PRÁCTICA Num. 1
Introducción a *MATHEMATICA*
métodos directos para la resolución de sistemas de ecuaciones lineales

INTRODUCCION:

El programa *Mathematica* tiene incorporadas la mayor parte de las funciones matemáticas elementales, por ejemplo **Exp** es la función exponencial, **Sqrt** es la raíz cuadrada, **Sin**, **Cos,** T**an,** ... son las funciones trigonométricas, ... Además pueden definirse nuevas funciones. La definición de una función se hace con una sentencia del tipo

nombre[variable1,variable2,...] := expresión.

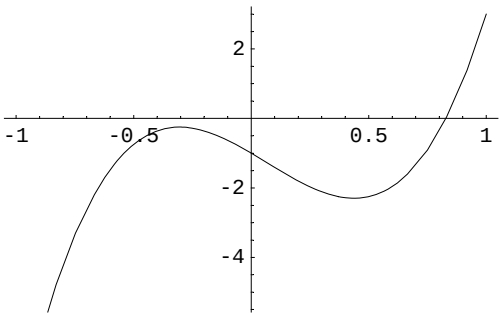
Por ejemplo

```
In[1]:= p[x_]:=10*x^3-2*x^2-4*x-1
```

define un polinomio de tercer grado. Es buena costumbre utilizar el comando **Clear[p]** antes de definir una función para eliminar algún valor que pudiera tener asignado con anterioridad. Si queremos eliminar todas las variables, funciones, ... entonces es mejor la orden **Clear["Global`*"],** que debería incluirse al comienzo de cada Notebook.

Si queremos dibujar la gráfica de la función p[x] definida anteriormente en un intervalo [a,b], la orden es **Plot[f[x],{x,a,b}]**. Si la queremos dibujar en el intervalo [-1,1], la sentencia que hay que escribir es

```
In[2]:= Plot[p[x],{x,-1,1}];
```



Las funciones pueden ser bastante mas complicadas. Por ejemplo, si tenemos

$$f(x)=1+x+x^2/2+...+x^{50}/50,$$

está claro que escribir todos los términos a mano sería muy pesado. Sin embargo, la suma anterior puede efectuarse directamente con la orden **Sum[expresion, {contador, inicio, final, paso}]**, que en nuestro caso es

```
In[3]:= Clear[f,x]
        f[x_]:= 1+Sum[x^i/i,{i,1,50,1}]
```

Si quisieramos evaluar f[1/2], obtendríamos

```
In[5]:= f[1/2]

Out[5]= 14769381771754510488752609273807905 9
        87230347965792839223946208178339840
```

y para saber cual es el valor numérico se utiliza **N[expresion,precision]**. Así, para ver el valor de f(1/2) con precisión de 10 dígitos, sería

```
In[6]:= N[f[1/2],10]

Out[6]= 1.693147181
```

Ejercicios: Definir las siguientes funciones y dibujarlas en el intervalo [0,5]

1) $\text{sen}(x)/(x + x^2 + 100\cos(x^3))$
2) $1+x/1+x^2/2!+x^3/3! + ...+x^{40}/40!$
3) $1-x^2/2!+x^4/4!-x^6/6! + ...+x^{60}/60!$

Es conveniente conocer las órdenes que tiene Mathematica que definen bucles o fenómenos repetitivos. Entre otras, podemos citar a **For** y **Do**, cuyas estructuras son del tipo

For[contador=inicio, condicion, paso, proceso]
Do[proceso,{contador, inicio, fin,paso}]

La función $f(x)=1+x+x^2/2+...+x^{50}/50$ se puede escribir como cualquiera de las formas siguientes utilizando las estructuras

```
In[13]:= suma=1;
         For[i=1, i<=50, i++, suma = suma + x^i/i]

         suma=1;
         For[i=50, i>=1, i--, suma = suma + x^i/i]

         suma=1;
         Do[suma = suma + x^i/i,{i,1,50}]
```

Para evaluar la expresion suma en x=1/2, al no ser una función es preciso sustituir este valor. Esto se hace

```
In[19]:= suma/.x->0.5

Out[19]= 1.69315
```

o bien definiendo la función de la forma

```
In[20]:= Clear[f]
         f[x_]=suma;
         f[0.5]

Out[22]= 1.69315
```

Ejercicio: Repetir los dos últimos ejercicios anteriores con las ordenes **For** y **Do**

ERRORES DE REDONDEO:

Ejemplo: Aunque *Mathematica* pueda trabajar en aritmética exacta, hay ocasiones en las que esto no es posible. Se trata de ver que las propiedades usuales que tienen los números reales se pierden en aritmética de coma flotante. En particular, depende cómo se hagan las operaciones el resultado puede ser distinto.

```
In[23]:= x=1.234567890123457;
         y=1.234567890123467;
         z=10^(-15);
         (x-y)/z+10
         x/z-y/z+10

Out[26]= 0.00799278

Out[27]= 0.
```

Ejemplo: En este ejemplo vamos a tratar la función exponencial, que siempre es positiva, y se trata de evaluar para valores de x negativos, en vez de la función exponencial su desarrollo de Taylor con suficientes términos .

```
In[28]:= x=-40.;
         Sum[x^i/i!,{i,0,100}]
         Sum[x^i/i!,{i,0,300}]

Out[29]= 53.7441

Out[30]= 4.79914
```

Ejercicio: Hacer lo mismo con la función cos(x)= 1-x^2/2!+x^4/4!-x^6/6! + ...+x^n/n!+... evaluándola en distintos valores de x.

Ejemplo: Consideramos la siguiente fórmula para estimar el número π, $p_{n+1}=2^n\sqrt{2\left(1-\sqrt{1-(p_n/2^n)^2}\right)}$, donde p_n es el perímetro de los polígonos regulares de2^n lados inscritos en una circunferencia de diametro unidad. Por este método, Arquímedes consiguió uitlizando un polígono de 96 lados los dos primeros decimales exactos. Siglos más tarde, Vietta tuvo que utilizar polígonos de 400.000 lados, para lograr una aproximación de 10 decimales.

```
In[31]:= Clear[p,n,f]
         f[p_,n_]=2^n*Sqrt[2*(1-Sqrt[1-(p/2^n)^2])];
```

Empezamos con n=2 y por tanto $p_2=2\sqrt{2}$. Con este valor calcularemos los restantes, escribiendo el error cometido cada dos iteraciones, viendo como se degrada la aproximación

```
In[33]:= p[2] = 2 * Sqrt[2.]; Do[p[n + 1] = f[p[n], n], {n, 2, 30}];
         Do[ Print[2^i, "  ", N[Pi - p[i]]], {i, 2, 30, 2}]
```

```
4  0.313166

16  0.0201475

64  0.0012615

256  0.0000788524

1024  4.92831×10^-6

4096  3.07979×10^-7

16384  2.01265×10^-8

65536  8.26858×10^-9

262144  -2.5735×10^-7

1048576  -3.90012×10^-6

4194304  0.0000738131

16777216  -0.000858619

67108864  -0.020685

268435456  0.313166

1073741824  3.14159
```

Ejemplo: En este otro ejemplo, se trata de generar la sucesión 1, 1/3, 1/9, 1/27, ... por varias fórmulas, viendo que en algún caso los resultados no son los esperados.

```
In[35]:= Clear[p,n];m=50;
         p[0]=1; Do[p[n]=1/3*p[n-1],{n,1,m}]; N[p[m]]
         p[0]=1;p[1]=0.33333;
         Do[p[n]=10/3*p[n-1]-p[n-2],{n,2,m}]; N[p[m]]

Out[36]= 1.39296×10^-24

Out[38]= -8.97372×10^17
```

Ejercicio: Sean las ecuaciones en diferencias p_n=5/6 p_{n-1}-1/6 p_{n-2}, p_n=5/3 p_{n-1}-4/9 p_{n-2}, con valores $p_0 = 1$, p_1=0.33333. Comprobar que $p_n = (1/3)^n$ es solución. Hallar los primeros términos de las mismas.

MATRICES:

Para definir y manejar matrices, *Mathematica* dispone de gran variedad de órdenes. A continuación supondremos que A es una matriz cuadrada de dimensión nxn y b es un vector de dimensión n. Las órdenes típicas son:

(1) **DiagonalMatrix[{a11,a22,a33,...}]** define una matriz diagonal con elementos a11, a22,...
(2) **IdentityMatrix[n]** define la matriz identidad de dimension n
(3) **Transpose[A]** da la transpuesta de la matriz A
(4) **Inverse[A]** da la inversa de la matriz A
(5) **Det[A]** da el determinate de A
(6) **LinearSolve[A, b]** resuelve el sistema lineal Ax=b
(7) **A[[i,j]]** es el elemento(i,j) de la matriz A
(8) **A[[i]]** es la fila i-ésima de la matriz A
(9) **Eigenvalues[A]** nos da los valores propios de A

Ejercicio: Definir la matriz diagonal A de dimensión 4 con valores {1,2,3,4}. Hallar su inversa, el determinante y A^3. Repetir lo anterior con la matriz A={{0,0},{1,1}} y calcular A^2.

Ejercicio: Hallar un polinomio de segundo grado, $p_2(x)$ que verifique $p_2(1) = 1$, $p_2(2) = 1$, $p_2(3) = 2$.

RESOLUCION DE SISTEMAS LINEALES: (metodos directos) En primer lugar se pretende resolver un sistema lineal con matriz triangular superior. En este caso, se va resolviendo de abajo hacia arriba y una expresión de la solución es

$$x_i = (b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j) / \, a_{ii}, \quad \text{i=n, n-1, ... , 1.}$$

Lo anterior puede implementarse de la siguiente manera:

```
A={{1,2,3},{0,1,2},{0,0,1}}; b={6,3,1};
dim=3;
x={0,0,0};
For[i=dim,i>=1,i--,
   x[[i]]=(b[[i]]-
       Sum[A[[i,j]]*x[[j]],{j,i+1,dim}])/A[[i,i]]
   ]
Print[x]
```

```
{1, 1, 1}
```

Utilizando **LinearSolve[A,b]**, obtenemos el mismo resultado

```
LinearSolve[A,b]
```

```
{1, 1, 1}
```

Ejemplo: En este ejemplo se trata de efectuar el proceso de eliminación Gaussiana sobre una matriz 4x4, definiendo todas las matrices L_i, por las que hay que multiplicar a izquierda a la matriz A para conseguir hacer ceros por debajo de la diagonal.

```
A={{10,7,8,7},{7,5,6,5},{8,6,10,9},{7,5,9,10}};
Aoriginal=A;
A//MatrixForm
```

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$$

```
L1=IdentityMatrix[4];
For[i=2,i<=4,i++,L1[[i,1]]=-A[[i,1]]/A[[1,1]]]
MatrixForm[L1]
A=L1.A; MatrixForm[A] (* construccion de la primera matriz L1 *)
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{7}{10} & 1 & 0 & 0 \\ -\frac{4}{5} & 0 & 1 & 0 \\ -\frac{7}{10} & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 0 & \frac{1}{10} & \frac{2}{5} & \frac{1}{10} \\ 0 & \frac{2}{5} & \frac{18}{5} & \frac{17}{5} \\ 0 & \frac{1}{10} & \frac{17}{5} & \frac{51}{10} \end{pmatrix}$$

```
L2=IdentityMatrix[4];
For[i=3,i<=4,i++,L2[[i,2]]=-A[[i,2]]/A[[2,2]]]
MatrixForm[L2]
A=L2.A; MatrixForm[A] (* construccion de la segunda matriz L2 *)
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -4 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 0 & \frac{1}{10} & \frac{2}{5} & \frac{1}{10} \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 3 & 5 \end{pmatrix}$$

```
L3=IdentityMatrix[4];
For[i=4,i<=4,i++,L3[[i,3]]=-A[[i,3]]/A[[3,3]]]
MatrixForm[L3]
A=L3.A; MatrixForm[A] (* construccion de la tercera matriz L3 *)
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{3}{2} & 1 \end{pmatrix}$$

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 0 & \frac{1}{10} & \frac{2}{5} & \frac{1}{10} \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

```
U=A; L=Inverse[L3.L2.L1]; MatrixForm[L]; (* como las modificaciones las hemos
realizado sobre la matriz A, al final del proceso es triangular superior, y la
llamamos U *)

Aoriginal==L.U
```

```
True
```

Resolveremos ahora el sistema lineal con vector independiente {32,23,33,31} con la orden **LinearSolve[]** y con la implementación directa de la fórmula para comprobar que da la misma solución

```
b={32,23,33,31};
LinearSolve[Aoriginal,b]

bb=L3.L2.L1.b;
dim=4;
x={0,0,0,0};
For[i=dim,i>=1,i--,
   x[[i]]=(bb[[i]]-
       Sum[A[[i,j]]*x[[j]],{j,i+1,dim}])/A[[i,i]]
   ]
Print[x]
```

```
{1, 1, 1, 1}
```

```
{1, 1, 1, 1}
```

A continuación se trata de resolver un sistema lineal ligeramente modificado en el vector independiente y ver lo que ocurre.

```
b={32.1,22.9,33.1,30.9};
LinearSolve[Aoriginal,b]

bb=L3.L2.L1.b;
dim=4;
x={0,0,0,0};
For[i=dim,i>=1,i--,
   x[[i]]=(bb[[i]]-
       Sum[A[[i,j]]*x[[j]],{j,i+1,dim}])/A[[i,i]]
   ]
Print[x]
```

```
{9.2, -12.6, 4.5, -1.1}
```

```
{9.2, -12.6, 4.5, -1.1}
```

Ejercicio: Hacer lo mismo con la matriz de Hilbert 4x4 y vector independiente b={4,163/60, 21/10, 241/140} utilizando 4 decimales.

Ejercicio: Comprobar que la matriz A={{ 5, 1, 2},{1,8,3},{2,3,8}} es simétrica y puede factorizarse en la forma $L\,\mathrm{DL}^T$

En este ejemplo vamos a ver cómo el no efectuar pivotaje lleva a una solución bastante mala.

```
Clear[b,A,boriginal,Aoriginal,p]
b={1,1,1,1,1,1};
A={{p,-1,0,0,0,0},{0,p,-1,0,0,0},{0,0,p,-1,0,0},
{0,0,0,p,-1,0},{0,0,0,0,p,-1},{1,1,1,1,1,1}};

Aoriginal=A;
boriginal=b;
A//MatrixForm
p=10.^(-10);
```

$$\begin{pmatrix} p & -1 & 0 & 0 & 0 & 0 \\ 0 & p & -1 & 0 & 0 & 0 \\ 0 & 0 & p & -1 & 0 & 0 \\ 0 & 0 & 0 & p & -1 & 0 \\ 0 & 0 & 0 & 0 & p & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

El proceso de multiplicar a izquierda por las matrices L_i para realizar la factorización LU de una matriz puede implementarse muy facilmente en *Mathematica* de la forma siguiente:

```
dim=6;
For[i=1,i<=dim,i++,
   For[j=i+1,j<=dim,j++,
      piv = -A[[j,i]]/A[[i,i]];
      A[[j]] = A[[j]] + piv*A[[i]];
      b[[j]] = b[[j]] + piv*b[[i]];
   ]
]
```

proceso de resolución un sistema lineal con matriz triangular superior

```
x = Table[0, {i, dim}];
For[i = dim, i >= 1, i--,
 x[[i]] = (b[[i]] -
   Sum[A[[i, k]] * x[[k]], {k, i + 1, dim}]) / A[[i, i]]
 ]
Print[N[x]]
```

```
{-8.26404×10^32, -8.26404×10^22, -8.26404×10^12, -827.404, -1., -1.}
```

Ahora vemos cómo de bien satisface la solución calculada el sistema lineal

```
N[Aoriginal.x-boriginal]
```

```
{-1., -0.000976563, 0., 0., 0., -8.26404×10^32}
```

Si lo hacemos con la orden **LinearSolve**, que sí que hace pivote, el resultado es bastante diferente

```
y=N[LinearSolve[Aoriginal,boriginal]]
Aoriginal.y-boriginal
```

```
{6., -1., -1., -1., -1., -1.}
```

```
{0., 0., 0., 0., 0., 0.}
```

RESOLUCIÓN DE SISTEMAS LINEALES (métodos iterativos) :

Ejemplo: Considerar el sistema de ecuaciones lineales Ax=b, con matriz A={{2,1,1},{1,2,1},{1,1,2}} y vector b={4,4,4}. Calcular el radio espectral de las matrices de iteración de Jacobi y Gauss-Seidel. Razonar la convergencia de dichos métodos.

```
In[39]:= A = {{2, 1, 1}, {1, 2, 1}, {1, 1, 2}};
         b = {4, 4, 4};
```

definimos la parte diagonal asi como las matrices triangulares cambiadas de signo

```
In[41]:= L = -{{0, 0, 0}, {1, 0, 0}, {1, 1, 0}};
         U = -{{0, 1, 1}, {0, 0, 1}, {0, 0, 0}};
         Di = {{2, 0, 0}, {0, 2, 0}, {0, 0, 2}};
         (* definicion de la matriz de iteracion de Jacobi *)
         BJ = Inverse[Di].(L + U); cJ = Inverse[Di].b;
         Eigenvalues[BJ]
         (* definicion de la matriz de iteracion de Gauss-Seidel *)
         BG = Inverse[Di - L].U; cG = Inverse[Di - L].b;
         Eigenvalues[BG]
         Abs[N[%]]
```

$$\text{Out[45]}= \left\{-1, \frac{1}{2}, \frac{1}{2}\right\}$$

$$\text{Out[47]}= \left\{\frac{1}{16}\left(5 + i\sqrt{7}\right), \frac{1}{16}\left(5 - i\sqrt{7}\right), 0\right\}$$

```
Out[48]= {0.353553, 0.353553, 0.}
```

En el caso de Jacobi, el radio espectral de la matriz sale 1, por lo que no hay convergencia. En Gauss-Seidel, como los valores propios salen complejos utilizamos la orden **Abs[N[%]]** para calcular el radio espectral (Notar que el simbolo % hace referencia al último resultado calculado). En el caso de Gauss-Seidel, el proceso es convergente. A continuación implementamos los procesos iterativos para verificar lo anterior.

```
In[49]:= x = {0, 0, 0}; maxite = 5;
         For[i = 0, i <= maxite, i++, x = BJ.x + cJ; Print[N[x]]]
```

```
{2., 2., 2.}

{0., 0., 0.}

{2., 2., 2.}

{0., 0., 0.}

{2., 2., 2.}

{0., 0., 0.}
```

```
In[51]:= x = {0, 0, 0}; maxite = 8;
         For[i = 0, i <= maxite, i++, x = BG.x + cG; Print[N[x]]]

        {2., 1., 0.5}

        {1.25, 1.125, 0.8125}

        {1.03125, 1.07813, 0.945313}

        {0.988281, 1.0332, 0.989258}

        {0.98877, 1.01099, 1.00012}

        {0.994446, 1.00272, 1.00142}

        {0.997932, 1.00032, 1.00087}

        {0.999402, 0.999863, 1.00037}

        {0.999885, 0.999874, 1.00012}
```

Ejercicio: Repetir el ejercicio anterior para el sistema lineal

x+z=2, -x+y=0, x+2y-3z=0.

Ejemplo: Considerar el sistema de ecuaciones

3x+y+z=4, -x+y+3z=4, 2x+5y+z=-1.

Aplicar el método de Jacobi para resolver dicho sistema.

Ejercicio: Repetir el ejercicio anterior para el sistema lineal resultante de permutar la segunda y tercera fila de la matriz de coeficientes del sistema lineal anterior.

Ejercicio: Considerar el esquema iterativo de Jacobi dado por

$$x^{(n+1)} = \begin{pmatrix} 0 & -0.1 \\ 2 & 0 \end{pmatrix} x^{(n)}$$

a) Hallar un sistema lineal Ax=b tal que al aplicar Jacobi, nos de el esquema iterativo anterior. Hallar su solución exacta.

b) Hallar el radio espectral y estudiar su convergencia.

c) Dar unas iteraciones partiendo del punto inicial $x^{(0)} = (1, 1)^T$ y verificar que el error no decrece de forma monótona.