiento terminado con éxito

>> z=L\b

z =

    1.9605
    2.1978
    1.0000

>> x=U\z

x =

    1.0000
```

```
    1.0000
    1.0000

>> [L, U, b2, men]=crout2(C, b)

L =

    3.1040         0         0
   -2.1320   -0.7209         0
    1.0120    0.1545    1.5990

U =

    1.0000   -2.2593    0.0045
         0    1.0000    9.7145
         0         0    1.0000

b2 =

   -3.8950
   -5.0490
    1.9840

men =

Procedimiento terminado con éxito

>> z=L\b2

z =

   -1.2548
   10.7145
    1.0000

>> x=U\z

x =

    1.0000
    1.0000
    1.0000

>> C*x

ans =

    1.9840
   -5.0490
   -3.8950
```

D.) Utilizando el método de Crout sin y con pivoteo máximo de columna

$$
\begin{aligned}
E_1: 3.107x_1 + 2.101x_2 &= 1.001,\\
E_2: \quad -1.213x_2 + 2.101x_3 &= 0.000,\\
E_3: \quad 2.179x_3 &= 7.013.
\end{aligned}
$$

```
>> D = [3.107 2.101 0; 0 -1.213 2.101; 0 0 2.179]

D =

    3.1070    2.1010         0
         0   -1.2130    2.1010
         0         0    2.1790

>> b = [1.001 0.0 7.013]'

b =

    1.0010
```

```
     0
 7.0130

>> [L, U, men]=crout(D)

L =

    3.1070         0         0
         0   -1.2130         0
         0         0    2.1790

U =

    1.0000    0.6762         0
         0    1.0000   -1.7321
         0         0    1.0000

men =

Procedimiento terminado con éxito

>> z=L\b

z =

    0.3222
         0
    3.2184

>> x=U\z

x =

   -3.4474
    5.5746
    3.2184

>> [L, U, b2, men]=crout2(D, b)

L =

    3.1070         0         0
         0   -1.2130         0
         0         0    2.1790

U =

    1.0000    0.6762         0
         0    1.0000   -1.7321
         0         0    1.0000

b2 =

    1.0010
         0
    7.0130

men =

Procedimiento terminado con éxito

>> z=L\b2

z =

    0.3222
         0
    3.2184

>> x=U\z

x =
```

```
   -3.4474
    5.5746
    3.2184

>> D*x

ans =

    1.0010
         0
    7.0130
```

E.) Utilizando el método de Choleski sin y con pivoteo máximo de columna

$$
\begin{aligned}
2x_1 - 1.5x_2 + 3x_3 &= 1,\\
-x1 \qquad\quad + 2x_3 &= 3,\\
4x_1 - 4.5x_2 + 5x_3 &= -1.
\end{aligned}
$$

```
>> E = [2 -1.5 3; -1 0 2; 4 -4.5 5]

E =

    2.0000   -1.5000    3.0000
   -1.0000         0    2.0000
    4.0000   -4.5000    5.0000

>> b = [1 3 -1]'

b =

     1
     3
    -1

>> [L, U, men]=choleski(E)

L =

   1.4142                  0                  0
  -0.7071            0 + 0.8660i              0
   2.8284            0 + 1.7321i        0 + 2.8284i

U =

   1.4142            -1.0607                2.1213
        0                  0 + 0.8660i        0 - 4.0415i
        0                  0                  0 + 2.8284i

men =

Procedimiento terminado con éxito

>> z=L\b

z =

   0.7071
        0 - 4.0415i
        0 + 3.5355i

>> x=U\z

x =

   -0.5000
    1.1667
    1.2500

>> [L, U, b2, men]=choleski2(E, b)
```

```
L =

   2.0000                  0                  0
  -0.5000            0 + 1.0607i              0
   1.0000            0 - 0.7071i         1.6330

U =

   2.0000            -2.2500            2.5000
        0                  0 + 1.0607i        0 - 3.0641i
        0                  0                  1.6330

b2 =

    -1
     3
     1

men =

Procedimiento terminado con éxito

>> z=L\b2

z =

  -0.5000
        0 - 2.5927i
   2.0412

>> x=U\z

x =

   -0.5000
    1.1667
    1.2500

>> E*x

ans =

    1.0000
    3.0000
   -1.0000
```

F.) Utilizando el método de Choleski sin y con pivoteo máximo de columna

$$
\begin{aligned}
E_1 &: 2.1756x_1 + 4.0231x_2 - 2.1732x_3 + 5.1967x_4 = 17.102,\\
E_2 &: -4.0231x_1 + 6.0000x_2 + 1.1973x_4 = -6.1593,\\
E_3 &: -1.0000x_1 - 5.2107x_2 + 1.1111x_3 = 3.0004,\\
E_4 &: 6.0235x_1 + 7.0000x_2 - 4.1561x_4 = 0.0000.
\end{aligned}
$$

```
>> F = [2.1756 4.0231 -2.1732 5.1967; -4.0231 6.0 0 1.1973; -1.0 -5.2107
1.1111 0; 6.0235 7.0 0 -4.1561]

F =

    2.1756    4.0231   -2.1732    5.1967
   -4.0231    6.0000         0    1.1973
   -1.0000   -5.2107    1.1111         0
    6.0235    7.0000         0   -4.1561

>> b = [17.102 -6.1593 3.0004 0]'

b =

   17.1020
   -6.1593
    3.0004
```

```
         0

>> [L, U, men]=choleski(F)

L =

   1.4750         0         0         0
  -2.7275    3.6660         0         0
  -0.6780   -0.9169         0 + 0.9450i         0
   4.0838   -1.1289         0 - 5.0577i    3.4693

U =

   1.4750    2.7275   -1.4734    3.5232
        0    3.6660   -1.0962    2.9479
        0         0         0 + 0.9450i         0 -
5.3883i
        0         0         0    3.4693

men =

Procedimiento terminado con éxito

>> z=L\b

z =

  11.5946
   6.9464
        0 -18.2343i
  15.1949

>> x=U\z

x =

    2.9399
    0.0707
    5.6777
    4.3798

>> F*x

ans =

   17.1020
   -6.1593
    3.0004
    0.0000

>> [L, U, b2, men]=choleski2(F, b)

L =

   2.4543         0         0         0
  -1.6392    3.2673         0         0
   0.8865    0.4575         0 + 1.4742i         0
  -0.4075   -1.2391         0 - 0.7537i    1.4996

U =

   2.4543    2.8522         0   -1.6934
        0    3.2673         0   -0.4831
        0         0         0 + 1.4742i         0 -
4.6934i
        0         0         0    1.4996

b2 =

         0
   -6.1593
   17.1020
    3.0004
```

```
men =

Procedimiento terminado con éxito

>> z=L\b2

z =

         0
   -1.8851
         0 -12.1861i
    6.5679

>> x=U\z

x =

    2.9399
    0.0707
    5.6777
    4.3798

>> F*x

ans =

   17.1020
   -6.1593
    3.0004
         0
```

Técnicas iterativas en álgebra matricial

1. Introducción a los método iterativos en general.
2. Métodos de Jacobi, Gauss-Seidel.
3. Convergencia.
4. Metodo de relajación
5. Estimación del error y refinamiento iterativo.

Para resolver sistemas lineales grandes que involucran matrices con un alto porcentaje de componentes cero, matrices esparcidas, frecuentemente se usan técnicas que son iterativas en vez de directas.

1. Técnicas iterativas para resolver sistemas lineales.

Una técnica iterativa para resolver un sistema lineal $A \cdot x = b$ de $n \times n$ empieza con una aproximación inicial $x^{(0)}$ a la solución x, y genera una sucesión de vectores $\{x^{(k)}\}_{k=0}^{\infty}$ que converga a x. La Mayoría de estas técnicas iterativas involucran un proceso que convierte al sistema $A \cdot x = b$ en un sistema equivalente de la forma $x = T \cdot x + c$ para alguna matriz T de $n \times n$.

Una vez seleccionado $x^{(0)}$ la sucesión de vectores a la solución aproximada se genera calculando $x^{(k)} = T \cdot x^{(k-1)} + c$ para cada $k = 1, 2, 3, \ldots$

2. Método iterativo de Jacobi.

Consideremos el sistema lineal $A \cdot x = b$ dado por:

$$\begin{aligned} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n &= b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n &= b_2 \\ &\ldots \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + \ldots + a_{nn} \cdot x_n &= b_n \end{aligned}$$

o en forma matricial:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}$$

Para convertir $A \cdot x = b$ a la forma $x = T \cdot x + c$, despejamos x_i para cada $i = 1, 2, \ldots, n$ siempre y cuando $a_{ii} = 0$, es decir,

$$x_1 = \frac{b_1}{a_{11}} - \frac{a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n}{a_{11}} = \frac{b_1}{a_{11}} - \frac{\sum_{j=2}^{n} a_{1j} \cdot x_j}{a_{11}}$$

$$x_2 = \frac{b_2}{a_{22}} - \frac{a_{21} \cdot x_1 + \ldots + a_{2n} \cdot x_n}{a_{22}} = \frac{b_2}{a_2} - \frac{\sum_{j=1, j \neq 2}^{n} a_{2j} \cdot x_j}{a_{22}}$$

$$\ldots$$

$$x_n = \frac{b_n}{a_{nn}} - \frac{a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + \ldots + a_{n(n-1)} \cdot x_{n-1}}{a_{nn}} = \frac{b_n}{a_{nn}} - \frac{\sum_{j=1}^{n-1} a_{nj} \cdot x_j}{a_{nn}}$$

por tanto la i-ésima incógnita viene definida por la ecuación:

$$x_i = \frac{b_i}{a_{ii}} - \frac{\sum_{j=1, j \neq i}^{n} a_{ij} \cdot x_j}{a_{ii}} \quad para\ i = 1, 2, \ldots$$

Por otra parte, la matriz A puede expresarse como la suma de una matriz diagonal, D, una matriz triangular inferior, L, y una matriz triangular superior U.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \dots & 0 \\ -a_{21} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ -a_{n1} & -a_{n2} & \dots & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \dots & -a_{1n} \\ 0 & 0 & \dots & -a_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

$$A \cdot x = (D - L - U) \cdot x = b$$

$$D \cdot x = (L + U) \cdot x + b$$

$$x = D^{-1} \cdot [(L + U)x + b]$$

forma matricial de la técnica iterativa de Jacobi.
Para este método T y c tienen las siguientes expresiones

$$T = D^{-1} \cdot (L + U) \, y \, c = D^{-1} \cdot b$$

```
% Jacobi.m - Método iterativo de Jacobi.
%
% Resolución de un sistema lineal Ax = b mediante la
% genereación de una sucesión {Pk} que converge a la
% solución, a partir de un punto inicial Po.
% Una condición suficiente para que el método sea aplicable
% es que A sea de diagonal estrictamente dominante.
%
% [x, error] = jacobi(A, x, b)
%
% entradas:
%   A   matriz invertible de orden N x N
%   x0  matriz de orden N x 1: punto inicial
%   b   matriz de orden N x 1
%   tol: tolerancia
%   iteraciones: número de iteraciones realizadas hasta alcanzar la
solución
%   estrategia_pivoteo
%   crtiterio_paro
%
% salidas:
%   x: matriz de orden N x 1 aproximación a la solución Ax = B
%   error: norma del error
%   numero de iteraciones

function [x, error, iteraciones] = jacobi(A, x0, b, tolerancia,
iteraciones, estrategia_pivoteo, criterio_paro)

crt_paro = 'cociente_norma';
estr_piv = 'eliminacion_ceros';
tol = 10^-4;
max_iter = 20;

if nargin > 6
    crt_paro = criterio_paro;
end
if nargin > 5
    estr_piv = estrategia_pivoteo;
end
```

```
if nargin > 4
    max_iter = iteraciones
end
if nargin > 3
    tol = tolerancia;
end

% Comprueba que la matriz A es no singular
if det(A) == 0
    x = [];
    error = 'la matriz es singular';
    iteraciones = 0;
end

% Aplica la estrategia de pivoteo
Aumentada = [A b];
Aumentada = feval(estr_piv, Aumentada);
[m n] = size(Aumentada);
A = Aumentada(1:m, 1:n-1);
b = Aumentada(:, n);

D = diag(diag(A));
L = D - tril(A);
U = D - triu(A);
T = D \ (L + U);
if norm(eig(T), inf) >= 1
    x = [];
    error = 'radio espectral mayor o igual que 1, el método no converge';
    iteraciones = 0;
    return
end

% Norma euclídea de b
nrm2 = norm(b);
if  (nrm2 == 0.0), nrm2 = 1.0; end

% Vector residual de la primer iteración
r = b - A*x0;
error = norm(r) / nrm2;
if (error < tol)
    x = x0;
    iteraciones = 0;
    return
end

x(:, 1) = x0;
for i = 2 : max_iter
    x(:, i) = D \ ((L + U) * x(:, i - 1) + b);        % nueva aproximación
    error = feval(crt_paro, x(:, i), x(:, i - 1));   % computa el error
    if (error <= tol ), break, end                    % comprueba la
convergencia
end
iteraciones = i;
if (error > tol)
    error='no converge';
end
```

Ejemplo 1. El sistema lineal $A \cdot x = b$ dado por:

$$
\begin{aligned}
E_1 &: 10 \cdot x_1 - x_2 + 2 \cdot x_3 = 6 \\
E_2 &: -x_1 + 11 \cdot x_2 - x_3 + 3 \cdot x_4 = 25 \\
E_3 &: 2 \cdot x_1 - x_2 + 10 \cdot x_3 - x_4 = -11 \\
E_4 &: 3 \cdot x_2 - x_3 + 8 \cdot x_4 = 15
\end{aligned}
$$

tiene por solución a $x = (1, 2, -1, 1)^t$. Tomando como aproximación inicial

$x^{(0)} = (0,0,0,0)^t$ y una tolerancia de 10^{-4} se genera la siguiente sucesión de aproximaciones:

0	0.6000	1.0473	0.9326	1.0152	0.9890	1.0032	0.9981	1.0006	0.9997	1.0001
0	2.2727	1.7159	2.0533	1.9537	2.0114	1.9922	2.0023	1.9987	2.0004	1.9998
0	-1.1000	-0.8052	-1.0493	-0.9681	-1.0103	-0.9945	-1.0020	-0.9990	-1.0004	-0.9998
0	1.8750	0.8852	1.1309	0.9738	1.0214	0.9944	1.0036	0.9989	1.0006	0.9998

0.9999	1.0000
2.0001	2.0000
-1.0001	-1.0000
1.0001	1.0000

La última aproximacióncon comete un error de 8.7971e-005.

El algoritmo iterativo de Jacobi requiere que $a_{ii} \neq 0$ para cada $i = 1, 2, \ldots, n$. Si este no es el caso, se puede realizar un ordenamiento de las ecuaciones para que ninguna $a_{ii} = 0$, a menos que el sistema sea singular.

Se sugiere que las ecuaciones sean arregladas de tal manera que a_{ii} sea lo más grande posible para acelerar la convergencia.

Método iterativo de Gauss-Seidel.

Un análisis de la ecuación

$$x_i^{(k)} = \frac{\sum_{j=1, j\neq i}^{n} (-a_{ij} \cdot x_j^{(k-1)}) + b_i}{a_{ii}}$$

sugiere una posible mejora para calcular $x_i^{(k)}$, usar las componentes de $x_i^{(k-1)}$. Para $i > 1$, $x_1^{(k)}, \ldots, x_{i-1}^{(k)}$ ya han sido calculadas y supuestamente son mejores aproximaciones a la solución real $x_1, \ldots, x_{j-1}$ que $x_1^{(k-1)}, \ldots, x_{i-1}^{(k-1)}$, parece razonable calcular $x_i^{(k)}$ usando los valores calculados más recientes, es decir,

$$x_1^{(k)} = \frac{b_1}{a_{11}} - \frac{a_{12} \cdot x_2^{(k-1)} + \ldots + a_{1n} \cdot x_n^{(k-1)}}{a_{11}} = \frac{b_1}{a_{11}} - \frac{\sum_{j=2}^{n} a_{1j} \cdot x_j^{(k-1)}}{a_{11}}$$

$$x_2^{(k)} = \frac{b_2}{a_{22}} - \frac{a_{21} \cdot x_1^{(k)} + \ldots + a_{2n} \cdot x_n^{(k)}}{a_{22}} = \frac{b_2}{a_2} - \frac{a_{21} \cdot x_1^{(k)}}{a_{22}} - \frac{\sum_{j=3}^{n} a_{2j} \cdot x_j^{(k-1)}}{a_{22}}$$

$$\ldots$$

$$x_n = \frac{b_n}{a_{nn}} - \frac{a_{n1} \cdot x_1^{(k)} + a_{n2} \cdot x_2^{(k)} + \ldots + a_{n(n-1)} \cdot x_{n-1}^{(n-1)}}{a_{nn}} = \frac{b_n}{a_{nn}} - \frac{\sum_{j=1}^{n-1} a_{nj} \cdot x_j^{(k)}}{a_{nn}}$$

$$x_i^{(k)} = \frac{b_{ii} - \sum_{j=1}^{i-1} (a_{ij} \cdot x_j^{(k)}) - \sum_{j=1+i}^{n} (a_{ij} \cdot x_j^{(k-1)})}{a_{ii}}$$

para cada $i = 1, 2, \ldots, n$

Multiplicando ambos miembros de la ecuación por a_{ii}

$$a_{ii} \cdot x_i^{(k)} = b_{ii} - \sum_{j=1}^{i-1} (a_{ij} \cdot x_j^{(k)}) - \sum_{j=1+i}^{n} (a_{ij} \cdot x_j^{(k-1)})$$

agrupando incógnias

$$a_{ii} \cdot x_i^{(k)} + \sum_{j=1}^{i-1} (a_{ij} \cdot x_j^{(k)}) = b_{ii} - \sum_{j=1+i}^{n} (a_{ij} \cdot x_j^{(k-1)})$$

y desarrollando los sumatorios

$$a_{il} \cdot x_1^{(k)} + a_{i2} \cdot x_2^{(k)} + \ldots + a_{ii} \cdot x_i^{(k)} = -a_{i,i+1} \cdot x_{i+1}^{(k-1)} - \ldots - a_{i,n} \cdot x_n^{(k-1)} + b_i$$

para

$$\begin{aligned}
&i = 1 : a_{11} \cdot x_1^{(k)} = -a_{12} \cdot x_2^{(k-1)} - \ldots - a_{1n} \cdot x_n^{(k-1)} + b_1 \\
&i = 2 : a_{21} \cdot x_1^{(k)} + a_{22} \cdot x_2^{(k)} = -a_{23} \cdot x_3^{(k-1)} - \ldots - a_{2n} \cdot x_n^{(k-1)} + b_2 \\
&\ldots \\
&i = n : a_{n1} \cdot x_1^{(k)} + a_{n2} \cdot x_2^{(k)} + \ldots + a_{nn} \cdot x_n^{(k)} = b_n
\end{aligned}$$

En forma matricial:

$$\begin{bmatrix} a_{11} & 0 & \ldots & 0 & 0 \\ a_{21} & a_{22} & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_n n-1 & a_{nn} \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \ldots \\ x_n^{(k)} \end{bmatrix} =$$

$$\begin{bmatrix} 0 & -a_{12} & -a_{13} & \ldots & -a_{1n} \\ 0 & 0 & -a_{23} & \ldots & -a_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ \ldots \\ x_n^{(k-1)} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}$$

$$(D - L) \cdot x^{(k)} = U \cdot x^{(k-1)} + b$$

despejando $x^{(k)}$ se obtiene la forma matricial de la técnica iterativa de Gauss-Seidel.

$$x^{(k)} = (D - L)^{-1} \cdot [U \cdot x^{(k-1)} + b]$$

```
% GS.m - Método iterativo de Gauss-Seidel
%
% Resolución de un sistema lineal Ax = b mediante la
% genereación de una sucesión {Pk} que converge a la
% solución, a partir de un punto inicial Po.
% Una condición suficiente para que el método sea aplicable
% es que A sea de diagonal estrictamente dominante.
%
% [x, error] = gs(A, x, b)
%
% entradas:
%   A   matriz invertible de orden N x N
%   x0  matriz de orden N x 1: punto inicial
%   b   matriz de orden N x 1
%
% salidas:
%   x: matriz de orden N x 1 aproximación a la solución Ax = B
%   error: norma del error
function [x, error, iteraciones] = gs(A, x0, b, tolerancia, iteraciones,
estrategia_pivoteo, criterio_paro)

crt_paro = 'cociente_norma';
estr_piv = 'eliminacion_ceros';
tol = 10^-4;
max_iter = 20;

if nargin > 6
```

```
    crt_paro = criterio_paro;
end
if nargin > 5
    estr_piv = estrategia_pivoteo;
end
if nargin > 4
    max_iter = iteraciones
end
if nargin > 3
    tol = tolerancia;
end

% Comprueba que la matriz A es no singular
if det(A) == 0
    x = [];
    error = 'la matriz es singular';
    iteraciones = 0;
end

% Aplica la estrategia de pivoteo
Aumentada = [A b];
Aumentada = feval(estr_piv, Aumentada);
[m n] = size(Aumentada);
A = Aumentada(1:m, 1:n-1);
b = Aumentada(:, n);

D = diag(diag(A));
L = D - tril(A);
U = D - triu(A);
T = (D - L) \ U;
if norm(eig(T), inf) >= 1
    x = [];
    error = 'radio espectral mayor o igual que 1, el método no converge';
    iteraciones = 0;
    return
end

% Norma euclídea de b
nrm2 = norm(b);
if  (nrm2 == 0.0), nrm2 = 1.0; end

% Vector residual de la primer iteración
r = b - A*x0;
error = norm(r) / nrm2;
if (error < tol)
    x=x0;
    iteraciones=0;
    return
end

x(:, 1) = x0;
for i = 2 : max_iter
    x(:, i) = (D - L) \ (U * x(:, i - 1) + b);       % nueva aproximación
    error = feval(crt_paro, x(:, i), x(:, i - 1));  % computa el error
    if (error <= tol ), break, end                   % comprueba la
convergencia
end
iteraciones = i;
if (error > tol)
    error='no converge';
end
```

Ejemplo 2. El método iterativo de Gauss-Seidel aplicado a la resolución del sistema lineal

$$E_1: 10 \cdot x_1 - x_2 + 2 \cdot x_3 = 6$$
$$E_2: -x_1 + 11 \cdot x_2 - x_3 + 3 \cdot x_4 = 25$$
$$E_3: 2 \cdot x_1 - x_2 + 10 \cdot x_3 - x_4 = -11$$
$$E_4: 3 \cdot x_2 - x_3 + 8 \cdot x_4 = 15$$

genera los siguientes vectores iterados:

0	0.6000	1.0302	1.0066	1.0009	1.0001	1.0000
0	2.3273	2.0369	2.0036	2.0003	2.0000	2.0000
0	-0.9873	-1.0145	-1.0025	-1.0003	-1.0000	-1.0000
0	0.8789	0.9843	0.9984	0.9998	1.0000	1.0000

con un error en la aproximación final de 3.4251e-005, mientras que, el método de Jacobi para resolver el sistema, con una precisión del mismo orden, requiere el doble de interaciones.

3. Convergencia de las técnicas de iteración.

Consideremos la fórmula:

$$x^{(k)} = T \cdot x^{(k-1)} + c$$

para $k = 1, 2, \ldots$ donde $x^{(0)}$ es arbitrario.
Si el radio espectral $p(T) < 1$, entonces $(I - T)^{-1}$ existe y
$(I - T)^{-1} = I + T + T^2 + \ldots$
Para cualquier $x^{(0)} \in \mathbb{R}^n$ la sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ definida por $x^{(k)} = T \cdot x^{(k-1)} + c$ para cada $k \geq 1$ y $c \neq 0$ converge a la solución única de $x = T \cdot x + c$ si y solo si $p(T) < 1$.

Para Jacobi: $T_j = D^{-1} \cdot (L + U)$ y para Gauss-Seidel: $T_g = (D - L)^{-1} \cdot U$

Además, si A es estrictamente dominante diagonalmente, entonces, para cualquier elección de $x^{(0)} = 0$ ambos métodos, el de Jacobi o el de Gauss-Seidel, dan lugar a sucesiones $\{x^{(k)}\}_{k=0}^{\infty}$ que convergen a la solución de $A \cdot x = b$

4. Métodos de relajación.

Si $\tilde{x} \in \mathbb{R}^n$ es una aproximación a la solución del sistema lineal definido por $A \cdot x = b$, el vector residual de $\tilde{x}$ con respecto a este sistema se define como $r = b - A \cdot \tilde{x}$

Supongamos que tomamos a

$$r_i^{(k)} = (r_{1i}^k, r_{2i}^k, \ldots, r_i^{(k)})'$$

para denotar al vector residual para el método de Gauss-Seidel correspondiente al vector solución aproximado

$$(x_1^{(k)}, x_2^{(k)}, \ldots, x_i - 1^{(k)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)})'$$

La i-ésima componente de $r_i^{(k)} es$

$$r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k)} - \sum_{j=1}^{n} a_{ij} \cdot x_j^{(k-1)}$$

o

$$r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} \cdot x_j^{(k-1)} - a_{ij} \cdot x_i^{(k-1)}$$

así que

$$a_{ij} \cdot x_i^{(k-1)} + r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} \cdot x_j^{(k-1)}$$

Recuérdese que en el método de Gauss-Seidel

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

así que

$$a_{ii} \cdot x_i^{(k-1)} + r_{ii}^{(k)} = a_{ii} \cdot x_i^{(k)}$$

o

$$x_i^{(k)} = x_i^{(k-1)} + \frac{r_{ii}^{(k)}}{a_{ii}}$$

modificando el procedimiento de Gauss-Seidel en la forma de la ecuación

$$x_i^{(k)} = x_i^{(k-1)} + w \cdot \frac{r_{ii}^{(k)}}{a_{ii}}$$

para ciertas elecciones de w positivo nos llevará a una convergencia significativamente más rápida.

Los métodos que emplean esta ecuación se conocen como métodos de relajación. Para $0 < w < 1$, los procedimientos se llaman metodos de subrelajación y se pueden emplear para obtener la convergencia de algunos sistemas que no son convergentes por el método de Gauss-Seidel. Para $w > 1$ los procedimientos se llaman métodos de sobrerelajación y se pueden usar para acelerar la convergencia de sistemas que son convergentes por el método de Gauss-Seidel. Estos métodos se abrevian como SOR y son particularmente útiles para resolver los sistemas lineales que aparecen en la solución numérica de ciertas ecuaciones diferenciales parciales.

La ecuación

$$x_i^{(k)} = x_i^{(k-1)} + w \cdot \frac{r_{ii}^{(k)}}{a_{ii}}$$

se puede reformular como

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

para propósitos de cómputo.

Para determinar la forma matricial del método SOR reescribimos la ecuación anterior como

$$a_{ii} \cdot x_i^{(k)} + \omega \cdot \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k)} = (1-\omega) \cdot a_{ii} \cdot x_i^{(k-1)} - \omega \sum_{j=i+1}^{n} a_{ij} \cdot x_j^{(k-1)} + \omega \cdot b_i$$

así que

$$(D - \omega L) \cdot x^{(k)} = [(1-\omega) D + \omega U] \cdot x^{(k-1)} + \omega \cdot b$$

o

$$x^{(k)} = (D - \omega L)^{-1} [(1-\omega) D + \omega U] \cdot x^{(k-1)} + \omega (D - \omega L)^{-1} \cdot b$$

Si $a_{ii} \neq 0$ para cada $i = 1, 2, \ldots, n$, entonces $\rho(T_\omega) \geq |\omega - 1|$. Esto implica que $\rho(T_\omega) < 1$ sólo si $0 < \omega < 2$, donde

$$T_\omega = (D - \omega L)^{-1} [(1-\omega) D + \omega U]$$

es la matriz de iteración del método SOR.

Si A es una matriz positiva definida y $0 < \omega < 2$, entonces el método SOR converge para cualquier elección de la aproximación inicial $x^{(0)}$ del vector solución.

```
% SOR.m - Método SOR
%
% Resolución de un sistema lineal Ax = b mediante la
% genereación de una sucesión {Pk} que converge a la
% solución, a partir de un punto inicial Po.
% Una condición suficiente para que el método sea aplicable
% es que A sea de diagonal estrictamente dominante.
%
% [x, error] = sor(A, x, b)
%
% entradas:
%   A   matriz invertible de orden N x N
%   x0  matriz de orden N x 1: punto inicial
%   b   matriz de orden N x 1
%   w   factor de relajación
%
% salidas:
%   x: matriz de orden N x 1 aproximación a la solución Ax = B
%   error: norma del error
%   w: factor de relajación.

function [x, error, w] = sor(A, x0, b, omega, tolerancia, iteraciones,
estrategia_pivoteo, criterio_paro)

crt_paro = 'cociente_norma';
estr_piv = 'eliminacion_ceros';
tol = 10^-4;
max_iter = 20;
w = 0;

if nargin > 7
    crt_paro = criterio_paro;
end
if nargin > 6
    estr_piv = estrategia_pivoteo;
```

```
end
if nargin > 5
    max_iter = iteraciones
end
if nargin > 4
    tol = tolerancia;
end
if nargin > 3
    w = omega;
end

% Comprueba que la matriz A es no singular
if det(A) == 0
    x = [];
    error = 'la matriz es singular';
    iteraciones = 0;
end

% Aplica la estrategia de pivoteo
Aumentada = [A b];
Aumentada = feval(estr_piv, Aumentada);
[m n] = size(Aumentada);
A = Aumentada(1:m, 1:n-1);
b = Aumentada(:, n);

D = diag(diag(A));
L = D - tril(A);
U = D - triu(A);

% Calcula w, si fuera necesario
if w == 0
    w = escoge_w(D, L, U);
end

T = (D - w * L) \ ((1 - w) * D + w * U);
if norm(eig(T), inf) >= 1
    x = [];
    error = 'radio espectral mayor o igual que 1, el método no converge';
    iteraciones = 0;
    return
end

% Norma euclídea de b
nrm2 = norm(b);
if  (nrm2 == 0.0), nrm2 = 1.0; end

% Vector residual de la primer iteración
r = b - A*x0;
error = norm(r) / nrm2;
if (error < tol) return, end

x(:, 1) = x0;
for i = 2 : max_iter
    x(:, i) = (D - w*L) \ (((1 - w) * D + (w * U)) * x(:, i - 1) + w * b);
% nueva aproximación
    error = feval(crt_paro, x(:, i), x(:, i - 1));  % computa el error
    if (error <= tol ), break, end                  % comprueba la
convergencia
end
global iter; iter = i;
if (error > tol) flag = 1; end                      % no convergence

function w=escoge_w(D, L, U)
w = 1;
Ri = 1;
for j = 0.1:0.01:2
    T = (D - j*L) \ ((1 - j)*D + j*U);
    Rj = max(abs(eig(T)));
    if Rj < Ri
        w = j;
        Ri = Rj;
    end
end
```

Ejemplo 4. El sistema lineal dado por

$$\begin{aligned} 4 \cdot x_1 + 3 \cdot x_2 &= 24 \\ 3 \cdot x_1 + 4 \cdot x_2 - x_3 &= 30 \\ -x_2 + 4 \cdot x_3 &= -24 \end{aligned}$$

tiene la solución $(3, 4, -5)^t$.
Empleando el método de Gauss-Seidel se obtiene la siguiente sucesión

1.0000	5.2500	3.1406	3.0879	3.0549	3.0343	3.0215	3.0134	3.0084	3.0052	3.0033
1.0000	3.8125	3.8828	3.9268	3.9542	3.9714	3.9821	3.9888	3.9930	3.9956	3.9973
1.0000	-5.0469	-5.0293	-5.0183	-5.0114	-5.0072	-5.0045	-5.0028	-5.0017	-5.0011	-5.0007

3.0020	3.0013	3.0008
3.9983	3.9989	3.9993
-5.0004	-5.0003	-5.0002

con un error igual a 8.9415e-005.
En cambio, para el método SOR con $\omega = 1.25$

1.0000	6.3125	2.6223	3.1333	2.9571	3.0037	2.9963	3.0000	2.9997
1.0000	3.5195	3.9585	4.0103	4.0075	4.0029	4.0009	4.0003	4.0001
1.0000	-6.6501	-4.6004	-5.0967	-4.9735	-5.0057	-4.9983	-5.0003	-4.9999

con un error de 8.2259e-005.

5. Estimaciones de error y refinamiento iterativo.

Parece razonable que si $\tilde{x}$ es una aproximación a la solución de x de $A \cdot x = b$ y el vector residual $r = A \cdot \tilde{x} - b$ tiene la propiedad de que $\|r\|$ es pequeño, entonces $\|x - \tilde{x}\|$ será también pequeño. Sin embargo, ciertos sistemas especiales no tienen esta propiedad.

Si $\tilde{x}$ es una aproximación a la solución $A \cdot x = b$ y A es una matriz no singular, entonces para cualquier norma natural,

$$\|x - \tilde{x}\| \leq \|r\| \cdot \|A^{-1}\|$$

y

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|r\|}{\|b\|}$$

Siempre que $x \neq 0 \, y \, b \neq 0$, donde r es el vector residual de $\tilde{x}$ con respecto al sistema $A \cdot x = b$

El número de condición K(A) de la matriz no singular A relativo a una norma natural se define como

$$K(A) = \|A\| \cdot \|A^{-1}\|$$

con esta notación, las desigualdades anteriores se transforman en

$$\|x - \tilde{x}\| \leq K(A) \cdot \frac{\|r\|}{\|A\|}$$

y

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq K(A) \cdot \frac{\|r\|}{\|b\|}$$

ya que para cualquier matriz no-singular A

$$1 = \|I\| = \|A \cdot A^{-1}\| \leq \|A\| \|A^{-1}\| = K(A)$$

se espera que la matriz A tenga un buen comportamiento (llamada formalmente una matriz bien condicionada) si K(A) está cerca de uno y un comportamiento defectuoso (llamada mal condicionada) cuando K(A) sea significativamente mayor que uno. El comportamiento en esta situación se refiere a la relativa seguridad de que un vector residual pequeño implique correspondientemente una solución aproximada precisa.

La aproximación del número de condicón K(A) a t dígitos viene de considerar el sistema lineal

$$A \cdot y = r$$

La solución de este sistema puede aproximarse fácilmente ya que los multiplicadores para el método de eliminación gaussiana han sido ya calculados y supuestamente retenidos. De hecho $\tilde{y}$, la solución aproximada de $A \cdot y = r$, satisface que

$$\tilde{y} \approx A^{-1} \cdot r = A^{-1}(b - A \cdot \tilde{x}) = A^{-1} \cdot b - A^{-1} A \tilde{x} = x - \tilde{x}$$

así que $\tilde{y}$ es una estimación del error cometido al aproximar la solución del sistema original.

A lo largo de esta sección se ha venido suponiendo que, en el sistema lineal $A \cdot x = b$, A y b se pueden representar exactamente. En realida, las componentes a_{ij} y b_j pueden estar alteradas o perturbadas por una cantidad $\delta a_{ij} y \delta b_j$, causando que se tenga que resolver el sistema lineal

$$(A + \delta A) \cdot x = b + \delta b$$

Normalmente, si $\|\delta A\| y \|\delta b\|$ son pequeñas, del orden de 10^{-t}, la aritmética de t dígitos debe generar una solución $\tilde{x}$ para la cual $\|x - \tilde{x}\|$ es correspondientemente pequeña. Sin embargo, en el caso de las matrices mal condicionadas hemos visto que aun cuando A y b se representen exactamente, los errores de redondeo pueden causar que $\|x - \tilde{x}\|$ sea grande.

Supóngase que A es no singular y que

$$\|\delta A\| < \frac{1}{\|A^{-1}\|}$$

La solución $\tilde{x}$ de $(A + \delta A) \cdot \tilde{x} = b + \delta b$ aproxima a la solución x de $A \cdot x = b$ con una estimación de error

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq K \frac{(A)}{1 - K(A)(\frac{\|\delta(A)\|}{\|A\|})} \left(\frac{\|\delta(b)\|}{\|b\|} + \frac{\|\delta(A)\|}{\|A\|} \right)$$

```
% refit2.m - Refinamiento iterativo
%
% [x, error] = refit(A, x0, b)
%
% entradas:
%   A:   matriz invertible de orden N x N
%   x0:  matriz de orden N x 1: punto inicial
%   b:   matriz de orden N x 1
%   alg: algoritmo iterativo para resolver el sistema lineal Ax = b,
(expresado como cadena).
%
% salidas:
%   x: matriz de orden N x 1 aproximación a la solución Ax = B
%   error: norma del error
%
function [x, error, iter] = refit2(A, x0, b, algoritmo, tolerancia,
iteraciones, criterio_paro)

alg = 'sor';
tol = 10^-4;
max_iter = 20;
crt_paro = 'cociente_norma';

if nargin > 6
    crt_paro = criterio_paro;
end
if nargin > 5
    max_iter = iteraciones
end
if nargin > 4
    tol = tolerancia;
```

```
end
if nargin > 3
    alg = algoritmo;
end
iter = 0;

% Norma euclídea de b
nrm2 = norm(b);
if  (nrm2 == 0.0), nrm2 = 1.0; end

% Vector residual de la primer iteración
r = b - A*x0;
error = norm(r) / nrm2;
if (error < tol)
    x = x0;
    return
end

% Resolución del sistem Ax=b.
[sol, error] = feval(alg, A, x0, b);
if sol == []
    x = [];
    return
end
x(:, 1) = sol(:, length(sol));

for i = 2 : max_iter
    r = b - A * x(:, i-1);

    [sol, error] = feval(alg, A, x(:, i-1), r);
    if sol == [] return, end
    y(:, i-1) = sol(:, length(sol));
    x(:, i) = x(:, i-1) + y(:, i-1);
    if i > 1
        error = feval(crt_paro, x(:, i), x(:, i - 1));  % computa el error
        if (error <= tol ), break, end                  % comprueba la
convergencia
    end
end
iter = i;
```

Ejemplo. Resolver el siguientes sistemas lineales con x0 = (0, 0, 0)'

```
>> H = [4 -1 0 -1 0 0; -1 4 -1 0 -1 0; 0 -1 4 0 0 -1; -1 0 0 4 -1 0; 0 -1
0 -1 4 -1; 0 0 -1 0 -1 4]

H =

     4    -1     0    -1     0     0
    -1     4    -1     0    -1     0
     0    -1     4     0     0    -1
    -1     0     0     4    -1     0
     0    -1     0    -1     4    -1
     0     0    -1     0    -1     4

>> b = [0 5 0 6 -2 6]'

b =

     0
     5
     0
     6
    -2
     6

>> x0 = zeros(6, 1)

x0 =

     0
     0
```

```
     0
     0
     0
     0

>> [x_j, error_j] = jacobi(H, x0, b)

x_j =

  Columns 1 through 5

                  0                   0   0.68750000000000
0.62500000000000   0.88671875000000
                  0   1.25000000000000   1.12500000000000
1.73437500000000   1.67968750000000
                  0                   0   0.68750000000000
0.62500000000000   0.88671875000000
                  0   1.50000000000000   1.37500000000000
1.81250000000000   1.77343750000000
                  0  -0.50000000000000   0.56250000000000
0.46875000000000   0.83984375000000
                  0   1.50000000000000   1.37500000000000
1.81250000000000   1.77343750000000

  Columns 6 through 10

   0.86328125000000   0.95874023437500   0.95019531250000
0.98497009277344   0.98185729980469
   1.90332031250000   1.88330078125000   1.96478271484375
1.95748901367188   1.98717117309570
   0.86328125000000   0.95874023437500   0.95019531250000
0.98497009277344   0.98185729980469
   1.93164062500000   1.91748046875000   1.97509765625000
1.96994018554688   1.99092864990234
   0.80664062500000   0.94165039062500   0.92956542968750
0.97874450683594   0.97434234619141
   1.93164062500000   1.91748046875000   1.97509765625000
1.96994018554688   1.99092864990234

  Columns 11 through 15

   0.99452495574951   0.99339103698730   0.99800556898117
0.99759250879288   0.99927347525954
   1.98451423645020   1.99532675743103   1.99435889720917
1.99829764664173   1.99794507771730
   0.99452495574951   0.99339103698730   0.99800556898117
0.99759250879288   0.99927347525954
   1.98904991149902   1.99669551849365   1.99601113796234
1.99879625439644   1.99854695051908
   0.99225711822510   0.99065351486206   0.99717944860458
0.99659529328346   0.99897253885865
   1.98904991149902   1.99669551849365   1.99601113796234
1.99879625439644   1.99854695051908

  Columns 16 through 20

   0.99912300705910   0.99973534396850   0.99968053190969
0.99990359197545   0.99988362521981
   1.99937987234443   1.99925143970177   1.99977410194697
1.99972731692833   1.99991771060377
   0.99912300705910   0.99973534396850   0.99968053190969
0.99990359197545   0.99988362521981
   1.99956150352955   1.99947068793699   1.99984026595484
1.99980718395091   1.99994181260990
   0.99875974468887   0.99962571985088   0.99954820389394
0.99986365846416   0.99983542120754
   1.99956150352955   1.99947068793699   1.99984026595484
1.99980718395091   1.99994181260990


error_j =

    9.519960742776062e-005
```

```
>> [x_g, error_g] = gs(H, x0, b)

x_g =

  Columns 1 through 5

                   0                   0   0.68750000000000
0.81640625000000   0.92944335937500
                   0   1.25000000000000   1.54687500000000
1.83300781250000   1.93914794921875
                   0   0.31250000000000   0.79296875000000
0.92797851562500   0.97398376464844
                   0   1.50000000000000   1.71875000000000
1.88476562500000   1.95715332031250
                   0   0.18750000000000   0.72265625000000
0.89916992187500   0.96327209472656
                   0   1.62500000000000   1.87890625000000
1.95678710937500   1.98431396484375

  Columns 6 through 10

   0.97407531738281   0.99054241180420   0.99655395746231
0.99874463304877   0.99954269570298
   1.97783279418945   1.99192500114441   1.99705846607685
1.99892846774310   1.99960966577055
   0.99053668975830   0.99655359983444   0.99874461069703
0.99954269430600   0.99983341440384
   1.98433685302734   1.99429082870483   1.99792006611824
1.99924231506884   1.99972399219405
   0.98662090301514   0.99512630701065   0.99822462722659
0.99935327307321   0.99976441245235
   1.99428939819336   1.99791997671127   1.99924230948091
1.99972399184480   1.99989945671405

  Columns 11 through 12

   0.99983341449115   0.99993931676818
   1.99985781033683   1.99994820361940
   0.99993931676272   0.99997789451169
   1.99989945673588   1.99996337442872
   0.99991418094669   0.99996873811887
   1.99996337442735   1.99998665815764


error_g =

    5.295149174911819e-005

>> [x_s, error_s, w] = sor(H, x0, b)

x_s =

  Columns 1 through 5

                   0                   0   0.86240000000000
0.86879744000000   0.99302237798400
                   0   1.40000000000000   1.66790400000000
1.95989749760000   1.99412468039680
                   0   0.39200000000000   0.94481408000000
0.98751184896000   0.99835381192458
                   0   1.68000000000000   1.80454400000000
1.95895275520000   1.99773473177600
                   0   0.30240000000000   0.90083840000000
0.98129570201600   0.99846547445514
                   0   1.87443200000000   1.97185085440000
1.99464401174528   1.99975211877689

  Columns 6 through 8

   0.99855795005030   0.99985078156230   0.99999774884291
   1.99941066455279   1.99996050535304   1.99999751436137
   0.99996312190136   0.99998253423312   1.00000085332148
```

```
   1.99943839104841   1.99996750404118   2.00000156493352
   0.99979247189125   0.99999391336587   0.99999992600693
   1.99996131200870   1.99999804788667   2.00000045246555


error_s =

    7.348358280544553e-005


w =

   1.12000000000000

>> [H*x_j(:, length(x_j)) H*x_g(:, length(x_g)) H*x_s(:, length(x_s))]

ans =

  -0.00032502233444  -0.00015431097540  -0.00000808392326
   5.00006817076792   4.99990686507886   4.99999152927416
  -0.00032502233444  -0.00002328373029   0.00000544645899
   6.00004820401227   5.99994544282782   6.00000858488422
  -2.00045965099343  -2.00002328373029  -1.99999982773271
   6.00004820401227   6.00000000000000   6.00000103053381

Solución del mismo sistema lineal empleando el método de refinamiento
iterativo

>> [x, error]=refit2(H, x0, b)

x =

   0.99999774884291   1.00000000000110
   1.99999751436137   1.99999999999932
   1.00000085332148   1.00000000000016
   2.00000156493352   1.99999999999978
   0.99999992600693   1.00000000000024
   2.00000045246555   2.00000000000019


error =

    1.242818973623430e-006

>> H*x(:, 2)

ans =

   0.00000000000529
   4.99999999999577
   0.00000000000111
   5.99999999999778
  -1.99999999999833
   6.00000000000037
```

Aproximación de valores característicos

1. Álgebra lineal y valores característicos.
2. Método de las potencias.
3. Método de las potencias inversas.
4. Método de Householder.
5. Algoritmo QR.

1. Álgebra lineal y valores característicos.

Una matriz A de $n\,x\,n$ tiene precisamente n valores característicos, no necesariamente distintos, que son las raíces del polinomio $p(\lambda)=det(A-\lambda\cdot I)$. Teóricamente los valores característicos de A se pueden obtener encontrando las n raíces de $p(\lambda)$ y posteriormente, resolviendo el sistema lineal asociado, se determinan los vectores característicos correspondientes. En la práctica $p(\lambda)$ es difícil de obtener y, de cualquier modo, la determinación de las raíces de un polinomio de n-ésimo grado puede ser difícil también, excepto para valores pequeños de n. Como consecuencia, es necesario desarrollar técnicas de aproximación para encontrar los valores característicos.

2. Método de la potencia.

Se trata de una técnica de naturaleza iterativa para aproximar autovalores. Para aplicar el método de la potencia debemos suponer que la matriz A de $n\,x\,n$ tiene n valores característicos $\lambda_1,\lambda_2,\ldots,\lambda_n$ con una colección asociada de vectores característicos $\{v^{(1)},v^{(2)},\ldots,v^{(n)}\}$ que es linealmente independiente. Además, supondremos que A tiene presicamente un valor característico que es el mayor en magnitud y que, por esta razón, supondremos que $\lambda_1,\lambda_2,\ldots,\lambda_n$ denotan a los valores característicos de A con $|\lambda_1|>|\lambda_2|\geq\ldots\geq|\lambda_n|\geq 0$.

Supongamos que la matriz A tiene un autovalor dominante, λ , y que existe un único autovector real normalizado, V, correspondiente a *lambda*. Este para autovalor-autovector (λ,V) puede encontrarse mediante el siguiente procedimiento iterativo. Empezamos con el vector $x_0=[1\,1\ldots 1]^t$ y generamos recursivamente una sucesion $\{x_k\}$ de acuerdo con el siguiente esquema:

$$Y_k=A\cdot X_k$$
$$X_{k+1}=\frac{1}{c_{k+1}}Y_k$$

donde c_{k+1} es la coordenada Y_k de mayor tamaño. Las sucesiones $\{X_k\}$ y $\{c_k\}$ convergerán respectivamente a V y λ:

$$\lim_{k\to\infty}x_k=V \quad \text{y} \quad \lim_{k\to\infty}c_k=\lambda$$

Si x_0 se escoge adecuadamente, entonces las sucesiones $\left\{x_k=\left[x_1^{(k)},x_2^{(k)},\ldots,x_n^{(k)}\right]^t\right\}$ y $\{c_k\}$ generadas recursivamente con el esquema iterativo:

$$Y_k=A\cdot X_k$$
$$X_{k+1}=\frac{1}{c_{k+1}}Y_k$$

Codificación en Matlab

```
function [lambda, V, x, nu]=potencias(A, x0, tolerancia, iteraciones,
criterio_paro)
```

```
tol = 10^-4;
max_iter = 20;
crt_paro = 'diferencia_normas';

if nargin > 4
    crt_paro = criterio_paro;
end

if nargin > 3
    max_iter = iteraciones;
end
if nargin > 2
    tol = tolerancia;
end

lambda = 0;
error = 1;

i = 1;
x(:, i) = x0
while (i < max_iter)
    y = A * x(:, i);
    % Normalización de y
    yp = max(abs(y));
    nu(i) = yp;
    y = (1/yp)*y;

    error = feval(crt_paro, x(:, i), y);  % computa el error
    if (error <= tol ), break, end        % comprueba la convergencia

    i = i + 1;
    x(:, i) = y;
end
if i > max_iter lambda = nu(max_iter);
else lambda = nu(i);
V = x(:, i);
```

Ejemplo La matriz A:

$$A = \begin{bmatrix} -4 & 14 & 0 \\ -5 & 13 & 0 \\ -1 & 0 & 2 \end{bmatrix}$$

tiene autovaleres $\lambda_1 = 6$, $\lambda_2 = 3$, y $\lambda_3 = 2$, por tanto, el método de las potencias convergerá al autovalor dominante λ_1 .

```
>> A = [-4 14 0; -5 13 0; -1 0 2]

A =

    -4    14     0
    -5    13     0
    -1     0     2

>> [lambda, V, x, nu] = potencias(A)

lambda =

    6.0004

V =

    1.0000
    0.7143
   -0.2499
```

La siguiente tabla muestra los valores de las aproximaciones al autovector y al autovalor en cada

una de las iteraciones:

```
>> [x' nu']

ans =

    1.0000    1.0000    1.0000   10.0000
    1.0000    0.8000    0.1000    7.2000
    1.0000    0.7500   -0.1111    6.5000
    1.0000    0.7308   -0.1880    6.2308
    1.0000    0.7222   -0.2209    6.1111
    1.0000    0.7182   -0.2359    6.0545
    1.0000    0.7162   -0.2431    6.0270
    1.0000    0.7152   -0.2466    6.0135
    1.0000    0.7148   -0.2483    6.0067
    1.0000    0.7145   -0.2492    6.0034
    1.0000    0.7144   -0.2496    6.0017
    1.0000    0.7143   -0.2498    6.0008
    1.0000    0.7143   -0.2499    6.0004
```

La sucesión anterior converge linealmente. Se puede utilizar el procedimiento de Aitken Δ^2 para acelerar la convergencia.

```
% potencias_aitken.m - Algoritmo del método de potencia mejorado con el
procedimiento de Aitken.
%
% Para aproximar el valor característico dominante y un vector
característico asociado de la matriz
% A de n x n dado un vector no cero x.
%
% Entradas:
%  matriz A,
%  tolerancia,
%  número de iteraciones
%  criterio de paro,
%       (una cadena de caracteres que indica el nombre del archivo que
contiene el criterio de paro)
%
% Salidas:
%   valor característico, lambda
%   vector característico, V
%   sucesión de aproximaciones al vector característico, x
%   sucesión de aproximaciones al autovalor, nu
function [lambda, V, x, nu]=potencias_aitken(A, x0, tolerancia,
iteraciones, criterio_paro)
[m n] = size(A);

tol = 10^-4;
max_iter = 20;
crt_paro = 'diferencia_norma';

if nargin > 4
    crt_paro = criterio_paro;
end

if nargin > 3
    max_iter = iteraciones;
end
if nargin > 2
    tol = tolerancia;
end

lambda = 0;
error = 1;

i = 1;
x(:, i) = x0;
nu_0 = 0;
nu_1 = 0;
while (i <= max_iter)
    y = A * x(:, i);
```

```
    % Normalización de y
    yp = max(abs(y));
    y = (1/yp)*y;

    nu = yp;
    nu_tilde(i) = nu_0 - ((nu_1 - nu_0)^2 / (nu - 2*nu_1 + nu_0));

    error = feval(crt_paro, x(:, i), y);  % computa el error
    if (error <= tol ), break, end  % comprueba la convergencia

    i = i + 1;
    x(:, i) = y;
    nu_0 = nu_1;
    nu_1 = nu;
end
if i > max_iter lambda = nu(max_iter);
else lambda = nu(i);
V = x(:, i);
nu = nu_tilde;
```

```
>> [lambda2, V2, x2, nu2] = potencias_aitken(A)

lambda2 =

    6.0000


V2 =

    1.0000
    0.7143
   -0.2499
```

La siguiente tabla muestra las aproximaciones al autovector V, al autovalor dominante λ_1 y al mismo autovalor con el algoritmo de las potencias modificado con el procedimiento Δ^2 de Aitken.

```
>> [x' nu' nu2']

ans =

    1.0000    1.0000    1.0000   10.0000         0
    1.0000    0.8000    0.1000    7.2000    7.8125
    1.0000    0.7500   -0.1111    6.5000    6.2667
    1.0000    0.7308   -0.1880    6.2308    6.0625
    1.0000    0.7222   -0.2209    6.1111    6.0154
    1.0000    0.7182   -0.2359    6.0545    6.0038
    1.0000    0.7162   -0.2431    6.0270    6.0010
    1.0000    0.7152   -0.2466    6.0135    6.0002
    1.0000    0.7148   -0.2483    6.0067    6.0001
    1.0000    0.7145   -0.2492    6.0034    6.0000
    1.0000    0.7144   -0.2496    6.0017    6.0000
    1.0000    0.7143   -0.2498    6.0008    6.0000
    1.0000    0.7143   -0.2499    6.0004    6.0000
```

Cuando A es una matriz simétrica se puede hacer una variación en la elección de los vectores $x^{(m)}, y^{(m)}$, y de los escalares $\mu^{(m)}$ para mejorar mucho la rapidez de convergencia de la sucesión $\{muy^{(m)}\}_{m=1}^{\infty}$ al valor característico dominante λ_1.

```
% potencias_msimetrica.m - Algoritmo del método de potencia para matrices
% simétricas
%
% Para aproximar el valor característico dominante y un vector
% característico asociado de la matriz simétrica A de n x n dado un vector
% no cero x.
%
% Entradas:
%  matriz A,
%  vector inicial, x0
```

```
%  tolerancia,
%  número de iteraciones
%  criterio de paro,
%       (una cadena de caracteres que indica el nombre del archivo que
%        contiene el criterio de paro)
%
% Salidas:
%   valor característico, lambda
%   vector característico, V
%   sucesión de aproximaciones al vector característico, x
%   sucesión de aproximaciones al autovalor, nu

function [lambda, V, x, nu]=potencias_msimetrica(A, x0, tolerancia,
iteraciones, criterio_paro)
x0 = x0';

tol = 10^-4;
max_iter = 20;
crt_paro = 'diferencia_norma_euclidea';

if nargin > 4
    crt_paro = criterio_paro;
end

if nargin > 3
    max_iter = iteraciones;
end
if nargin > 2
    tol = tolerancia;
end

lambda = 0;
error = 1;

i = 1;
x(:, i) = x0 / norm(x0);
while (i < max_iter)
    y = A * x(:, i);

    nu(i) = x(:, i)' * y;

    error = feval(crt_paro, x(:, i), y / norm(y));  % computa el error
    if (error <= tol ), break, end                  % comprueba la
convergencia

    i = i + 1;
    x(:, i) = y / norm(y);
end
if i > max_iter lambda = nu(max_iter);
else lambda = nu(i);
V = x(:, i);
```

Ejemplo 2. La matriz

$$A=\begin{bmatrix} 4 & -1 & 1 \\ -1 & 3 & -2 \\ 1 & -2 & 3 \end{bmatrix}$$

es simétrica, sus valores característicos son $\lambda_1 = 6, \lambda_2 = 3, y \lambda_3 = 1.$

```
>> [lambda, V, x, nu] = potencias_aitken(A, [1 0 0]');
>> lambda, V

lambda =

    6.0000


V =
```

```
    1.0000
   -0.9998
    0.9998

>> [lambda2, V2, x2, nu2] = potencias_msimetrica(A, [1 0 0]');
>> lambda, V2

lambda2 =

    6.0000


V2 =

    1.0000
   -0.9998
    0.9998

>> [x(:, 1:10)' nu(:, 1:10)']

ans =

    1.0000         0         0         0
    1.0000   -0.2500    0.2500    4.5714
    1.0000   -0.5000    0.5000      -Inf
    1.0000   -0.7000    0.7000    7.0000
    1.0000   -0.8333    0.8333    6.2000
    1.0000   -0.9118    0.9118    6.0476
    1.0000   -0.9545    0.9545    6.0118
    1.0000   -0.9769    0.9769    6.0029
    1.0000   -0.9884    0.9884    6.0007
    1.0000   -0.9942    0.9942    6.0002

>> [x2(:, 1:10)' nu2(:, 1:10)']

ans =

    1.0000         0         0    4.0000
    0.9428   -0.2357    0.2357    5.0000
    0.8165   -0.4082    0.4082    5.6667
    0.7107   -0.4975    0.4975    5.9091
    0.6470   -0.5392    0.5392    5.9767
    0.6128   -0.5588    0.5588    5.9942
    0.5952   -0.5682    0.5682    5.9985
    0.5863   -0.5728    0.5728    5.9996
    0.5819   -0.5751    0.5751    5.9999
    0.5796   -0.5762    0.5762    6.0000
```

Haciendo una lista de los resultados de las 10 primeras iteraciones, vemos que usando el segundo algoritmo los resultados son mejores.

3. El método de potencia inverso.

Para aplicar este método es necesario disponer de una buena aproximación inicial a un autovalor y lo que se consigue mediante las iteraciones del método de las potencias inversas es obtener una solución más precisa. Cuando el autovalor es complejo, cuando es múltiple o cuando hay una pareja de autovalores del mismo tamaño aparecen dificultades computacionales que deben abordarse usando métodos más avanzados.

El método de las potencias inversas con traslación se basa en los siguientes teoremas.

Autovalores trasladados. Supongamos que (λ, X) es una pareja autovalor-autovector de una matriz A. Si α es una constante cualquier, entonces $(\lambda - \alpha, V)$ es una pareja autovalor-autovector de la matriz $A - \alpha \cdot I$.

Autovalores inversos. Supongamos que (λ, X) es una pareja autovalor-autovector de una matriz A. Si α es un autovalor de A, entonces $(\frac{1}{\lambda - alphaa}, V)$ es una pareja autovalor-autovector de la matriz $(A - \alpha \cdot I)^{-1}$.

Método de las potencias inversas con traslación. Supongamos que una matriz A de orden $n \times n$ posee n autovalores distintos $\lambda_1, \lambda_2, \ldots, \lambda_n$ y fijemos uno de estos autovalores λ_j . Entonces existe una constante $alphaa$ tal que $\mu_1 = \frac{1}{\lambda_j - alphaa}$ es el autovalor dominante de la matriz $(A - \lambda \cdot I)^{-1}$. Es más, si X_0 se escoge adecuadamente, entonces las sucesiones $\left\{X_k = \left[x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right]^t\right\}$ y $\{c_k\}$ generadas recursivamente por las fórmulas

$$Y_k = (A - alphaa \cdot I)^{-1} X_k$$

y

$$X_k + 1 = \frac{1}{c_k + 1} Y_k$$

donde

$$c_{k+1} = x_j^{(k)} \; y \; x_j^{(k)} = max\left\{\left|x_i^{(k)}\right| : 1 \le j \le n\right\}$$

convergen a la pareja autovalor-autovector dominante (μ_1, V_j) de la matriz $(A - alphaa \cdot I)^{-1}$. Finalmente, el autovalor de A que hemos fijado viene dado por

$$\lambda_j = \frac{1}{muy_1} + alphaa$$

```
% potencias_inversas.m - Algoritmo del método de potencia inverso.
%
% Para aproximar el valor característico y un vector característico
asociado de la matriz
% A de n x n dado un vector no cero x.
%
% Entradas:
%  matriz A,
```

```
%  vector inicial, x0'
%  valor aproximado del autovalor,q
%  tolerancia,
%  número de iteraciones
%  criterio de paro,
%        (una cadena de caracteres que indica el nombre del archivo que
contiene el criterio de paro)
%
% Salidas:
%    valor característico, lambda
%    vector característico, V
%    sucesión de aproximaciones al vector característico, x
%    sucesión de aproximaciones al autovalor, nu
function [lambda, V, x, nu]=potencias_inversas(A, x0, q, tolerancia,
iteraciones, criterio_paro, metodo_resolucion)
tol = 10^-4;
max_iter = 20;
crt_paro = 'diferencia_norma';
mr = 'factorizacion_triangular';

if nargin > 6
    mr = metodo_resolucion;
end

if nargin > 5
    crt_paro = criterio_paro;
end

if nargin > 4
    max_iter = iteraciones;
end
if nargin > 3
    tol = tolerancia;
end

[n n] = size(A);
A = A - q * eye(n);
[L U P] = lu(A);

lambda = 0;
error = 1;

% Encuentra el entero p con 1 <= p <= n y |xp| = norm(x, inf)
[m j]=max(abs(x0));
xp = x0(j);

i = 1;
x(:, i) = (1/xp)*x0;
while (i < max_iter)
    % Resuelve el sistema Ay = x
    y = feval(mr, A, x(:, i));

    % Normalización de y
    [m j]=max(abs(y));
    yp = y(j);
    y = (1/yp)*y;
    nu(i) = yp;

    error = feval(crt_paro, x(:, i), y);  % computa el error
    if (error <= tol ), break, end        % comprueba la convergencia

    i = i + 1;
    x(:, i) = y;
end
lambda = (1/nu(i)) + q;
V = x(:, i);
```

Ejemplo 3. Calcular el autovalor-autovector dominante de la matriz

$$A = \begin{bmatrix} 0 & 11 & -5 \\ -2 & 17 & -7 \\ -4 & 26 & -10 \end{bmatrix}$$

Para ello usaremos el método de las potencias con convergencia alecerada de Aitken para obtener una primera aproximación a los autovalores de A.

```
>> [lambda, V] = potencias_aitken(A, [1 1 1]', 10^-6, 5)

lambda =

    4.0202

V =

    0.4038
    0.6026
    1.0000

>> [lambda, V, x, nu]=potencias_inversas(A, V, lambda)

lambda =

    4.0000

V =

    0.4000
    0.6000
    1.0000


x =

    0.4038    0.4000
    0.6026    0.6000
    1.0000    1.0000


nu =

  -50.7565  -49.5123
```

Ejemplo 4. La matriz

$$\begin{bmatrix} -4 & 14 & 0 \\ -5 & 13 & 0 \\ -1 & 0 & 2 \end{bmatrix}$$

tiene como pareja autovalor-autovector dominante:

```
>> A = [-4 14 0; -5 13 0; -1 0 2]

A =

    -4    14     0
    -5    13     0
    -1     0     2

>> [lambda, V] = potencias_aitken(A, [1 1 1]', 10^-6, 5)

lambda =

   6.01538461538463
```

```
V =

   1.00000000000000
   0.71818181818182
  -0.23591470258137

>> [lambda_i, V_i, x, nu] = potencias_inversas(A, V, lambda, 10^-6)

lambda_i =

   6.00000000711484


V_i =

   1.00000000000000
   0.71428581389349
  -0.24999964490022


x =

   1.00000000000000   1.00000000000000   1.00000000000000
   0.71818181818182   0.71430523917995   0.71428581389349
  -0.23591470258137  -0.24992997337866  -0.24999964490022


nu =

 -66.17578849717980 -65.00589233411806 -65.00003006016777
```

Como la convergencia del método de potencia inverso es lienal, se puede usar el procedimiento de Δ^2 de Aitken para acelerar la convergencia.

```
% potencias_inversas_aitken.m - Algoritmo del método de potencia inverso
mejorado con Aitken.
%
% Para aproximar el valor característico y un vector característico
asociado de la matriz
% A de n x n dado un vector no cero x.
%
% Entradas:
%  matriz A,
%  vector inicial, x0
%  valor aproximado del autovalor, q
%  tolerancia,
%  número de iteraciones
%  criterio de paro,
%       (una cadena de caracteres que indica el nombre del archivo que
contiene el criterio de paro)
%
% Salidas:
%   valor característico, lambda
%   vector característico, V
%   sucesión de aproximaciones al vector característico, x
%   sucesión de aproximaciones al autovalor, nu

function [lambda, V, x, nu]=potencias_inversas_aitken(A, x0, q,
tolerancia, iteraciones, criterio_paro, metodo_resolucion)
tol = 10^-4;
max_iter = 20;
crt_paro = 'diferencia_norma';
mr = 'factorizacion_triangular';

if nargin > 6
    mr = metodo_resolucion;
end

if nargin > 5
    crt_paro = criterio_paro;
end
```

```
if nargin > 4
    max_iter = iteraciones;
end
if nargin > 3
    tol = tolerancia;
end

[n n] = size(A);
A = A - q * eye(n);
[L U P] = lu(A);

lambda = 0;
error = 1;

% Encuentra el entero p con 1 <= p <= n y |xp| = norm(x, inf)
[m j]=max(abs(x0));
xp = x0(j);

i = 1;
nu_0 = 0;
nu_1 = 0;
x(:, i) = (1/xp)*x0;
while (i < max_iter)
    % Resuelve el sistema Ay = x
    y = feval(mr, A, x(:, i));

    % Normalización de y
    [m j]=max(abs(y));
    yp = y(j);
    nu = yp;
    y = (1/yp)*y;
    nu_tilde(i) = nu_0 - ((nu_1 - nu_0)^2 / (nu - 2*nu_1 + nu_0));


    error = feval(crt_paro, x(:, i), y);  % computa el error
    if (error <= tol ), break, end        % comprueba la convergencia

    i = i + 1;
    x(:, i) = y;
    nu_0 = nu_1;
    nu_1 = nu;
end
nu = nu_tilde;
lambda = (1/nu(i)) + q;
V = x(:, i);
```

```
>> [lambda_i, V_i, x, nu] = potencias_inversas_aitken(A, V, lambda, 10^-6)

lambda_i =

   6.00000000012705


V_i =

   1.00000000000000
   0.71428581389349
  -0.24999964490022


x =

   1.00000000000000   1.00000000000000   1.00000000000000
   0.71818181818182   0.71430523917995   0.71428581389349
  -0.23591470258137  -0.24992997337866  -0.24999964490022


nu =

                   0 -65.02621519577210 -65.00000053675252
```

4. Método de Householder.

Permite encontrar una matriz simétrica tridiagonal cuyos valores característicos coinciden con los de la matriz original.

Si los vectores X e Y tienen la misma norma, entonces existe una matriz ortogonal y simétrica P tal que Y = PX donde $P = I - 2WW^t$ y $W = \frac{X-Y}{\|X-Y\|_2}$ ya que P es ortogonal y simétrica, es decir, $P^{-1} = P$.

El efecto de la aplicación Y = PX es reflejar X tomando la dirección z como eje de reflexión.

El método de Householder empieza determinando una matriz $P^{(1)}$ con $P^{(1)} = \left[P^{(1)}\right]^{-1}$. La matriz $P^{(1)}$ se define por

$$P^{(1)} = I - 2w \cdot w^t$$

para un vector especialmente definido w. Este vector

$$w = (w_1, w_2, \ldots, w_{n-1}, 0)^t$$

debe satisfacer que $w \cdot w^t = 1$ y además, las componentes $w_1, w_2, \ldots, w_{n-1}$ se seleccionan de tal manera que la matriz resultante $A^{(2)} = P^{(1)} \cdot A \cdot P^{(1)}$ tenga $a_{n,j}^{(2)} = 0$ para cada j = 1, 2, ..., n-2. Por simetría esto también implica que $a_{j,n}^{(2)} = 0$ para cada j = 1, 2, ..., n-2. Esto impone n-1 condiciones en las n-1 incógnitas $w_1, w_2, \ldots, w_{n-1}$. La última componente de w se escoge como cero para dejar invariante el elemento $a_{n,n}$ de A.

Se produce una matriz tridiagonal $A^{(n-1)}$ repitiendo este procedimiento n-3 veces, donde el k-ésimo paso se describe por

$$A^{(k+1)} = P^{(k)} \cdot A^{(k)} \cdot P^{(k)}$$

para una matriz $P^{(k)}$ que se escoge con las propiedades

$$\left[P^{(k)}\right]^{-1} = P^{(k)}$$

y

$$a_{n-k+1,j}^{(k+1)} = a_{j,n-k+1}^{(k+1)} = 0 \quad \text{para cada j = 1, 2, ..., n-k-1.}$$

```
% householder.m - Algoritmo de Householder.
%
% Para obtener una matriz simétrica tridiagonal A(n-1) similar a la matriz inicial A = A(1)
%
function A=householder(A)
[n n] = size(A);
for k=1:n-2
```

```
    i = n - k + 2;
    q = norm(A(i-1, 1:i-2), 2);
    if A(i-1, i-2) == 0
       s = q;
    else
       s = q * A(i-1, i-2) / abs(A(i-1, i-2));
    end
    r = q^2 + s * A(i-1, i-2);
    v(i-1) = 0;
    v(i-2) = A(i-1, i-2) + s;
    if i > 3
       for j = 1:i-3
          v(j) = A(i-1, j);
       end
    end
    v = v';
    w = (1/sqrt(2*r))*v;
    u = A(1:i-1, 1:i-1)*(v/r);
    z = u-w*w'*u;
    A(1:i-1, 1:i-1) = A(1:i-1, 1:i-1) - v*z'-z*v';
    clear v w u;
end
```

Ejemplo 5. La matriz de 4×4 :

$$A = \begin{bmatrix} 4 & -2 & 1 & 2 \\ -2 & 3 & 0 & -2 \\ 1 & 0 & 2 & 1 \\ 2 & -2 & 1 & -1 \end{bmatrix}$$

es simétrica. Utilizando el algoritmo de Householder obtenemos una matriz tridiagonal, simétrica y que tiene los mismos valores característicos que A.

```
>> A = [4 -2 1 2; -2 3 0 -2; 1 0 2 1; 2 -2 1 -1]

A =

     4    -2     1     2
    -2     3     0    -2
     1     0     2     1
     2    -2     1    -1

>> H=householder(A)

H =

    2.2615   -0.0923   -0.0000         0
   -0.0923    1.1829    0.8958         0
   -0.0000    0.8958    5.5556   -3.0000
         0         0   -3.0000   -1.0000
```

5. Método QR.

Sea A una matriz real y simétrica de orden n, A puede factorizarse como A = Q R donde Q es una matriz ortogonal y R una matriz triangular superior.

$$A_1 = A$$
$$A_1 = Q_1 \cdot R_1 \qquad como\ Q_1\ es\ ortogonal: Q_1^t \cdot A_1 = Q_1^t \cdot Q_1 \cdot R_1 = R_1$$

$$A_2 = R_1 \cdot Q_1$$
$$A_2 = Q_1^t \cdot A_1 \cdot Q_1$$

puesto que $Q_1^t = Q_1^{-1}$, A_2 es semejante a la matriz inicial A_1 . En el paso general:

$$A_k = Q_k \cdot R_k$$
$$A_{k+1} = R_k \cdot Q_k = Q_k^t \cdot A_k \cdot Q_k$$

Factorización QR

Sea a una matriz cuadra de orden n y $x = a_{il} \qquad para\ i = 1, 2, \ldots, n$; x es la primera columna de A, entonces, podemos encontrar una matriz de Householder P_1 de manera que todas las componentes del vector $P_1 \cdot x$ desde la segunda hasta la última son iguales a cero:

$$P_1 \cdot A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22} & \ldots & a_{2n} \\ 0 & a_{32} & \ldots & a_{3n} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & a_{n2} & \ldots & a_{nn} \end{bmatrix}$$

De forma similar, se puede obtener una matriz de Householder P_2 de manera que los elementos de las dos primeras columnas de la matriz producto $P_2 \cdot P_1 \cdot A$ que están por debajo de la diagonal principal son iguales a cero.

$$P_2 \cdot P_1 \cdot A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22} & \ldots & a_{2n} \\ 0 & 0 & \ldots & a_{3n} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & a_{nn} \end{bmatrix}$$

Después de n-1 paso obtenemos:

$$P_{n-1} \cdot \ldots \cdot P_2 \cdot P_1 \cdot A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22} & \ldots & a_{2n} \\ 0 & 0 & \ldots & a_{3n} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & a_{nn} \end{bmatrix}$$

Puesto que cada matriz de Householder es ortogonal, $Q = P_1^t \cdot P_2^t \cdot \ldots \cdot P_{n-1}^t$.

Aceleración mediante traslaciones.

Si λ_j es un autovalor de A, entonces $\lambda_j - s$ es un autovalor de A – sI. Partiendo de la matriz simétrica y tridiagonal $A_1 = A$ se calcula:

$$A_i - s_i \cdot I = Q_i \cdot R_i$$
$$A_i + 1 = R_i \cdot Q_i \qquad para\ i = 1, 2, \ldots, k_j$$

donde $\{s_i\}$ es una sucesión cuya suma aproxima al autovalor $\lambda_j : \sigma_j = \sum_{i=1}^{k_j} s_i$.

El valor adecuado de la traslación se calcula en cada paso usando los elementos de la esquina inferior derecha de la matriz. Para el primer autovalor, λ_1 , se calculan los autovalores de la submatriz de orden 2×2 extraida de la esquina inferior derecha

$$\begin{bmatrix} d_{n-1} & e_{n-1} \\ e_{n-1} & d_n \end{bmatrix}$$

Los autovalores de esta matriz son las raíces $x_1 y x_2 de:$

$$x^2 - (d_{n-1} + d_n)x + d_{n-1} d_n - e_{n-1} e_{n-1} = 0$$

Tomamos como valor s_1 del primer desplazamiento la raíz de la ecuación que está más cerca de d_n .

La iteración QR con traslación se repite hasta que tengamos $e_{n-1} \approx 0$, digamos que esto ocurre en el paso k_1 , momento en el que tomamos como aproximación al primer autovalor $\lambda_1 \approx s_1 + s_2 + \ldots + s_{kl}$. Aplicamos el mismo proceso a la submatriz que resulta de eliminar la última fila y la última columna, hasta obtener $e_{n-2} \approx 0$, lo que nos proporciona la aproximación al segundo autovalor, λ_2 . Seguimos con las iteraciones, que se van aplicando a matrices cada vez más pequeñas, hasta que obtengamos $e_2 \approx 0$ y la aproximación al autovalor λ_{n-2} . Finalmente, los autovalores de la matrizde orden 2×2 restante se calculan usando la fórmula para la resolución de la ecuación de segundo grado.

```
function D=qr2(M , tol)
[n n]=size(M);
A = M;

i = 0;
while (n > 2)
```

```
    lambda = 0;
    i = i + 1;
    % La iteración QR con traslación se repite hasta que A(n, n-1) sea
    % aproximadamente 0
    while (abs(A(n, n-1)) > tol)
        % Los cuatro elementos de la esquina inferior derecha son:
        B = A(n-1:n, n-1:n);

        % con ellos formamos la ecuación de segundo grado cuyos
        % coeficientes son los elementos de v:
        v=[1 -(B(1, 1) + B(2, 2)) B(1, 1)*B(2, 2)-B(1, 2)*B(2, 1)];

        % Las raíces de esta ecuación son:
        r=roots(v);

        % Elegimos la más cercana a A(n, n) como el valor del
        % desplazamiento
        [j k]=min(abs(A(n, n)*[1 1]' - r));
        s=r(k);

        % La matriz trasladada es:
        A = A - s*eye(n);

        % Calculamos su factorización QR
        [Q R] = qr(A);

        % La matriz que resulta de multiplicar R por Q es:
        A = R*Q;

        lambda = lambda + s;
    end
    % Almacenamos el i-ésimo autovalor
    D(i) = lambda;
    if (i > 1)
        D(i) = D(i) + D(i-1);
    end
    n = n - 1;
    A = A(1:n, 1:n);
end

% Se calculan los autovalores de la matriz de orden 2 x 2 final
B = A(n-1:n, n-1:n);

% Formamos la ecuación de segundo grado cuyos coeficientes son los
elementos de v:
v=[1 -(B(1, 1) + B(2, 2)) B(1, 1)*B(2, 2)-B(1, 2)*B(2, 1)];

% Las raíces de esta ecuación son:
r=roots(v);

% Los dos ultimos autovalores vienen dados por:
D(i+1) = D(i) + r(1);
D(i+2) = D(i) + r(2);
```

Ejemplo 6

$$M = \begin{bmatrix} 4 & 2 & 2 & 1 \\ 2 & -3 & 1 & 1 \\ 2 & 1 & 3 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}$$

```
>> M = [4 2 2 1; 2 -3 1 1; 2 1 3 1; 1 1 1 2]

M =

     4     2     2     1
     2    -3     1     1
     2     1     3     1
     1     1     1     2
```

```
>> vc=qr2(M, 10^-6)

vc =

    1.3542    1.6458   -3.6458    6.6458
```

Ejercicio. Calcular los autovalores y los autovectores de la siguiente matriz simétrica.

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

```
>> A

A =

     4     3     2     1
     3     4     3     2
     2     3     4     3
     1     2     3     4

>> H = householder(A)

H =

    0.6747   -0.2254    0.0000         0
   -0.2254    3.0396    2.6030         0
    0.0000    2.6030    8.2857   -3.7417
         0         0   -3.7417    4.0000

>> vc = qr2(H, 10^-6)

vc =

    0.9010    0.5858   11.0990    3.4142

>> [lambda, x] = potencias_inversas_aitken(H, [1 1 1 1]', 0.9010)

lambda =

    0.9010

x =

    0.9089
   -0.9123
    0.8283
    1.0000

>> H*x

ans =

    0.8188
   -0.8220
    0.7462
    0.9010

>> lambda*x

ans =

    0.8189
   -0.8220
    0.7463
    0.9010
```

```
>> [lambda, x] = potencias_inversas_aitken(H, [1 1 1 1]', vc(2))

lambda =

    0.5858

x =

    1.0000
    0.3945
   -0.2853
   -0.3127

>> H*x

ans =

    0.5858
    0.2311
   -0.1671
   -0.1832

>> lambda*x

ans =

    0.5858
    0.2311
   -0.1671
   -0.1832

>> [lambda, x] = potencias_inversas_aitken(H, [1 1 1 1]', vc(3))

lambda =

   11.0990

x =

   -0.0070
    0.3232
    1.0000
   -0.5271

>> H*x

ans =

   -0.0776
    3.5869
   11.0990
   -5.8499

>> lambda*x

ans =

   -0.0776
    3.5869
   11.0990
   -5.8499

>> [lambda, x] = potencias_inversas_aitken(H, [1 1 1 1]', vc(4))

lambda =

    3.4142

x =

   -0.0823
    1.0000
    0.1368
```

```
    0.8738

>> H*x

ans =

   -0.2809
    3.4142
    0.4671
    2.9833

>> lambda*x

ans =

   -0.2809
    3.4142
    0.4671
    2.9833
```

Problemas de contorno para ecuaciones diferenciales

1. El método de diferencias finitas.
2. Métodos variacionales.
   Algoritmo de Rayleigh-Ritz segmentario lineal
   Algoritmo de Rayleigh-Ritz con trazador cúbico-B

1. Diferencias finitas.

Los métodos de diferencias finitas tienen mejores características de estabilidad que los métodos de disparo, pero requieren generalmente mayor trabajo para alcanzar una precisión específica.

Los métodos que involucran diferencias finitas para resolver problemas de valor de frontera consisten en reemplazar cada una de las derivadas en la ecuación diferenial por una aproximación diferencia-cociente elegida de tal manera que mantenga cierto orden de error de truncamiento.

El problema lineal de segundo orden de valor de frontera:

$$y = p(x)y + q(x)y + r(x) \qquad a \le x \le b \qquad y(a) = \alpha \qquad y(b) = \beta$$

requiere que se usen aproximaciones diferencia-cociente para aproximar y y y . Para hacer esto, seleccionamos un número n > 0 y dividimos el intervalo [a, b] en (n+1) subintervalos iguales, cuyos puntos extremos son los puntos de red $x_i = a + i \cdot h$ para i = 0, 1, ..., n+1 donde

$$h = \frac{b-a}{n+1}$$

Escogiendo la constante h de esta manera facilitará la aplicación de un algoritmo para resolver sistemas lineales con una matriz de $n \times n$.
En los puntos interiores de la red $x_i, i = 1, 2, \ldots, n$, la ecuación diferencial a aproximar es:

$$y(x_i) = p(x_i)y(x_i) + q(x_i)y(x_i) + r(x_i)$$

Expandiendo y en el polinomio de Taylor de tercer grado alrededor de x_i y evaluando en x_{i+1} y x_{i-1}:

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + h\,y(x_i) + \frac{h^2}{2}y(x_i) + \frac{h^3}{6}y(x_i) + \frac{h^4}{24}y^{(iv)}(\xi_i)$$

para alguna $x_i < \xi < x_{i+1}$ y

$$y(x_{i-1}) = y(x_i - h) = y(x_i) - h\,y(x_i) + \frac{h^2}{2}y(x_i) - \frac{h^3}{6}y(x_i) + \frac{h^4}{24}y^{(iv)}(\xi_i)$$

para alguna $x_{i-1} < \xi < x_i$.

Sumando estas expresiones:

$$y(x_{i-1}) + y(x_{i+1}) = 2y(x_i) + h^2 y(x_i) + \frac{h^4}{12}y^{(iv)}(\xi_i)$$

para alguna $x_{i-1} < \xi < x_{i+1}$. Despejando:

$$\ddot{y}(x_i)=\frac{1}{h^2}\left[y(x_{i-1})-2y(x_i)+y(x_{i+1})\right]-\frac{h^4}{12}y^{(iv)}(\xi_i)$$

Esta ecuación se llama fórmula de diferencia centrada para $\ddot{y}(x_i)$.

De forma similar se obtiene una fórmula centrada para $\dot{y}(x_i)$. Expandiendo y en un polinomio de Taylor de segundo grado alrededor de x_i y evaluando en $x_{i+1}\, y\, x_{i-1}$:

$$y(x_{i+1})=y(x_i+h)=y(x_i)+h\,\dot{y}(x_i)+\frac{h^2}{2}\ddot{y}(x_i)+\frac{h^3}{6}\dddot{y}(\mu_i) \qquad para\ x_i<muy_i<x_{i+1}$$

$$y(x_{i-1})=y(x_i-h)=y(x_i)-h\,\dot{y}(x_i)+\frac{h^2}{2}\ddot{y}(x_i)-\frac{h^3}{6}\dddot{y}(\mu_i) \qquad para\ x_{i-1}<muy_i<x_i$$

$$y(x_{i+1})-y(x_{i-1})=2h\,\dot{y}(x_i)+\frac{k^3}{3}\dddot{y}(\mu_i)$$

Despejando :

$$\dot{y}(x_i)=\frac{1}{2h}\left[y(x_{i+1})-y(x_{i-1})\right]-\frac{h^2}{6}\dddot{y}(\mu_i) \qquad para\ x_{i-1}<my_i<x_{i+1}$$

El uso de estas fórmulas de diferencia centrada en la ecuación:

$$\ddot{y}=p(x)\dot{y}+q(x)y+r(x) \qquad a\le x\le b \qquad y(a)=\alpha \qquad y(b)=\beta$$

da lugar a la nueva ecuación:

$$\frac{y(x_{i+1})-2y(x_i)+y(x_{i-1})}{h^2}=p\frac{(x_i)\cdot y(x_{i+1})-y(x_i-1)}{2h}+q(x_i)\cdot y(x_i)+r(x_i)$$

con error de truncamiento de orden $O(h^2)$.

Si $w_i\approx y(x_i)$, entonces:

$$w_0\approx y(x_0)=\alpha\,,$$

$$w_{n+1}\approx y(x_{n+1})=\beta\,,$$

$$\frac{2w_i-w_{i+1}-w_{i-1}}{h^2}+p(x_i)\cdot\frac{w_{i+1}-w_{i-1}}{2h}+q(x_i)w_i=-r(x_i) \qquad para\ cada\ i=1,2,\ldots,n$$

Agrupando los términos w_{i-1}, w_i, w_{i+1}:

$$-(1+\frac{h}{2}p(x_i))w_{i-1}+(2+h^2q(x_i))w_i-(1-\frac{h}{2}p(x_i))w_{i+1}=-h^2r(x_i)$$

El sistema resultante de ecuaciones en forma matricial tridiagonal de $n\times n$, $A\,w=b$ es:

$$\begin{pmatrix} 2+h^2q(x_1) & -1+\frac{h}{2}p(x_1) & 0 & 0 & \cdots & 0 \\ -1-\frac{h}{2}p(x_2) & 2+h^2q(x_2) & -1+\frac{h}{2}p(x_2) & 0 & \cdots & 0 \\ 0 & -1-\frac{h}{2}p(x_3) & 2+h^2q(x_3) & -1+\frac{h}{2}p(x_3) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & -1-\frac{h}{2}p(x_{n-1}) & 2+h^2q(x_{n-1}) & -1+\frac{h}{2}p(x_{n-1}) \\ 0 & \cdots & 0 & 0 & -1-\frac{h}{2}p(x_n) & 2+h^2q(x_n) \end{pmatrix}$$

¿

$$w=(w_1, w_2, \ldots, w_{n-1}, w_n)^t$$

$$b=\left(-h^2r(x_1)+(1+\frac{h}{2}p(x_1))w_0, -h^2r(x_2), \ldots, -h^2r(x_{n-1}), -h^2r(x_n)+(1-\frac{h}{2}p(x_n)w_{n+1})\right)$$

```
% diffin.m
%
% function F=diffin(p, q, r, a, b, alpha, beta, N)
%
% Metodo de diferencias finitas. Construccion de una aproximacion a la
solucion del problema de contorno x''=p(t)x'(t) + q(t)x(t) + r(t),
% con x(a)=alfa y x(b)=beta en el intervalo [a, b], usando el metodo de
diferencias finitas de orden O(h^2).
% Observacion. Los nodos son a= t1 < ... < tN+1 = b y los puntos de la
solucion numerica son {(tj, xj)} para j=1...N+1
%
% Datos:
%   - p, q y r son las funciones coeficientes de la ecuacion almacenadas
como cadenas de caracteres 'p', 'q' y 'r'
%   - a y b son los extremos izquierdo y derecho del intervalo
%   - alfa=x(a) y beta=x(b) son los valores en la frontera
%   - N es el numero de pasos
%
% Resultado:
%   - F=[T' X'] siendo T' el vector de orden 1 x N de los nodos y X' el
vector 1 x N de los valores

function F=diffin(p, q, r, a, b, alfa, beta, N)
% Inicializacion de los vectores y de h
T =zeros(1, N+1);
X =zeros(1, N-1);
Va=zeros(1, N-2);
Vb=zeros(1, N-1);
Vc=zeros(1, N-2);
Vd=zeros(1, N-1);
h=(b-a)/N;

% Calculo del vector de los terminos independientes B en AX=B
Vt=a+h:h:a+h*(N-1);
Vb=-h^2*feval(r, Vt);
Vb(1)=Vb(1)+(1+h/2*feval(p, Vt(1)))*alfa;
Vb(N-1)=Vb(N-1)+(1-h/2*feval(p, Vt(N-1)))*beta;

% Calculo de la diagonal principal de A en AX=B
Vd=2+h^2*feval(q, Vt);

% Calculo de la superdiagonal de A en AX=B
Vta=Vt(1, 2:N-1);
Va=-1-h/2*feval(p,Vta);

% Calculo de la subdiagonal de A en AX=B
```

```
Vtc=Vt(1, 1:N-2);
Vc=-1+h/2*feval(p, Vtc);

% Resolucion de AX=B usando trisys
X=trisys(Va, Vd, Vc, Vb);
T=[a, Vt, b];
X=[alfa, X, beta];
F=[T' X'];
```

Ejemplo 1

Vamos a resolver el problema de contorno

$$\ddot{x}(t)=\frac{2t}{1+t^2}\dot{x}(t)-\frac{2}{1+t^2}x(t)+1$$

con x(0) = 1.25 y x(4) = -0.95 en el intervalo [0, 4].

Solución

Las funciones p, q y r son

$$p(t)=\frac{2t}{1+t^2}$$
$$q(t)=\frac{-2}{1+t^2}$$
$$r(t)=1$$

Creamos los archivos p.m, q.m y r.m:

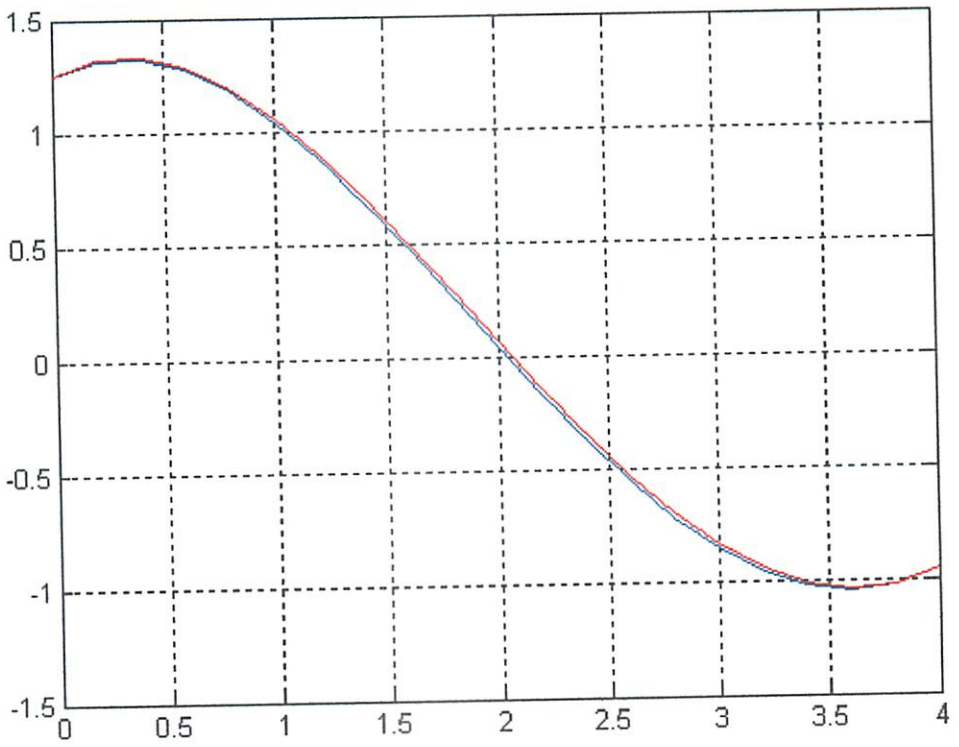
```
function y = p(x)
y = 2*t/(1+x^2);

function y=q(x)
y=-2/(1+x^2)

function r(x)
y = 1;

>> L=diffin('p', 'q', 'r', 0, 4, 1.25, -0.95, 4/0.2);
```

La siguiente tabla muestra la solución aproximada, segunda columna, y la compara con la exacta, tercera columna.

0	1.2500	1.2500	0
0.2000	1.3145	1.3174	0.0028
0.4000	1.3206	1.3265	0.0059
0.6000	1.2728	1.2818	0.0090
0.8000	1.1774	1.1894	0.0120
1.0000	1.0421	1.0569	0.0148
1.2000	0.8749	0.8921	0.0172
1.4000	0.6837	0.7029	0.0192
1.6000	0.4764	0.4972	0.0208
1.8000	0.2603	0.2822	0.0219
2.0000	0.0424	0.0649	0.0225
2.2000	-0.1706	-0.1480	0.0226
2.4000	-0.3726	-0.3503	0.0222
2.6000	-0.5576	-0.5363	0.0213
2.8000	-0.7201	-0.7003	0.0199
3.0000	-0.8550	-0.8371	0.0179
3.2000	-0.9572	-0.9419	0.0154

3.4000	-1.0222	-1.0099	0.0123
3.6000	-1.0455	-1.0367	0.0087
3.8000	-1.0227	-1.0181	0.0046
4.0000	-0.9500	-0.9500	0.0000

La gráfica muestra en azul la solución aproximada y en rojo la solución real.



Ejemplo 2

$$\ddot{y} = -\left(\frac{2}{x}\right)\dot{y} + \left(\frac{2}{x^2}\right)y + \frac{sen(\ln x)}{x^{2,}} \qquad x \in [1,2], \qquad y(1)=1, y(2)=2$$

La siguiente tabla compara la solución aproximada del problema de contorno anterior con su solución exacta y muestra el error cometido.

1.0000	1.0000	1.0000	0
1.0500	1.0454	1.0461	0.0007
1.1000	1.0916	1.0926	0.0010
1.1500	1.1385	1.1396	0.0011
1.2000	1.1861	1.1871	0.0010
1.2500	1.2341	1.2350	0.0009
1.3000	1.2827	1.2834	0.0007
1.3500	1.3317	1.3322	0.0005
1.4000	1.3811	1.3814	0.0003
1.4500	1.4310	1.4311	0.0001
1.5000	1.4812	1.4812	0.0000
1.5500	1.5317	1.5316	0.0001
1.6000	1.5826	1.5824	0.0002

```
    1.6500    1.6338    1.6335    0.0002
    1.7000    1.6852    1.6850    0.0002
    1.7500    1.7370    1.7368    0.0002
    1.8000    1.7891    1.7889    0.0002
    1.8500    1.8414    1.8413    0.0001
    1.9000    1.8940    1.8939    0.0001
    1.9500    1.9469    1.9468    0.0000
    2.0000    2.0000    2.0000         0
```

Ejemplo 3

$$\dot{x} = 2\dot{x} - x + t^2 - 1 \qquad en[0,1] \qquad con\, x(0) = 5 \qquad y \qquad x(1) = 10$$

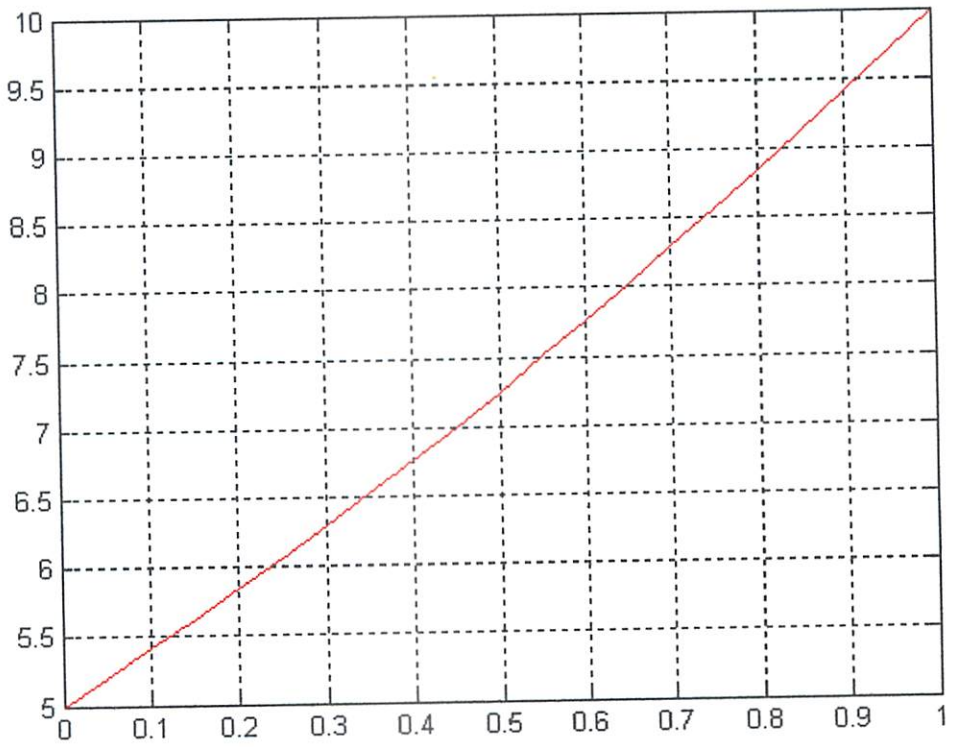
La solución es

$$x(t) = t^2 + 4t + 5$$

Creamos las funciones p, q y r

```
function y = p(x)
n=length(x);
y=zeros(n, 1);

% Copiamos la constante en todos los elementos del vector
for i=1:n
    y(i)=2;
end


function y=r(x)
y =x.^2-1

function y=q(x)
n=length(x);
y=zeros(n, 1);

% Copiamos la constante en todos los elementos del vector
for i=1:n
    y(i)=-1;
end

function y=f(x)
y = x.^2 + 4.*x + 5;

>> L=diffin('p', 'q', 'r', 0, 1, 5, 10, 1/0.05);
>> [L y abs(L(:, 2)-y)]

ans =

                  0   5.00000000000000   5.00000000000000                  0
   0.05000000000000   5.20250000000000   5.20250000000000   0.00000000000000
```

0.10000000000000	5.41000000000000	5.41000000000000	0.00000000000000
0.15000000000000	5.62250000000000	5.62250000000000	0.00000000000000
0.20000000000000	5.83999999999999	5.84000000000000	0.00000000000001
0.25000000000000	6.06249999999999	6.06250000000000	0.00000000000001
0.30000000000000	6.28999999999999	6.29000000000000	0.00000000000001
0.35000000000000	6.52249999999999	6.52250000000000	0.00000000000001
0.40000000000000	6.75999999999999	6.76000000000000	0.00000000000001
0.45000000000000	7.00249999999999	7.00250000000000	0.00000000000001
0.50000000000000	7.24999999999999	7.25000000000000	0.00000000000001
0.55000000000000	7.50249999999999	7.50250000000000	0.00000000000001
0.60000000000000	7.75999999999999	7.76000000000000	0.00000000000001
0.65000000000000	8.02249999999999	8.02250000000000	0.00000000000001
0.70000000000000	8.28999999999999	8.29000000000000	0.00000000000001
0.75000000000000	8.56249999999999	8.56250000000000	0.00000000000001
0.80000000000000	8.83999999999999	8.84000000000000	0.00000000000001
0.85000000000000	9.12249999999999	9.12250000000000	0.00000000000001
0.90000000000000	9.40999999999999	9.41000000000000	0.00000000000001
0.95000000000000	9.70249999999999	9.70250000000000	0.00000000000001
1.00000000000000	10.00000000000000	10.00000000000000	0



Ejemplo 4

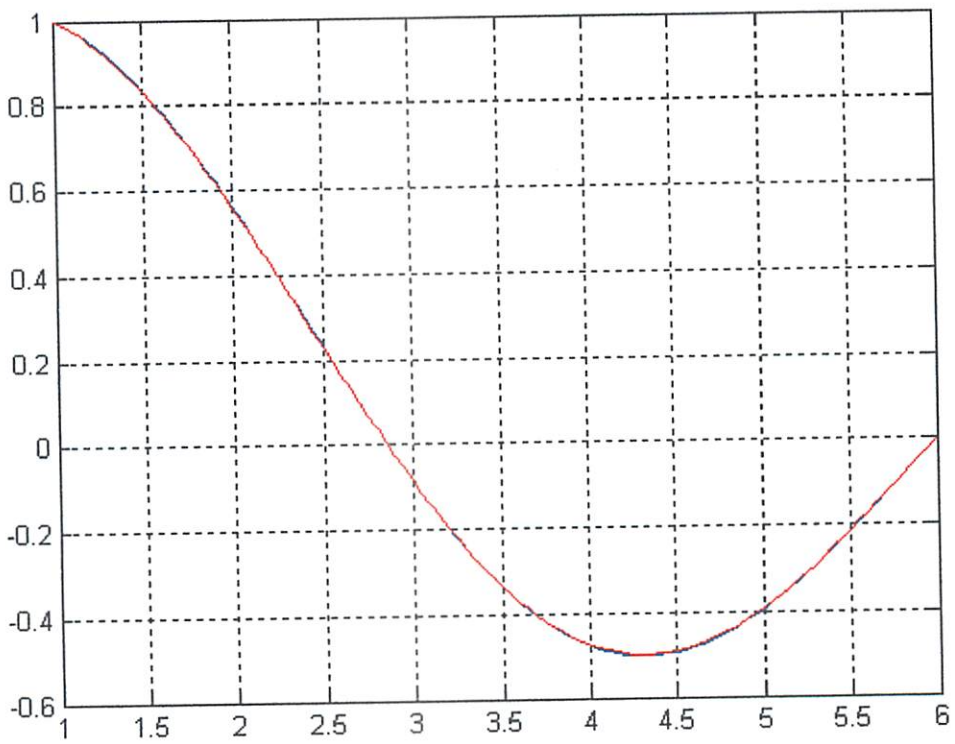
$$\ddot{x} + (1/t)\dot{x} + (1 - \frac{1}{4t^2})x = 0 \qquad t \in [1,6] \qquad con\ x(1) = 1 \quad y \quad x(6) = 0$$

La solución exacta es:

$$x(t)=\frac{0.2913843206\cos(t)+1.001299385\,\mathrm{sen}(t)}{\sqrt{(t)}}$$

Definimos las funciones p, q y r:

```
function y = p(x)
y=-1./x;

function y=q(x)
y=1./(1+4*x.^2) - 1;

function y=r(x)
y = zeros(length(x), 1);

>> L=diffin('p', 'q', 'r', 1, 6, 1, 0, 5/0.05);

function y=f(x)
y = (0.2913843206.*cos(x) + 1.001299385.*sin(x))./sqrt(x);

>> y=f(L(:, 1));
>> plot(L(:, 1), L(:, 2))
>> grid on
>> hold on
>> plot(L(:, 1), y)
```



2. Método de Rayleigh-Ritz.

Sea la ecuación

$$\frac{-d}{dx}\left(p(x)\cdot\frac{dy}{dx}\right)+q(x)y=f(x) \qquad para\, 0\le x\le 1,$$

conocida como ecuación de la viga, con condiciones de frontera y(0) = (1) = 0. Además, supondremos que $p\in C^1[0,1]\, y\, q, f\in C[0,1]$ y existe una constante $\delta>0$ tal que

$$p(x)\ge\delta>0 \qquad para\, 0\le x\le 1$$

y

$$q(x)\ge 0 \qquad para\, 0\le x\le 1$$

En estas condiciones, el problema de valor de frontera anterior tiene una solución única. Como en el caso de muchos problemas de valor de frontera que describen fenómenos físico, la solución a la ecuación de la viga satisface una propiedad variacional. El principio variacional para la ecuación de la viga es fundamental en el desarrollo del método de Rayleigh-Ritz y caracteriza a la solución de la ecuación de la viga como la función que minimiza cierta integral sobre todas las funciones en $C_0^2[0,1]$, el conjunto de aquellas funciones en $C^2[0,1]$ con la propiedad de que u(0) = u(1) = 0. El siguiente teorema da la caracterización.

Sean $p\in C^1[0,1], q, f\in C[0,1] y\, p(x)\ge\delta>0, q(x)\ge 0\ para\, 0\le x\le 1$

La función $y\in C_0^2[0,1]$ es la única solución de la ecuación diferencial:

$$\frac{-d}{dx}\left(p(x)\cdot\frac{dy}{dx}\right)+q(x)y=f(x) \qquad para\, 0\le x\le 1,$$

si y sólo si y es la única función en $C_0^2[0,1]$ que minimiza la integral:

$$I[u]=\int_0^1 \{p(x)[u(x)]^2+q(x)[u(x)]^2-2f(x)u(x)\}dx$$

En el procedimiento de Rayleigh-Ritz, la integral I se minimiza, no sobre todas las funciones en $C_0^2[0,1]$, sino sobre un conjunto más pequeño de funciones que consisten de combinaciones lineales de cierta base de funciones $\phi_1,\phi_2,\dots,\phi_n$. Las funciones de la base $\{\phi_i\}_{i=1}^n$ deben ser linealemente independientes y satisfacer

$$\phi_i(0)=\phi_i(1)=0 \qquad para\ cada\ i=1,2,\dots,n$$

Una aproximación $\phi(x)=\sum_{i=1}^n c_i\phi_i(x)$ a la solución y(x) de la ecuación de la viga se obtiene

entonces encontrando constantes $c_1, c_2, \ldots, c_n$ que minimicen $I[\sum_{i=1}^{n} c_i \phi_i]$

$$I[\phi] = I[\sum_{i=1}^{n} c_i \phi_i] = \int_0^1 \{p(x)[\sum_{i=1}^{n} c_i \phi_i]^2 + q(x)[\sum_{i=1}^{n} c_i \phi_i]^2 - 2f(x)[\sum_{i=1}^{n} c_i \phi_i]\} dx$$

para que ocurra un mínimo es necesario que todas las derivadas parciales se anulen,

$$\frac{\delta I}{\delta c_j} = 0 \qquad \textit{para cada } j = 1, 2, \ldots, n$$

Diferenciando $I[\phi]$:

$$\frac{\delta I}{\delta c_j} = \int_0^1 \{2p(x) \sum_{i=1}^{n} c_i \phi_i(x) \phi_j(x) + 2q(x) \sum_{i=1} n c_i \phi_i(x) \phi_j(x) - 2f(x) \phi_j(x)\} dx$$

de donde:

$$0 = \sum_{i=1}^{n} \left[\int_0^1 \{p(x) \phi_i(x) \phi_j(x) + q(x) \phi_i(x) \phi_j(x)\} dx \right] c_i - \int_0^1 f(x) \phi_j(x) dx$$

para cada j = 1, 2, ..., n. Las ecuaciones anteriores determinan un sistema **Ac = b,** donde la matriz A está dada por

$$a_{ij} = \int_0^1 \{p(x) \phi_i(x) \phi_j(x) + q(x) \phi_i(x) \phi_j(x)\} dx$$

y el vector b es

$$b_i = \int_0^1 f(x) \phi_i(x) dx$$

La primera base de funciones que consideraremos está formada por polinomios lineales segmentarios. El primer paso consiste en formar una partición de [0, 1] eescogiendo puntos $x_0, x_1, \ldots, x_{n+1}$ con

$$0 = x_0 < x_1 < \ldots < x_n < x_{n+1} = 1$$

Tomando $h_i = x_{i+1} - x_i$ para cada i = 0, 1, ..., n, definimos las funciones bases $\phi_1(x), \phi_2(x), \ldots, \phi_n(x)$ mediante

$$\phi_i(x)=\begin{cases} 0, & 0\le x\le x_{i-1},\\ \dfrac{x-x_{i-1}}{h_{i-1}}, & x_{i-1}<x\le x_i,\\ \dfrac{x_{i+1}-x}{h_i}, & x_i<x\le x_{i+1},\\ 0, & x_{i+1}<x\le 1 \end{cases} \qquad \textit{para cada } i=1,2,\dots,n$$

Como las funciones ϕ_i son lineales por pedazos, las derivadas ϕ_i , auqnue son continuas, son cosntantes en el subintervalo abierto (x_j, x_{j+1}) para cada j = 0, 1, ..., n. Por lo tanto,

$$\phi_i(x)=\begin{cases} 0, & 0<x<x_{i-1},\\ \dfrac{1}{h_{i-1}}, & x_{i-1}<x<x_i,\\ \dfrac{1}{h_i}, & x_i<x<x_{i+1},\\ 0, & x_{i+1}<x<1 \end{cases} \qquad \textit{para cada } i=1,2,\dots,n$$

Debido a que ϕ_i y ϕ_i son diferentes de cero sólo en (x_{i-1}, x_{i+1}) ,

$$\phi_i(x)\phi_j(x)\equiv 0 \qquad y \qquad \phi_i(x)\phi_j(x)\equiv 0$$

excepto cuando j es i-1, i, o i+1. En consecuencia, el sistema lineal se reduce a un sistema lineal tridiagona de $n\times n$.

$$a_{ii}=\int_{x_{i-1}}^{x_i}\left(\frac{1}{h_{i-1}}\right)^2 p(x)dx+\int_{x_i}^{x_{i+1}}\left(\frac{-1}{h_i}\right)^2 p(x)dx$$
$$+\int_{x_{i-1}}^{x_i}\left(\frac{1}{h_{i-1}}\right)^2 (x-x_{i-1})^2 q(x)dx+\int_{x_i}^{x_{i+1}}\left(\frac{1}{h_i}\right)^2 (x_{i+1}-x)^2 q(x)dx$$

para cada i = 1, 2, ..., n

$$a_{i,i+1}=-\int_{x_i}^{x_{i+1}}\left(\frac{1}{h_i}\right)^2 p(x)dx+\int_{x_i}^{x_{i+1}}\left(\frac{1}{h_i}\right)^2 (x_{i+1}-x)(x-x_i)q(x)dx$$

para cada i =1, 2, ..., n-1

$$a_{i,i-1}=-\int_{x_{i-1}}^{x_i}\left(\frac{1}{h_{i-1}}\right)^2 p(x)dx+\int_{x_{i-1}}^{x_i}\left(\frac{1}{h_i}\right)^2 (x_i-x)(x-x_{i-1})q(x)dx$$

para cada i = 2, 3, ..., n

$$b_i = \int_{x_{i-1}}^{x_i} \frac{1}{h_{i-1}}(x - x_{i-1}) f(x)\,dx + \int_{x_i}^{x_{i+1}} \frac{1}{h_i}(x_{i+1} - x) f(x)\,dx$$

para cada i = 1, ..., n.

Codificación en Matlab

```
% Algoritmo segmentario lineal de Rayleigh-Ritz
%
% Entrada:
%   xi, puntos de red
%   tol, tolerancia en las aproximaciones a las integrales
%
% Salida:
%   c, coeficientes

function [c] = RRL(x, tol)
n = length(x);

% Paso 1
for i=1:n-1
    h(i) = x(i+1) - x(i);
end

% Paso 2
% La base segmentaria está definida implícitamente

% Paso 3
i = 2;
IOLD1 = (1/h(i-1))^2 * romberg('p', x(i-1), x(i), tol);
IOLD2 = (1/h(i-1))^2 * romberg2('((x-a)^2)*q(x)', x(i-1), x(i), tol);

% Pasos 4 y 5
for i = 2:n-1
    INEW1 = (1/h(i))^2 * romberg('p', x(i), x(i+1), tol);
    INEW2 = (1/h(i-1))^2 * romberg2('((x-a)^2)*q(x)', x(i-1), x(i), tol);
    I3 = (1/h(i))^2 * romberg2('((b-x)^2)*q(x)', x(i), x(i+1), tol);
    I4 = (1/h(i))^2 * romberg2('(b-x)*(x-a)*q(x)', x(i), x(i+1), tol);
    I5 = (1/h(i-1)) * romberg2('(x-a)*f(x)', x(i-1), x(i), tol);
    I6 = (1/h(i))   * romberg2('(b-x)*f(x)', x(i), x(i+1), tol);

    alfa(i-1) = IOLD1 + INEW1 + IOLD2 + I3;
    beta(i-1) = -INEW1 + I4;
    b(i-1) = I5 + I6;
    IOLD1 = INEW1;
    IOLD2 = INEW2;
end

% Paso 6
a(1) = alfa(1);
epsilon(1) = beta(1)/alfa(1);

% Paso 7
for i = 2:n-3
    a(i) = alfa(i)-beta(i-1)*epsilon(i-1);
    epsilon(i) = beta(i) / a(i);
end

% Paso 8
n=i+1;
a(n) = alfa(n)-beta(n-1)*epsilon(n-1);

% Paso 9
z(1) = b(1) / a(1);

% Paso 10
```

```
for i = 2:n
    z(i) = (b(i) - beta(i-1)*z(i-1)) / a(i);
end

% Paso 11
c(n) = z(n);

% Paso 12
for i = n-1:-1:1
    c(i) = z(i) - epsilon(i)*c(i+1);
end
```

Las componentes en **c** son los coeficientes desconocidos $c_1, c_2, \ldots, c_n$ de los cuales se construye la arpoximación de Rayleigh-Ritz ϕ dada por

$$\phi(x) = \sum_{i=1}^{n} c_i \phi_i(x)$$

```
% Calcula la aproximación segmentaria lineal en el punto x
%
% Entradas:
%   c, vector con los coeficientes de la aproximación obtenidos mediante la
función RRL
%   x, punto donde calcular la aproximación a la solución
%   h, diferencia entre dos nodos
function y = aproxseglin(c, x, n)
for i = 1:length(x)
    y(i) = sigma(c, x(i), n);
end

function y=sigma(c, x, n)
y = 0;
for i = 0:n-1
    y = y + c(i+1)*phi(i, x, n);
end


% Define la base segmentaria lineal
function y = phi(i, x, n)
h = 1 / (n+1);
if x <= (i-1)*h                                  % (i-1)*h es x(i-1)
    y = 0;
elseif x <= i*h                                  % i*h es x(i)
    y = (x - (i-1)*h) / (i*h - (i-1)*h);
elseif x <= (i+1)*h                              % (i+1)*h es x(i+1)
    y = ((i+1)*h - x) / ((i+1)*h - i*h);
else
    y = 0;
end
```

Ejemplo 5

Considere el problema de valor de frontera

$$-y + \pi^2 y = 2\pi^2 \ sen(\pi x), \qquad 0 \le x \le 1, \ y(0) = y(1) = 0.$$

Solución

Definamos las funciones

$$p(x)=1$$
$$q(x)=\pi^2 y$$
$$f(x)=2\pi^2 sen(\pi x)$$

mediante los siguientes archivos. Archivo p.m:

```
function y = p(x)
y = 1;
```

Archivo q.m:

```
function y = q(x)
y=pi^2;
```

Archivo f.m:

```
function y = f(x)
y = 2 * (pi^2) * sin(pi * x);
```

Ahora, podemos utilizar el método de Rayleigh-Ritz para resolver el problema de valor de frontera del enunciado.

```
>> x = 0:.1:1;
>> c = RRL(x, 10^-6)
```

El vector c contiene los coeficientes con los cuales se construye la aproximación de Rayleigh-Ritz:

$(x) = c_i (x)_i$ para i = 1, 2, ..., n

$c_9 = 0.31028667561447$
$c_8 = 0.59020032952541$
$c_7 = 0.81234106301491$
$c_6 = 0.95496419334351$
$c_5 = 1.00410877480087$
$c_4 = 0.95496419334351$
$c_3 = 0.81234106301491$
$c_2 = 0.59020032952541$
$c_1 = 0.31028667561447$

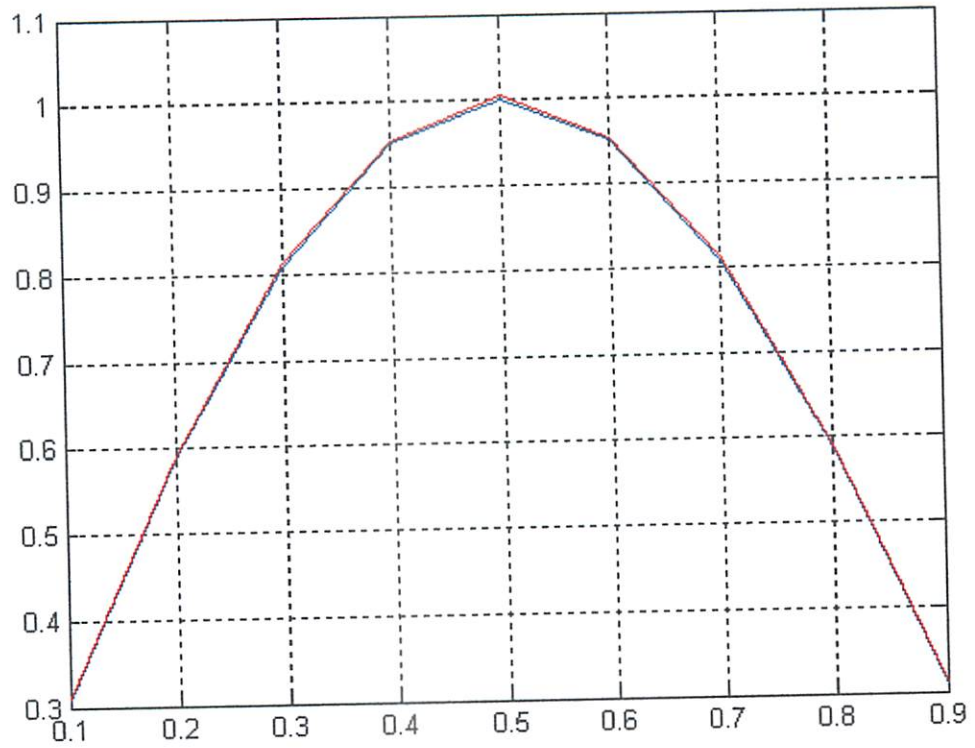
La solución real del problema de valor de frontera es:

$y(x) = sen(x)$

La siguiente tabla muestra la solución real, la aproximación y el error cometido en x_i:

```
>> for i=2:length(x)-1, phi(i-1)=sin(pi*x(i)); end
>> y=aproxseglin(c, x, .1);
>> [x(2:10)' phi' y' abs(phi-y)']

 0.10000000000000   0.30901699437495   0.31028667561447   0.00126968123953
 0.20000000000000   0.58778525229247   0.59020032952541   0.00241507723294
 0.30000000000000   0.80901699437495   0.81234106301491   0.00332406863996
 0.40000000000000   0.95105651629515   0.95496419334351   0.00390767704835
 0.50000000000000   1.00000000000000   1.00410877480087   0.00410877480087
 0.60000000000000   0.95105651629515   0.95496419334351   0.00390767704835
 0.70000000000000   0.80901699437495   0.81234106301491   0.00332406863996
```

```
0.80000000000000   0.58778525229247   0.59020032952541   0.00241507723294
0.90000000000000   0.30901699437495   0.31028667561447   0.00126968123953

>> plot(x(2:10), phi)
>> hold on
>> grid on
>> plot(x(2:10), y, 'r')
```



El uso de una base de funciones segmentarias lineales resulta en una solución aproximada continua pero no diferenciable en [0, 1]. Para construir una aproximación que pertenezca a $C_0^2[0,1]$ se necesita un conjunto más complicdo de funciones base. Las funciones que usaremosson los trazadores-B o trazadores de forma de campana.

$$S(x)=\begin{cases} 0, & x\le -2; \\ \frac{1}{4}[(2-x)^3-4(1-x)^3-6x^3+4(1+x)^3], & -2<x\le -1; \\ \frac{1}{4}[(2-x)^3-4(1-x)^3-6x^3], & -1<x\le 0; \\ \frac{1}{4}[(2-x)^3-4(1-x)^3], & 0<x\le 1; \\ \frac{1}{4}(2-x)^3, & 1<x\le 2; \\ 0, & 2<x. \end{cases}$$

$$\dot{S}(x)=\begin{cases} 0, & x\le -2; \\ \frac{1}{4}[-3(2-x)^2+12(1-x)^2-18x^2+12(1+x)^2], & -2<x\le -1; \\ \frac{1}{4}[-3(2-x)^2+12(1-x)^2-18x^2], & -1<x\le 0; \\ \frac{1}{4}[-3(2-x)^2-12(1-x)^2], & 0<x\le 1; \\ \frac{1}{4}[-3(2-x)^2], & 1<x\le 2; \\ 0, & 2<x. \end{cases}$$

Para construir las funciones base $\phi_i \in C_0{}^2[0,1]$ dividimos el intervalo [0, 1] escogiendo un entero positivo n y definiendo h = 1 / (n+1). Esto produce los nodos igualmente espaciados $x_i = ih$ para cada i = 0, 1, ..., n+1. Definimos entonces $S_i(x) = S((x - x_i)/h)$ para cada i = 0, 1, ..., n+1.

Es fácil demostrar que $\{S_i\}_{i=0}^{n+1}$ es un conjunto linealmente independiente de trazadores cúbicos. Para que el conjunto $\{S_i\}_{i=0}^{n+1}$ satisfaga las condiciones de frontera $\phi_i(0) = \phi_i(1) = 0$, es necesario modificar $S_0, S_1, S_n \, y \, S_{n+1}$ La base con esta modificación se define como

$$\phi_i(x)=\begin{cases} S_0(x)-4S(\frac{x+h}{h}), & i=0, \\ S_1(x)-S(\frac{x+h}{h}), & i=1, \\ S_i(x) & 2\le i\le n-1, \\ S_n(x)-S(\frac{x-(n+2)h}{h}), & i=n, \\ S_{n+1}(x)-4S(\frac{x-(n+2)h}{h}), & i=n+1. \end{cases}$$

$$\dot{\phi}_i(x)=\begin{cases} \frac{1}{h}\dot{S}(\frac{x}{h})-\frac{4}{h}\dot{S}(\frac{x+h}{h}), & i=0, \\ \frac{1}{h}\dot{S}(\frac{x-x_1}{h})-\frac{1}{h}\dot{S}(\frac{x+h}{h}), & i=1, \\ \frac{1}{h}\dot{S}(\frac{x-x_i}{h}) & 2\le i\le n-1, \\ \frac{1}{h}\dot{S}(\frac{x-x_n}{h})-\frac{1}{h}\dot{S}(\frac{x-(n+2)h}{h}), & i=n, \\ \frac{1}{h}\dot{S}(\frac{x-x_{n+1}}{h})-\frac{4}{h}\dot{S}(\frac{x-(n+2)h}{h}), & i=n+1. \end{cases}$$

Como $\phi_i(x)$ y $\phi_i(x)$ son diferentes de cero sólo para $x_{i-2} \le x \le x_{i+2}$, la matriz en la aproximación de Rayleigh-Ritz será una matrix banda con ancho de banda de a lo más siete, donde

$$a_{ij} = \int_0^1 \{p(x)\phi_i(x)\phi_j(x) + q(x)\phi_i(x)\phi_j(x)\}dx$$

```
function [A, b]=RRTC2(n, tol)
h = 1 / (n+1);
for i = 0:n+1
    for j = i:min([i+3, n+1]);
        L = max([(j-2)*h, 0]);
        U = min([(i+2)*h, 1]);
        A(i+1, j+1) = romberg(@integ, L, U, tol, n, i, j);
        A(j+1, i+1) = A(i+1, j+1);
    end
    if i >= 4
        for j = 0:i-4
            A(i+1, j+1) = 0;
        end
    end
    if i <= n-3
        for j=i+4:n+1
            A(i+1, j+1) = 0;
        end
    end
    L = max([(i-2)*h, 0]);
    U = min([(i+2)*h, 1]);
    b(i+1) = romberg(@integ2, L, U, tol, n, i, j);
end

function[quad, R, err, h]=romberg(f, a, b, tol, n, i, j)
M=1;
h=b-a;
err=1;
J=0;
R=zeros(4,4);
R(1,1)=h*(feval(f, a, n, i, j) + feval(f, b, n, i, j))/2;
while(err>tol)|(J<4)
   J=J+1;
   h=h/2;
   s=0;
   for p=1:M
      x=a+h*(2*p-1);
      s=s+feval(f, x, n, i, j);
   end
   R(J+1,1)=R(J,1)/2+h*s;
   M=2*M;
   for K=1:J
      R(J+1,K+1)=R(J+1,K)+(R(J+1,K)-R(J,K))/(4^K-1);
   end
   err=abs(R(J,J)-R(J+1,K+1));
end
quad=R(J+1,J+1);

function y=integ(x, n, i, j)
y=p(x)*dPhi(i, x, n)*dPhi(j, x, n) + q(x)*phi(i, x, n)*phi(j, x, n);

function y=integ2(x, n, i, j)
y=f(x)*phi(i, x, n);

function y=phi(i, x, n)
h=1/(n+1);
if i == 0
    y = S(x/h) - 4*S((x+h)/h);
elseif i == 1
    y = S((x-h)/h) - S((x+h)/h);
elseif i >= 2 & i <= n-1;
    y = S((x-i*h)/h);
```

```
elseif i == n
    y = S((x-n*h)/h) - S((x-(n+2)*h)/h);
elseif i == n+1
    y = S((x-(n+1)*h)/h) - 4*S((x-(n+2)*h)/h);
end

function y=dPhi(i, x, n)
h=1/(n+1);
if i == 0
    y = (1/h)*dS(x/h) - (4/h)*dS((x+h)/h);
elseif i == 1
    y = (1/h)*dS((x-h)/h) - (1/h)*dS((x+h)/h);
elseif i >= 2 & i <= n-1;
    y = (1/h)*dS((x-i*h)/h);
elseif i == n
    y = (1/h)*dS((x-n*h)/h) - (1/h)*dS((x-(n+2)*h)/h);
elseif i == n+1
    y = (1/h)*dS((x-(n+1)*h)/h) - (4/h)*dS((x-(n+2)*h)/h);
end

function y=S(x)
y=0;
if x > -2 & x <= -1
    y=(1/4)*((2-x)^3 - 4*(1-x)^3 - 6*x^3 + 4*(1+x)^3);
elseif x > -1 & x <= 0
    y=(1/4)*((2-x)^3 - 4*(1-x)^3 - 6*x^3);
elseif x > 0 & x <= 1
    y=(1/4)*((2-x)^3 - 4*(1-x)^3);
elseif x > 1 & x <= 2
    y=(1/4)*(2-x)^3;
end

function y=dS(x)
y=0;
if x > -2 & x <= -1
    y=(1/4)*(-3*(2-x)^2 + 12*(1-x)^2 - 18*x^2 + 12*(1+x)^2);
elseif x > -1 & x <= 0
    y=(1/4)*(-3*(2-x)^2 + 12*(1-x)^2 - 18*x^2);
elseif x > 0 & x <= 1
    y=(1/4)*(-3*(2-x)^2 + 12*(1-x)^2);
elseif x > 1 & x <= 2
    y=(1/4)*(-3*(2-x)^2);
end
```

Ejemplo 5 (continuación)

$$-y + \pi^2 y = 2\pi^2 \ sen(\pi x), \qquad 0 \le x \le 1, \ y(0) = y(1) = 0.$$

La solución del problema aplicando trazadores-B es la siguiente:

```
>> [A,b]=RRTC2(9, 10^-6);
>> c=A\b';
```

$c_0 = 0.00000503019421$
$c_1 = 0.20942888903506$
$c_2 = 0.39836220879536$
$c_3 = 0.54829692979962$
$c_4 = 0.64456235449690$
$c_5 = 0.67773262095036$
$c_6 = 0.64456235449690$
$c_7 = 0.54829692979961$
$c_8 = 0.39836220879536$
$c_9 = 0.20942888903506$
$c_{10} = 0.00000503019421$

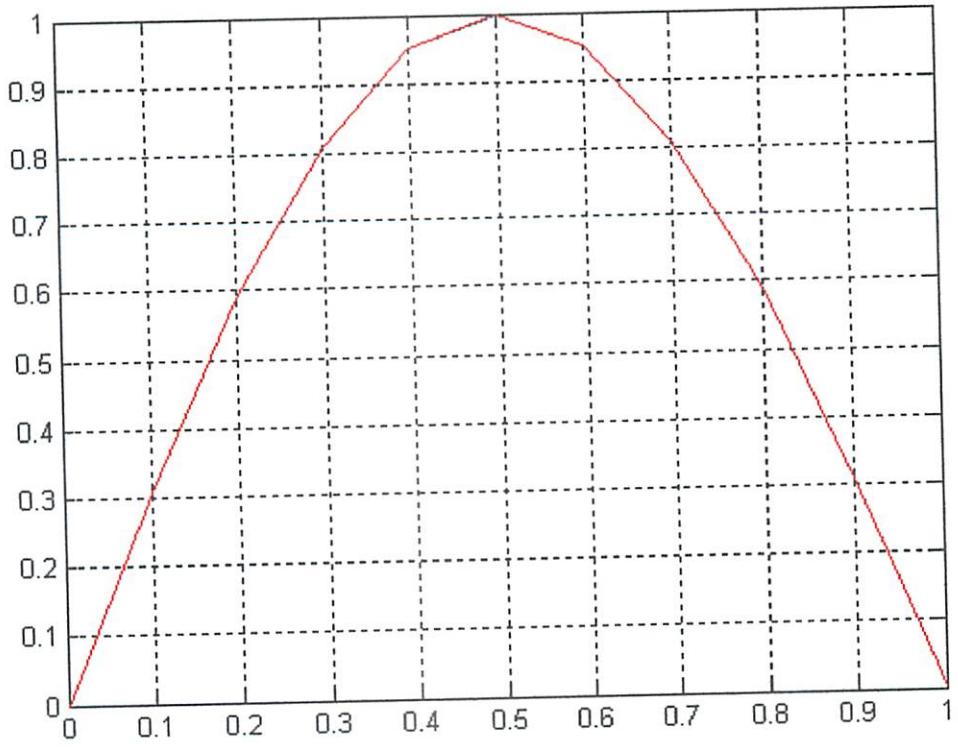
Para construir la aproximación empleamos la siguiente función

```
% Calcula la aproximación del trazado de campana en el punto x
%
% Entradas:
%   c, vector con los coeficientes de la aproximación obtenidos mediante la
función RRL
%   x, punto donde calcular la aproximación a la solución
%   h, diferencia entre dos nodos
function y = aproxbs(c, x, n)
for i = 1:length(x)
    y(i) = sigma(c, x(i), n);
end

function y=sigma(c, x, n)
y = 0;
for i = 0:n-1
    y = y + c(i+1)*phi(i, x, n);
end

% Define la base segmentaria de trazadores de campana
function y=phi(i, x, n)
h = 1/(n+1);
if i == 0
    y = S(x/h) - 4*S((x+h)/h);
elseif i == 1
    y = S((x-h)/h) - S((x+h)/h);
elseif i >= 2 & i <= n-1;
    y = S((x-i*h)/h);
elseif i == n
    y = S((x-n*h)/h) - S((x-(n+2)*h)/h);
elseif i == n+1
    y = S((x-(n+1)*h)/h) - 4*S((x-(n+2)*h)/h);
end

function y=S(x)
y=0;
if x > -2 & x <= -1
    y=(1/4)*((2-x)^3 - 4*(1-x)^3 - 6*x^3 + 4*(1+x)^3);
elseif x > -1 & x <= 0
    y=(1/4)*((2-x)^3 - 4*(1-x)^3 - 6*x^3);
elseif x > 0 & x <= 1
    y=(1/4)*((2-x)^3 - 4*(1-x)^3);
elseif x > 1 & x <= 2
    y=(1/4)*(2-x)^3;
end
```

```
>> for i=1:length(x), y(i)=sin(pi*x(i)); end
>> phi=aproxbs(c, x, .1);
>> [x' y' phi' abs(phi-y)']
```

0	0	0	0
0.1	0.30902069878246	0.30901699437495	0.00000370440751
0.2	0.58779366350403	0.58778525229247	0.00000841121156
0.3	0.80902807062268	0.80901699437495	0.00001107624774
0.4	0.95106974218439	0.95105651629515	0.00001322588924
0.5	1.00001379819881	1.00000000000000	0.00001379819881
0.6	0.95106974218439	0.95105651629515	0.00001322588924
0.7	0.80902807062268	0.80901699437495	0.00001107624773
0.8	0.58779366350403	0.58778525229247	0.00000841121156
0.9	0.30901944123390	0.30901699437495	0.00000244685896
1.0	0.00000000000000	0.00000000000000	0.00000000000000

```
>> plot(x, y)
>> hold on
>> grid on
>> plot(x, phi, 'r')
```


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Ecuaciones diferenciales parciales. Métodos basados en diferencias finitas

1. Ecuaciones en derivadas parciales parabólicas.
   Ecuación de difusión unidimensional:

   1. Método explícito.
   2. Método implícito.
   3. Método de Crank-Nicolson.

2. Ecuaciones en derivadas parciales hiperbólicas.
   Ecuación de onda unidimensional.

3. Ecuaciones en derivadas parciales elípticas.
   Ecuación de Laplace.

Se han desarrollado muchos algoritmos de computadora para resolver ecuaciones diferenciales parciales basados en diferencias finitas, elementos finitos, principios variacionales y métodos de proyección que incluyen el método de los momentos y, de manera más reciente, sus implantaciones basadas en ondeletas.

1. Ecuaciones en derivadas parciales parabólicas.

1.1. Método explícito para la ecuación de difusión unidimensional.

Sea la ecuación diferencial parcial parabólica

$$\frac{\partial u}{\partial t}(x,t)=c^2\frac{\partial^2 u}{\partial x^2}(x,t) \qquad 0<x<1, t>0$$

sujeta a las condiciones

$$u(0,t)=0 \qquad u(l,t)=0, \qquad t>0,$$

y

$$u(x,0)=f(x) \qquad 0\le x\le l$$

El enfoque que usaremos para aproximar la solución a este problema consiste en el uso de diferencias finitas. Primero seleccionamos dos constantes de malla h y k con la estipulación de que m = l/h es un entero. Los puntos de red para esta situación son

$$(x_i,t_i) \qquad donde\ x_i=ih\ y\ t_j=jk \qquad para\ i=0,1,\ldots,n, j=0,1,\ldots$$

Obtenemos el método de diferencia usando la serie de Taylor en t para formar el conciente diferencia:

$$\frac{\partial u}{\partial t}(x_i,t_j)=\frac{u(x_i,t_{j+k})-u(x_i,t_j)}{k}-\frac{k}{2}\frac{\partial^2}{\partial t^2}u(x_i,muy_j)$$

$$\frac{\partial^2 u}{\partial x^2}(x_i,t_j)=\frac{u(x_i+h,t_j)-2u(x_i,t_j)+u(x_i-h,t_j)}{h^2}-\frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i,t_j)$$

para algún $\mu_j\in(t_j,t_{j+1})y\xi_i\in(x_i-1,x_i+1)$.

La ecuación de difusión implica que en los puntos de red interiores

(x_i,t_j) i = 1, 2, ..., n-1 y j = 1, 2, ...

$$\frac{\partial u}{\partial t}(x_i,t_j)-c^2\frac{\partial^2 u}{\partial x^2}(x_i,t_j)$$

utilizando los cocientes diferenciales:

$$\frac{w_{i,j+1}-w_{i,j}}{k}-c^2\frac{w_{i+1,j}-2w_{i,j}+w_{i-1,j}}{h^2}=0$$

donde $w_{i,j}$ es la aproximación a $u(x_i,t_j)$. Resolviendo la ecuación para $w_{i,j+1}$:

$$w_{i,j+1}=k\left[\frac{w_{i,j}}{k}+c^2\frac{w_{i-1,j}-2w_{i,j}+2_{i+1,j}}{h^2}\right]$$

agrupando:

$$w_{i,j+1}=\left(1-\frac{2\alpha^2 k}{h^2}\right)+c^2\frac{k}{h^2}(w_{i+1,j}+w_{i-1,j})$$

para cada i = 1, 2, ..., n-1 y j = 1, 2, ... Como la condición inicial u(x, 0) = f(x), para cada $0\le x\le l$, implica que $w_{i,0}=f(x_i)$ para cada i = 1, ..., m estos valores pueden usarse para encontrar el valor de $w_{i,1}$ para cada i = 1, ..., n-1. Las condiciones adicionales $u(0,t)=0\,y\,u(l,t)=0$ implican que $w_{0,1}=w_{m,1}=0$; así que todas las componentes de la forma $w_{i,1}$ pueden determinarse. Si el procedimiento se aplica nuevamente, una vez que se conocen todas las aproximaciones $w_{i,1}$, los valores de $w_{i,2},w_{i,3},\cdots,w_{i,n-1}$ se pueden obtener de manera similar:

$$w^{(j)}=A\,w^{(j-1)}$$

donde

$$A=\begin{bmatrix}1-2\lambda & \lambda & 0 & \cdots & 0 & 0 & 0\\ \lambda & 1-2\lambda & \lambda & \cdots & 0 & 0 & 0\\ 0 & \lambda & 1-2\lambda & \cdots & 0 & 0 & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & \cdots & \lambda & 1-2\lambda & \lambda\\ 0 & 0 & 0 & \cdots & 0 & \lambda & 1-2\lambda\end{bmatrix}$$

y $\lambda=c^2\frac{k}{h^2}$. El método de diferencia progresiva será estable si $\rho(A)\le 1$ ó $\lambda\le 1/2$.

```
% Método de diferencia progresiva para la ecuación del calor
function [x, w]=forwdif(f, alfa, a, b, m, n)
h = a/m;
k = b/n;

lambda = alfa^2 * k / (h^2);

% Construcción de la primera fila de la matriz A
A(1, 1) = 1 - 2*lambda;
A(1, 2) = lambda;
% Construcción de las filas interiores de la matriz A
for i = 2:m-2
```

```
    A(i, i) = 1 - 2*lambda;
    A(i, i+1) = lambda;
    A(i, i-1) = lambda;
end
% Construcción de la última fila de la matriz A
A(m-1, m-2) = lambda;
A(m-1, m-1) = 1 - 2*lambda;

for i = 1:m-1
    x(i) = i*h;
    w(i) = feval(f, x(i));
end
w = w';
for j = 1:n
    w = A*w;
end
x=x';
```

Ejemplo 1

Considere la ecuación de calor

$$\frac{\partial u}{\partial t}(x,t)-\frac{\partial^2 u}{\partial x^2}(x,t)=0, \qquad 0<x<1, \qquad 0<t,$$

con condiciones de frontera

$$u(0,t)=u(1,t)=0, \qquad 0<t$$

y condiciones iniciales

$$u(x,0)=sen(\pi x), \qquad 0\le x\le 1.$$

Es fácil verificar que la solución a este problema es

$$u(x,t)=e^{-\pi^2 t}sen(\pi x)$$

Solución

Se aproximará la solución en t = 0.5 usando el método de diferencia progresiva primero con h = 0.1, k =0.0005 y λ = 0.05, y luego con h = 0.1, k = 0.01 y λ = 1.

Para emplear el método de diferencia progresiva es necesario definir el archivo f.m con las condiciones iniciales $u(x,0)=sen(\pi x)$

```
>> [x, w]=forwdif('f', 1, 1, .5, 1/0.1, .5/0.0005);
>> [x w]

ans =

    0.10000000000000   0.00228652078658
    0.20000000000000   0.00434922098744
    0.30000000000000   0.00598618913525
    0.40000000000000   0.00703718738226
    0.50000000000000   0.00739933669733
    0.60000000000000   0.00703718738226
    0.70000000000000   0.00598618913525
    0.80000000000000   0.00434922098744
    0.90000000000000   0.00228652078658
```

Este segundo caso muestra el problema de la estabilidad del método de diferencia progresiva.

```
>> [x, w]=forwdif('f', 1, 1, .5, 1/0.1, .5/0.01);
>> [x w]

ans =

  1.0e+006 *

   0.00000010000000   0.34540324073161
   0.00000020000000  -0.65593076045493
   0.00000030000000   0.90052692618896
   0.00000040000000  -1.05524767113756
   0.00000050000000   1.10560546497161
   0.00000060000000  -1.04773690954975
   0.00000070000000   0.88837425865892
   0.00000080000000  -0.64377809292543
   0.00000090000000   0.33789247914466
```

La solución exacta del problema es:

```
>> y=exp((-pi^2)*0.5) .* sin(pi.*x);
>> [x' y']

ans =

                   0                   0
   0.10000000000000   0.00222241417851
   0.20000000000000   0.00422728297276
   0.30000000000000   0.00581835585643
   0.40000000000000   0.00683988752999
   0.50000000000000   0.00719188335583
   0.60000000000000   0.00683988752999
   0.70000000000000   0.00581835585643
   0.80000000000000   0.00422728297276
   0.90000000000000   0.00222241417851
   1.00000000000000   0.00000000000000
```

1.2. Método implícito para la ecuación de difusión unidimensional.

Este método resulta de usar el cociente de diferencia regresiva para

$$\frac{\partial u}{\partial t}(x_i,t_j)=\frac{u(x_i,t_j)-u(x_i,t_{j-1})}{k}+\frac{k}{2}\frac{\partial^2}{\partial t^2}u(x_i,muy_j)$$

$\mu_j\in(t_{j-1},t_j)$.

Sustituyendo esta ecuación, junto con la aproximación diferencia-cociente para $\frac{\partial^2 u}{\partial x^2}(x_i,t_j)$ en la ecuación de calor se obtiene:

$$\frac{u(x_i,t_j)-u(x_i,t_{j-1})}{k}-c^2\frac{u(x_{i+1},t_j)-2u(x_i,t_j)+u(x_{i-1},t_j)}{h^2}=$$
$$=-\frac{k}{2}\frac{\partial^2 u}{\partial t^2}(x_i,\mu_j)-\frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i,t_j)$$

para $\mu_j\in(t_{j-1},t_{j+1})y\xi_i\in(x_{i-1},x_{i+1})$.

El método de diferencia regresiva se obtiene de desprecial el término de error:

$$\frac{w_{i,j}-w_{i,j-1}}{k}-c^2\frac{w_{i+1,j}-2w_{i,j}+w_{i-1,j}}{h^2}=0$$

donde $w_{i,j}\approx u(x_i,t_j)$. Resolviendo la ecuación para $w_{i,j-1}$:

$$w_{i,j-1}=-k\left(-\frac{w_{i,j}}{k}+c^2\frac{w_{i+1,j}-2w_{i,j}+w_{i-1,j}}{h^2}\right)$$
$$w_{i,j-1}=\left(1+\frac{2c^2k}{h^2}\right)w_{i,j}-k\frac{c^2}{h^2}(w_{i+1,j}+w_{i-1,j})$$

llamando $\lambda=\frac{kc^2}{h^2}$: $w_{i,j-1}=1+2\lambda w_{i,j}-\lambda(w_{i+1,j}+w_{i-1,j})$

para cada i = 1, 2, ..., m-1 y j = 1, 2, ...

$$w_{i,0}=f(x_i)\qquad i=1,2,\ldots,m-1$$
$$w_{m,j}=w_{o,j}\qquad j=1,2,\ldots$$

Matricialmente:

$$\begin{bmatrix} 1+2\lambda & -\lambda & 0 & \dots & 0 & 0 & 0 \\ -\lambda & 1+2\lambda & -\lambda & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -\lambda & 1+2\lambda & -\lambda \\ 0 & 0 & 0 & \dots & 0 & -\lambda & 1+2\lambda \end{bmatrix} \begin{bmatrix} w_{1,j} \\ w_{2,j} \\ \dots \\ w_{m-1,j} \end{bmatrix} = \begin{bmatrix} w_{1,j-1} \\ w_{2,j-1} \\ \dots \\ w_{m-1,j-1} \end{bmatrix}$$

$$A\,w^{(j)} = w^{(j-1)} \qquad j = 1, 2, \dots$$

$$w^{(j)} = A^{-1}\,w^{(j-1)}$$

```
% Método de diferencia regresiva para la ecuación del calor
function [x, w]=backdif(f, alfa, a, b, m, n)
h = a / m;
k = b / n;
lambda = alfa^2 * (k / h^2);

for i = 1:m-1
    x(i) = i*h;
    w(i) = feval(f, x(i));
end

% Resuelve un sistema lineal tridiagonal utilizando el algoritmo de
reducción de Crout
l(1) = 1 + 2*lambda;
u(1) = -lambda/l(1);

for i = 2:m-2
    l(i) = 1 + 2*lambda + lambda * u(i-1);
    u(i) = -lambda/l(i);
end
l(m-1) = 1 + 2*lambda + lambda*u(m-2);

for j = 1:n
    z(1) = w(1)/l(1);
    for i = 2:m-1
        z(i) = (w(i) + lambda*z(i-1)) / l(i);
    end
    w(m-1) = z(m-1);
    for i = m-2:-1:1
        w(i) = z(i) - u(i)*w(i+1);
    end
end
x=x'; w=w';
```

Ejemplo 1 (continuación)

Se usará el método de diferencia regresiva con h = 0.1 y k = 0.01 para aproximar la solución de la ecuación de calor.

```
>> [x, w]=backdif('f', 1, 1, .5, 1/0.1, .5/0.01);
>> [x w]

ans =

    0.10000000000000   0.00289801664505
    0.20000000000000   0.00551235522922
    0.30000000000000   0.00758710607671
    0.40000000000000   0.00891917811894
    0.50000000000000   0.00937817886332
```

```
0.60000000000000    0.00891917811894
0.70000000000000    0.00758710607671
0.80000000000000    0.00551235522922
0.90000000000000    0.00289801664505
```

1.3. Método de Crank-Nicolson.

Este método promedia el método de diferencia progresiva en el j-ésimo paso en t con el método de diferencia regresiva en el paraso (j+1) en t.

$$\frac{w_{i,j+1}-w\,i,j}{k}-c^2\frac{w_{i+1,j}-2w_{i,j}+w_{i-1,j}}{h^2}=0$$

$$\frac{w_{i,j+1}-w_{i,j}}{k}-c^2\frac{w_{i+1,j+1}-2w_{i,j+1}+w_{i-1,j+1}}{h^2}=0$$

$$\frac{w_{i,j+1}-w_{i,j}}{k}-\frac{c^2}{2}\left[\frac{w_{i+1,j}-2w_{i,j}+w_{i-1,j}}{h^2}+\frac{w_{i,j+1}-2w_{i,j+1}+w_{i-1,j+1}}{h^2}\right]=0$$

El error de truncamiento es de orden $O(k^2+h^2)$.

Matricialmente, $A\,w^{(j+1)}=B\,w^j$

$$A=\begin{bmatrix} 1+\lambda & \frac{-\lambda}{2} & 0 & \dots & 0 & 0 & 0 \\ \frac{-\lambda}{2} & 1+\lambda & \frac{-\lambda}{2} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \frac{-\lambda}{2} & 1+\lambda & \frac{-\lambda}{2} \\ 0 & 0 & 0 & \dots & 0 & \frac{-\lambda}{2} & 1+\lambda \end{bmatrix}$$

$$B=\begin{bmatrix} 1-\lambda & \frac{\lambda}{2} & 0 & \dots & 0 & 0 & 0 \\ \frac{\lambda}{2} & 1-\lambda & \frac{\lambda}{2} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \frac{\lambda}{2} & 1-\lambda & \frac{\lambda}{2} \\ 0 & 0 & 0 & \dots & 0 & \frac{\lambda}{2} & 1-\lambda \end{bmatrix}$$

donde $\lambda=c^2\frac{k}{h^2}$

$$w^{(j+1)}=A^{(-1)}B\,w^{(j)}$$

```
% Método de Crank-Nicolson para resolver la ecuación del calor
function [x, w]=CrankNicolson(f, alfa, a, b, m, n)%, tol, mit)
h = a / m;
k = b / n;
lambda = alfa^2 * (k / h^2);
w(m-1) = 0;

for i = 1:m
    x(i) = i*h;
    w(i) = feval(f, x(i));
end

% Resuelve un sistema lineal tridiagonal utilizando el algoritmo de
reducción de Crout
l(1) = 1 + lambda;
u(1) = -lambda/(2*l(1));

for i = 2:m-2
    l(i) = 1 + lambda + lambda*u(i-1)/2;
    u(i) = -lambda/(2*l(i));
end
l(m-1) = 1 + lambda + lambda*u(m-2)/2;

for j = 1:n
    z(1) = ((1-lambda)*w(1) + lambda/2*w(2))/l(1);
    for i = 2:m-1
        z(i) = ((1-lambda)*w(i) + (lambda/2)*(w(i+1)+w(i-1)+z(i-1))) /
l(i);
    end
    w(m-1) = z(m-1);
    for i = m-2:-1:1
        w(i) = z(i) - u(i)*w(i+1);
    end
end
x=x'; w=w';
```

Ejemplo 1 (continuación)

El método de Crank-Nicolson puede usarse para aproximar la solución al problema

```
>> [x, w]=CrankNicolson('f', 1, 1, .5, 1/0.1, .5/0.01);
>> [x w]

ans =

   0.10000000000000   0.00230512336779
   0.20000000000000   0.00438460519960
   0.30000000000000   0.00603489132513
   0.40000000000000   0.00709444024020
   0.50000000000000   0.00745953591469
   0.60000000000000   0.00709444024020
   0.70000000000000   0.00603489132513
   0.80000000000000   0.00438460519960
   0.90000000000000   0.00230512336779
   1.00000000000000   0.00000000000000
```

Ejemplo 2

Vamos a usar el método de diferencias progresivas para resolver la ecuación del calor:

$$u_t(x,t) = u_{xx}(x,t) \qquad para\ 0 < x < 1 \qquad y \qquad 0 < t < 0.20,$$

con las condiciones iniciales

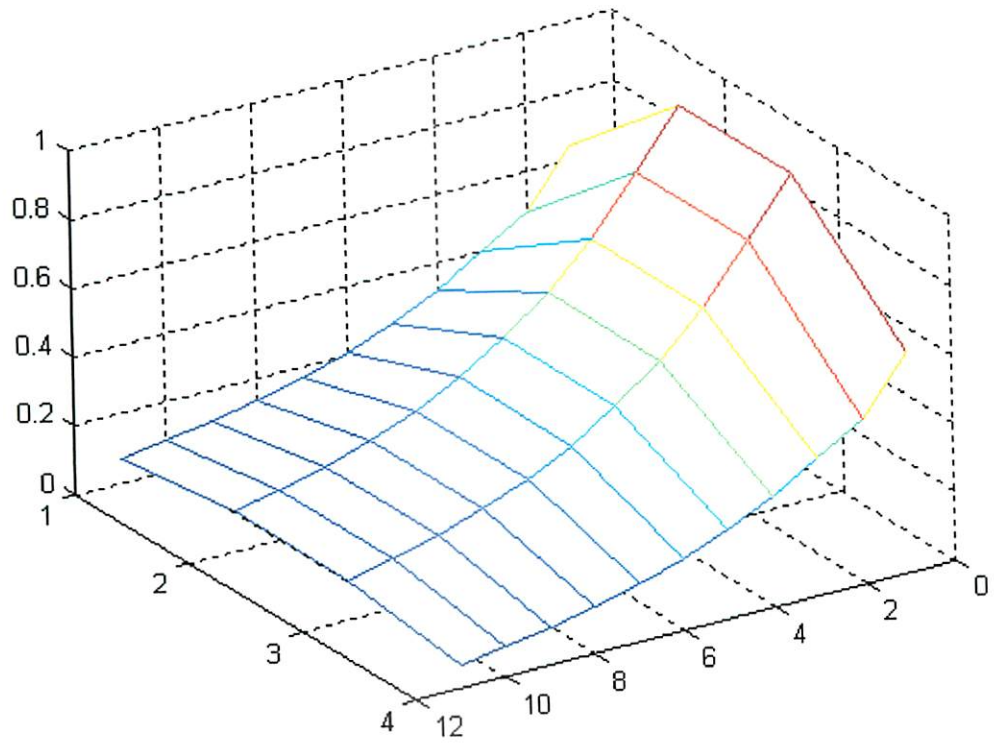
$$u(x,0) = f(x) = 4x - 4x^2 \qquad para\ t = 0 \qquad y \qquad 0 \le x \le 1$$

y las condiciones de contorno

$$u(0,t)=0, \qquad u(1,t)=0, \qquad x=0 \qquad y \qquad 0\le t\le 0.20$$

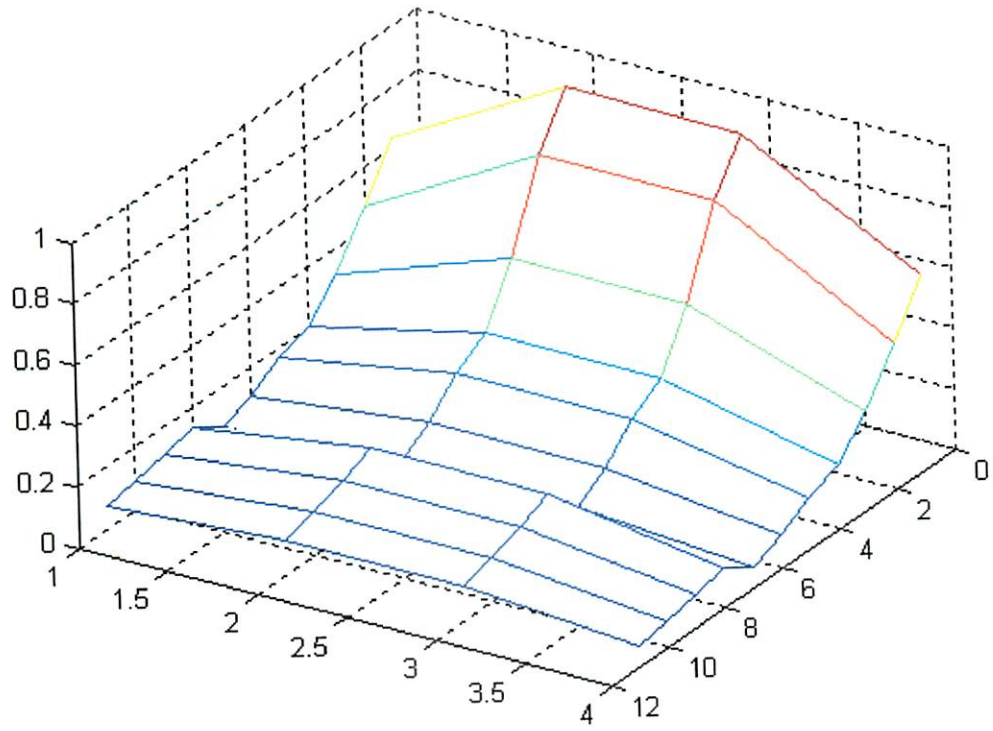
```
>> t=0:.02:.2;
>> for i = 1:length(t), [x(:,i), w(:,i)]=forwdif('f', 1, 1, t(i), 1/0.2,
t(i)/0.02); end
>> mesh(w)
```

$\lambda = 0.5$. La malla tiene n = 6 columnas de ancho y m = 11 filas de alto. La fórmula es estable para este valor de λ .



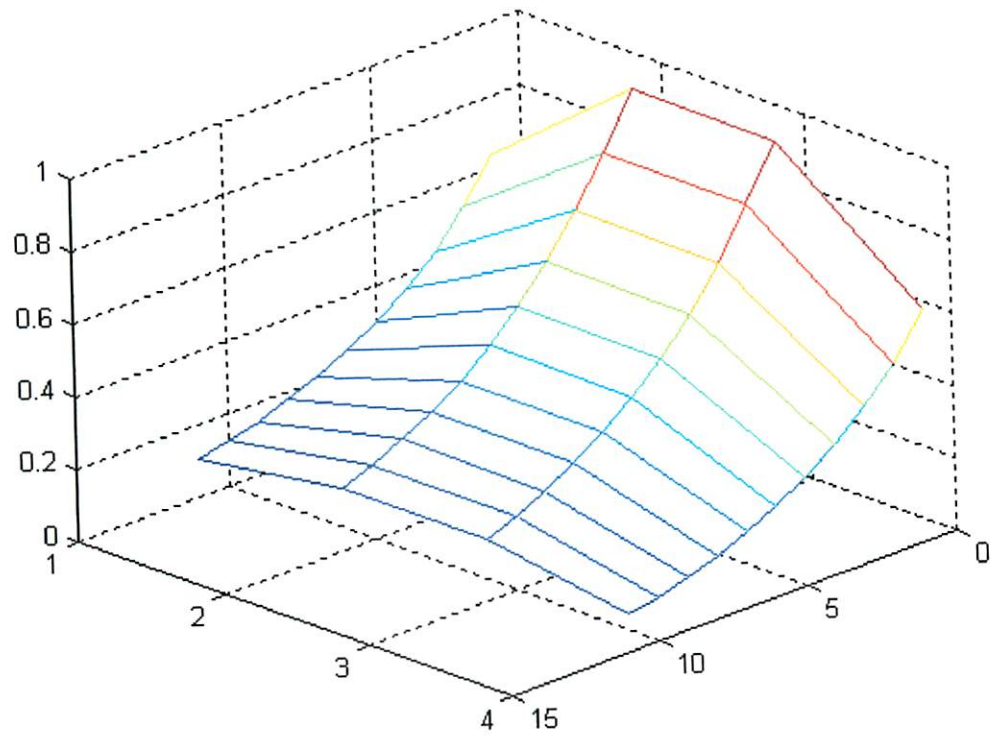
Ahora, tomamos como tamaño de paso para h = 0.2 y para k = 1/30. En este caso, $\lambda = 0.83333$ y el algoritmo se vuelve inestable porque $\lambda > ½$ y los errores introducidos en una fila se amplificarán en las filas posteriores.

```
>> t=0:1/30:.2;
>> for i = 1:length(t), [x(:,i), w(:,i)]=forwdif('f', 1, 1, t(i), 1/0.2,
t(i)/0.02); end
>> mesh(w)
```

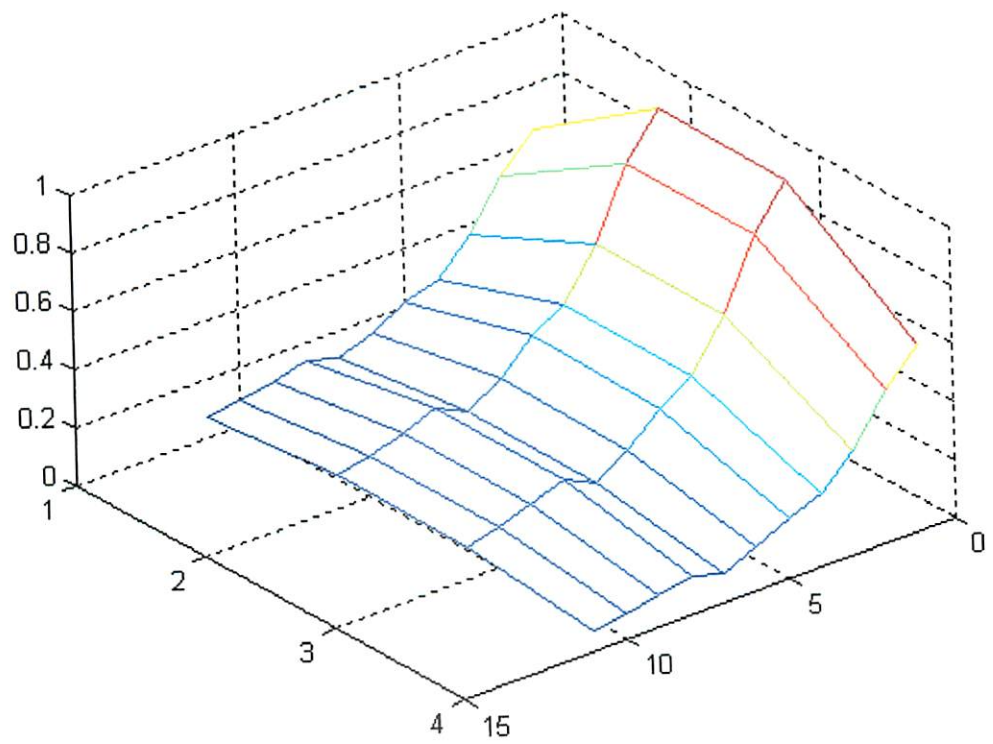


Resolución del problema con el método de diferencia regresiva

```
>> t=0:0.02:.2;
>> for i = 1:length(t), [x(:,i), w(:,i)]=backdif('f', 1, 1, t(i), 1/0.2,
t(i)/0.02); end
>> mesh(w)
```



```
>> t=0:1/30:.2;
>> for i = 1:length(t), [x(:,i), w(:,i)]=backdif('f', 1, 1, t(i), 1/0.2,
t(i)/0.02); end
>> mesh(w);
```



Ejemplo 3

Vamos a utilizar el método de Crank-Nicolson para resolver la ecuación

$$u_t(x,t) = u_{xx}(x,t) \qquad para\ 0 < x < 1 \qquad y \qquad 0 < t < 0.1$$

con las condiciones iniciales

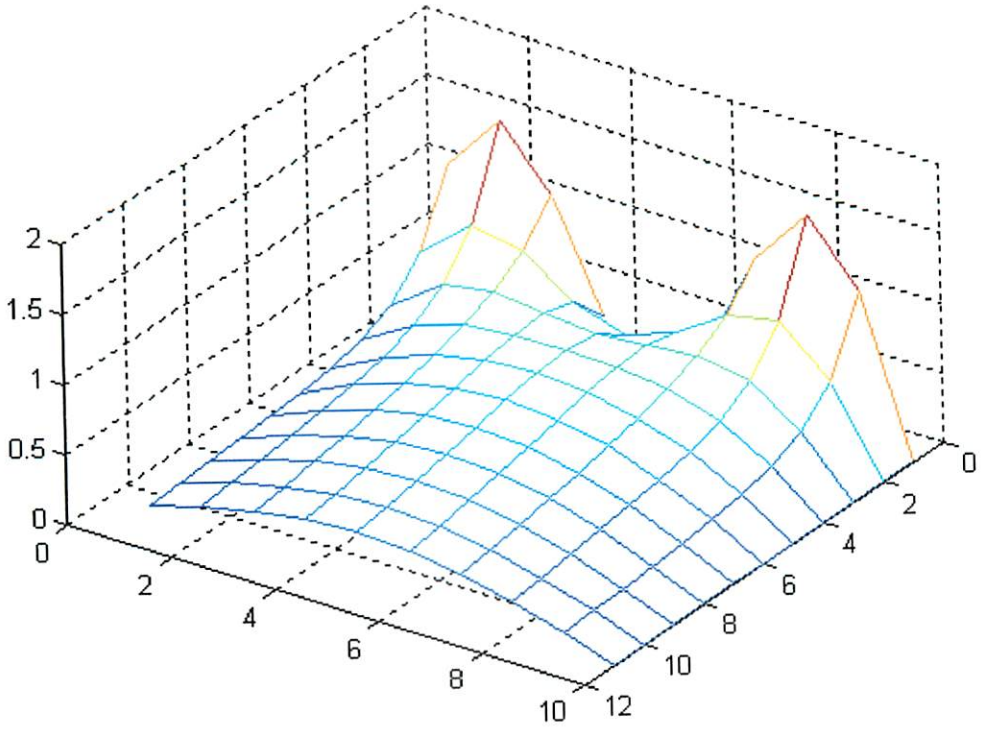
$$u(x,0) = f(x) = sen(\pi x) + sen(3\pi x) \qquad para\ t = 0 \qquad y \qquad 0 \le x \le 1.$$

y las condiciones de contorno

$$u(0,t) = u(1,t) = 0 \qquad para\ x = 0 \quad y \quad 0 \le t \le 0.1$$

Tomaremos como tamaño de paso h = 0.1, y k = 0.01 de manera que $\lambda = 1$.

```
>> t=0:0.01:.1;
>> for i = 1:length(t), [x(:,i), w(:,i)]=CrankNicolson('f', 1, 1, t(i),
1/0.1, t(i)/0.01); end
>> mesh(w)
```


2
1.5
1
0.5
0
0
2
4
6
8
10
12
0
2
4
6
8
10

2. Ecuaciónes en derivadas parciales hiperbólicas.

La ecuación de onda se define:

$$\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = 0 \qquad 0 < x < l, t > 0$$

donde c es una constante. Sujeta a las condiciones:

$$\begin{aligned} u(0,t) &= u(l,t) = 0 \quad t > 0 \\ u(x,0) &= f(x) \qquad 0 \le x \le l \\ \frac{\partial u}{\partial t}(x,0) &= g(x) \qquad 0 \le x \le l \end{aligned}$$

Seleccionamos un entero m > 0 tal que $h = \frac{l}{m}$ y un tamaño de paso de tiempo k > 0. Los puntos de red para el método de diferencia finita se definen como:

$$\begin{aligned} x_i &= ih \quad i = 0, 1, \ldots, m \\ t_j &= jk \quad j = 0, 1, \ldots \end{aligned}$$

La ecuación de onda en cualquier punto interior de la red es:

$$\frac{\partial^2 u}{\partial t^2}(x_i, t_j) - c^2 \frac{\partial^2 u}{\partial x^2}(x_i, t_j) = 0$$

Para obtener el método de diferencia para la ecuación de onda usamos el cociente de diferencia centrada para las segundas derivadas parciales:

$$\frac{\partial^2 u}{\partial t^2}(x_i, t_j) = \frac{u(x_i, t_{j+1}) - 2u(x_i, t_j) + u(x_i, t_{j-1})}{k^2} - \frac{k^2}{12} \frac{\partial^4 u}{\partial t^4}(x_i, muy_j)$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_j) = \frac{u(x_{i+1}, t_j) - 2u(x_i, t_j) + u(x_{i-1}, t_j)}{h^2} - \frac{h^2}{12} \frac{\partial^4 u}{\partial x^4}(\xi_i, t_j)$$

donde $t_{j-1} < muy_j < t_{j+1}$ y $x_{i-1} < \xi_i < x_{i+1}$.
Realizando la aproximación: $w_{i,j} \approx u(x_i, t_j)$, despreciando el error de truncamiento y sustituyendo en la ecuación de onda obtenemos la ecuación de diferencia:

$$\frac{w_{i,j+1} - 2w_{i,j} + w_{i,j-1}}{k^2} - c^2 \frac{w_{i+1,j} - 2w_{i,j} + w_{i-1,j}}{h^2} = 0$$

Definiendo $\lambda = c\frac{k}{h}$:

$$\frac{w_{i,j+1} - 2w_{i,j} + w_{i,j-1}}{k^2} - \lambda^2 \frac{w_{i+1,j} + 2\lambda w_{i,j} - \lambda^2 w_{i-1,j}}{h^2} = 0$$

Resolviendo para

$$w_{i,j+1} = 2(1-\lambda^2)w_{i,j} + \lambda^2(w_{i+1,j} + w_{i-1,j}) - w_{i,j-1} \quad para\, i = 1,2,\ldots,m-1 \quad y \quad j = 1,2,\ldots$$

Las condiciones de frontera dan:

$$w_{0,j} = w_{m,j} = 0 \quad j = 1,2,3,\ldots$$
$$w_{i,0} = f(x_i) \qquad i = 1,2,\ldots,m-1$$

los valores para j = 1 se obtienen de la condición de la velocidad inicial de propagación:

$$\frac{\partial u}{\partial t}(x,0) = g(x) \qquad 0 \le x \le l$$

En forma matricial: $w^{(j+1)} = A\,w^{(j)} - w^{(j-1)}$, donde

$$A = \begin{bmatrix} 2(1-\lambda^2) & \lambda^2 & 0 & \ldots & 0 & 0 & 0 \\ \lambda^2 & 2(1-\lambda^2) & \lambda^2 & \ldots & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & \lambda^2 & 2(1-\lambda^2) & \lambda^2 \\ 0 & 0 & 0 & \ldots & 0 & \lambda^2 & 2(1-\lambda^2) \end{bmatrix}$$

```
% Algoritmo de diferencia finita para la ecuación de onda
%
% Entradas
%   f, condiciones
%   alfa, constante
%   a, punto extremo de x
%   b, tiempo maximo
%   m, subdivisiones en x
%   n, subdivisiones en t

function [x, w]=onda(f, g, alfa, a, b, m, n)
h = a / m;
k = b / n;
lambda = alfa * (k / h);

for i = 0:m
    x(i+1) = i*h;
end

w(1, 1)   = feval(f, 0);
w(1, 2:n) = 0;
w(m+1, 1)   = feval(f, a);
w(m+1, 2:n) = 0;

% Inicialización para t=0 y t=k
for i = 0:m
    w(i+1, 1) = feval(f, i*h);
    w(i+1, 2) = (1-lambda^2)*feval(f, i*h) + (lambda^2/2)*(feval(f,
(i+1)*h)+feval(f, (i-1)*h)) + k*feval(g, i*h);
end

for j = 2:n
    for i = 2:m
        w(i, j+1) = 2*(1-lambda^2)*w(i,j) + (lambda^2)*(w(i+1, j) + w(i-1,
j)) - w(i, j-1);
    end
end
```

Ejemplo

Consideremos el problema hiperbólico

$$\frac{\partial^2 u}{\partial t^2}(x,t) - 4\frac{\partial^2 u}{\partial x^2}(x,t) = 0 \qquad 0 < x < 1, \quad 0 < t.$$

Con condiciones de frontera

$$u(0, t) = u(1, t) = 0, 0 < t,$$

y condiciones iniciales

$$u(x,0) = sen(\pi x), \qquad 0 \le x \le 1,$$

y

$$\frac{\partial u}{\partial t}(x,0) = 0, \qquad 0 \le x \le 1.$$

Es fácil verificar que la solución de este problema es

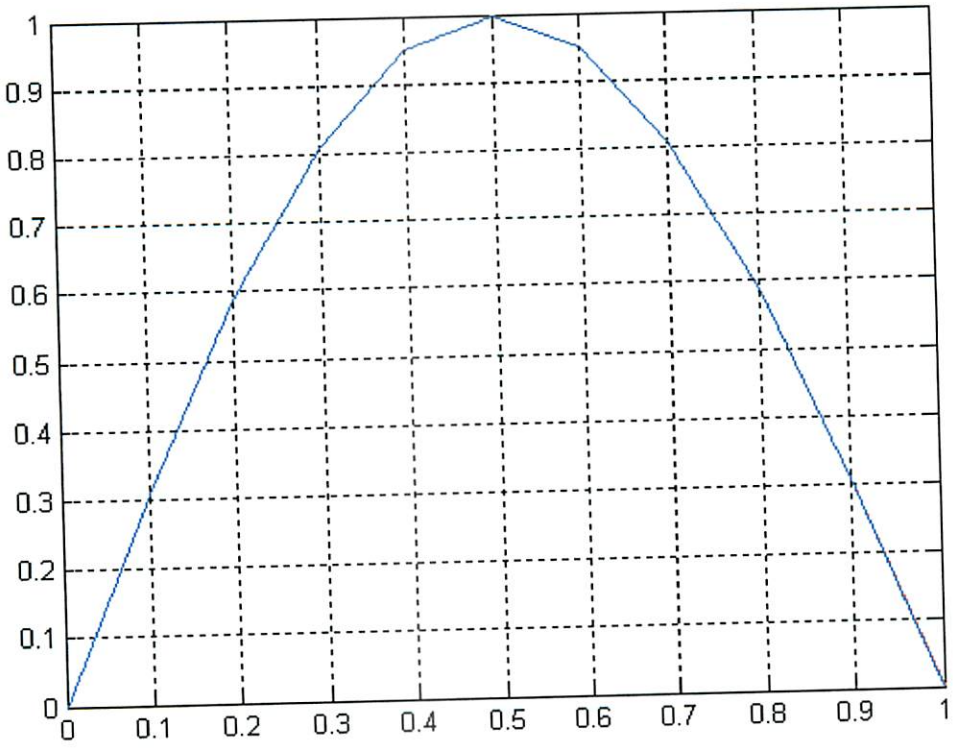
$$u(x,t) = sen(\pi x)\cos(2\pi t)$$

Definimos los archivos f.m y g.m:

```
function y=f(x)
y = sin(pi*x);

function y = g(x)
y = 0;

>> [x, w] = onda('f', 'g', 2, 1, 1, 10, 20);
>> [x' w(:,21) u(x, 20)' abs(w(:, 21)-u(x,20)')]

ans =

                  0                        0                        0                        0
   0.10000000000000   0.30901699437495   0.30901699437495   0.00000000000000
   0.20000000000000   0.58778525229247   0.58778525229247   0.00000000000000
   0.30000000000000   0.80901699437495   0.80901699437495                   0
   0.40000000000000   0.95105651629515   0.95105651629515   0.00000000000000
   0.50000000000000   1.00000000000000   1.00000000000000                   0
   0.60000000000000   0.95105651629515   0.95105651629515   0.00000000000000
   0.70000000000000   0.80901699437495   0.80901699437495                   0
   0.80000000000000   0.58778525229247   0.58778525229247                   0
   0.90000000000000   0.30901699437495   0.30901699437495   0.00000000000000
   1.00000000000000                   0   0.00000000000000   0.00000000000000
```

La tabla anterior muestra la solución aproximada, la solución real y el error cometido en la aproximación en el instante t = 20.

A continuación se muestra, en azul, la solución aproximada y, en rojo, la exacta.


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

3. Ecuanciones en derivadas parciales elípticas.

Sea

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) \qquad (x,y) \in \Re$$

conocida como ecuación de Poisson, con condiciones:

$$u(x,y) = g(x,y) \qquad (x,y) \in S$$

donde

$$R = \{(x,y), a < x < y, c < y < d\}$$

El primer paso consiste en realizar un *malleado* del rectángulo R escogiendo dos enteros n y m, y definiendo los tamaños de paso h y k:

$$h = \frac{b-a}{n} \qquad k = \frac{d-c}{m}$$

La malla hace pasar rectas verticales y horizontales por los puntos con coordenadas x_i, y_j :

$$x_i = a + ih \qquad i = 0, 1, \ldots, n$$
$$y_j = c + jh \qquad j = 0, 1, \ldots, m$$

En cada punto de red (x_i, y_j) :

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j)$$
$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2} - \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, muy_j)$$

donde $\xi_i \in (x_{i-1}, x_{i+1}), muy_j \in (y_{i-1}, y_{i+1})$. Por tanto,

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y)$$

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2} = f(x_i, y_j)$$

para cada i = 1, 2, ..., (n-1) y j = 1, 2, ..., (m-1), y las condiciones de frontera

$$
\begin{aligned}
u(x_0, y_j) &= g(x_0, y_j) \quad \textit{para cada } j = 0, 1, \ldots, m, \\
u(x_n, y_j) &= g(x_n, y_j) \quad \textit{para cada } j = 0, 1, \ldots, m, \\
u(x_i, y_0) &= g(x_i, y_0) \quad \textit{para cada } i = 1,2, \ldots, n-1, \\
u(x_i, y_m) &= g(x_i, y_m) \quad \textit{para cada } i = 1,2, \ldots, n-1.
\end{aligned}
$$

En la forma de ecuación de diferencia, esto da lugar a un método llamado de diferencia centrada, con error de truncamiento local de orden $O(h^2 + k^2)$, que puede escribirse como:

$$
2[(\frac{h}{k})^2 + 1]w_{i,j} - (w_{i+1,j} + w_{i-1,j}) - (\frac{h}{k})^2 (w_{i,j+1} + w_{i,j-1}) = -h^2 f(x_i, y_j)
$$

para cada i = 1, 2, ..., n y j = 1, 2, ..., m-1, y

$$
\begin{aligned}
w_{0,j} &= g(x_0, y_j) \quad \textit{para cada } j = 0, 1, \ldots, m, \\
w_{n,j} &= g(x_n, y_j) \quad \textit{para cada } j = 0, 1, \ldots, m, \\
w_{i,0} &= g(x_i, y_0) \quad \textit{para cada } i = 1,2, \ldots, n-1, \\
w_{i,m} &= g(x_i, y_m) \quad \textit{para cada } i = 1,2, \ldots, n-1.
\end{aligned}
$$

donde $w_{i,j}$ aproxima a $u(x_i, y_j)$.

```
% laplace.m - Aproximación a la solución de la ecuación de Laplace
%
% Entradas:
%   cf, condiciones de frontera
%   a <= x <= b
%   c <= y <= d
%
% Salidas:
%   A, matriz de los coeficientes
%   b, matriz de los términos independientes
%
function [A, b]=laplace(cf, a, b, c, d, n, m)
h = (b-a)/n;
k = (d-c)/m;

% Cálculo de los puntos de red
index = 1;
for i = 0:n
    x(index) = a + i*h;
    index = index + 1;
end
index = 1;
for j = 0:m
    y(index) = c + j*k;
    index = index + 1;
end

% Construcción de la matriz A y del vector b del sistema
b = zeros((n-1)*(m-1), 1);
for i = 1:n-1
    for j = 1:m-1
        % Punto Wij
        fila=i+(m-1-j)*(n-1);
        col=fila;
        P(fila, col) = 2*(h^2/k^2 + 1);

        % Punto Wi-1,j
        if i > 1
            col = (i-1) + (m-1-j)*(n-1);
            P(fila, col) = -1;
        else
```

```
            b(fila) = b(fila) + feval(cf, x(1), y(j+1));
        end

        % Punto Wi+1,j
        if i < n-1
            col=(i+1)+(m-1-j)*(n-1);
            P(fila, col) = -1;
        else
            b(fila) = b(fila) + feval(cf, x(n+1), y(j+1));
        end

        % Punto Wi,j-1
        if j < m-1
            col=i+(m-1-j-1)*(n-1);
            P(fila, col) = -(h^2/k^2);
        else
            b(fila) = b(fila) + feval(cf, x(i+1), y(m+1));
        end

        % Punto Wi,j+1
        if j > 1
            col=i+(m-1-j+1)*(n-1);
            P(fila, col) = -(h^2/k^2);
        else
            b(fila) = b(fila) + feval(cf, x(i+1), y(1));
        end
    end
end
A=P;
```

Ejemplo

Considérese el problema de determinar la distribución estacionaria de calor en una lámnia delgada de metal en forma de cuadrado con dimensiones de $0.5 \times 0.5\, m^2$, la cual se mantiene a 0°C en dos fronteras adyacentes mientras que el calor en las otras fronteras se va incrementando linealmente de 0°C en una esquina a 100°C donde estos lados se ecuentran. Si ponemos los lados con condición de frontera cero a lo largo de los ejes x e y, el problema se expresará matemáticamente como:

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = 0$$

para (x, y) en el conjunto R = {(x, y), 0 < x < 0.5, 0 < y < 0.5}, con las condiciones de frontera.

$$u(0, y) = 0, u(x, 0) = 0, u(x, 0.5) = 200x, u(0.5, y) = 200y$$

Solución

Definimos el archivo con las condiciones iniciales:

```
function u=condiciones(x, y)
if x == 0
    u = 0;
elseif x == 0.5
    u = 200*y;
elseif y == 0
    u = 0;
elseif y == 0.5
    u = 200*x;
end
```

Si n = m = 4,

```
>> [A, b]=laplace('condiciones', 0, 0.5, 0, 0.5, 4, 4)

A =

     4    -1     0    -1     0     0     0     0     0
    -1     4    -1     0    -1     0     0     0     0
     0    -1     4     0     0    -1     0     0     0
    -1     0     0     4    -1     0    -1     0     0
     0    -1     0    -1     4    -1     0    -1     0
     0     0    -1     0    -1     4     0     0    -1
     0     0     0    -1     0     0     4    -1     0
     0     0     0     0    -1     0    -1     4    -1
     0     0     0     0     0    -1     0    -1     4


b =

    25
    50
   150
     0
     0
    50
     0
     0
    25

>> A\b

ans =

   18.7500
   37.5000
   56.2500
   12.5000
   25.0000
   37.5000
    6.2500
   12.5000
   18.7500
```

Ejemplo

Vamos a determinarla solución aproximada de la ecuación de Laplace $\nabla^2 u = 0$ en el rectángulo $R = (x, y): 0 \le x \le 4, 0 \le y \le 4$, donde u(x, y) denota la temperatura en un punto (x, y), los valores en la frontera son:

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

Solución

Definimos el archivo con los valores de frontera:

```
% frontera.m
function u=frontera(x, y)
if x == 0
    u = 80;
elseif x == 4
    u = 0;
elseif y == 0
    u = 20;
elseif y == 4
    u = 180;
end
```

```
>> [A, b]=laplace('frontera', 0, 4, 0, 4, 4, 4);
>> A\b

ans =

  1.0e+002 *

   1.12857142857143
   1.11785714285714
   0.84285714285714
   0.79642857142857
   0.70000000000000
   0.45357142857143
   0.55714285714286
   0.43214285714286
   0.27142857142857
```