PRÁCTICA Num. 3
Aproximación e Interpolación.

El problema de la teoría de aproximación discreta consiste en el ajuste de funciones a unos datos conocidos y encontrar la que "mejor" represente al conjunto de datos en el sentido de los mínimos cuadrados. Por otra parte, la interpolación es un proceso por el cual se busca una función interpoladora (en nuestro caso un polinomio) que pase a través de todos los puntos de un conjunto dado.

Ejemplo: Halla la recta de mínimos cuadrados para el conjunto de puntos (2,2), (4,11), (5,23), (6,28), (7,38) y (8,40).
En primer lugar consideraremos la ecuación de la recta **y= a0 + a1 * x**.
Imponiendo las ecuaciones anteriores obtenemos la matriz A y el vector b dados por

```
A={{1,2},{1,4},{1,5},{1,6},{1,7},{1,8}};
b={2,11,23,28,38,40};
```

Planteamos las ecuaciones normales **Transpose[A] A x =Transpose[A] b**,

```
matriz = Transpose[A].A;
vector = Transpose[A].b;
MatrixForm[matriz]
MatrixForm[vector]
sol=LinearSolve[matriz,vector]
recta = {1,x}.sol
```

$$\begin{pmatrix} 6 & 32 \\ 32 & 194 \end{pmatrix}$$

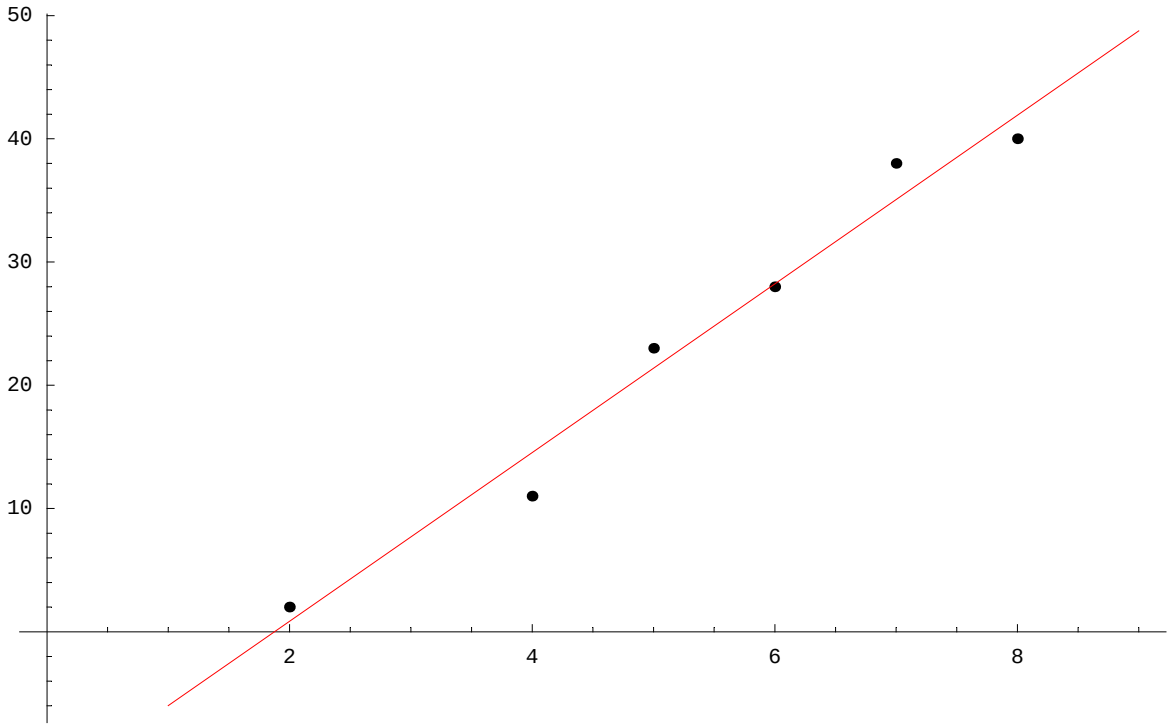
$$\begin{pmatrix} 142 \\ 917 \end{pmatrix}$$

$$\left\{-\frac{449}{35}, \frac{479}{70}\right\}$$

$$-\frac{449}{35} + \frac{479\,x}{70}$$

Dibujando los puntos y la recta, resulta

```
puntos={{2,2},{4,11},{6,28},{8,40}};
(* la orden ListPlot se utiliza para dibujar un conjunto de puntos, y la opción
Prolog -> AbsolutePointSize[7] para aumentar el tamaño de los mismos *)
dibu1=ListPlot[puntos,Prolog -> AbsolutePointSize[7]];
dibu2=Plot[recta,{x,1,11}];
Show[dibu1,dibu2];
```



El proceso anterior se podía haber realizado como la minimización de la siguiente función de dos variables, y hallando lo

```
F[a0_, a1_] = (a0 + a1 * 2 - 2) ^2 + (a0 + a1 * 4 - 11) ^2 + (a0 + a1 * 5 - 23) ^2 +
   (a0 + a1 * 6 - 28) ^2 + (a0 + a1 * 7 - 38) ^2 + (a0 + a1 * 8 - 40) ^2;
fa0 = D[F[a0, a1], a0];
fa1 = D[F[a0, a1], a1];
Solve[{fa0 == 0, fa1 == 0}, {a0, a1}]
```

$$\left\{\left\{a0 \to -\frac{449}{35},\ a1 \to \frac{479}{70}\right\}\right\}$$

que obviamente da la misma solución.

Ejercicio: Considera la evaluacion de un polinomio de grado n utilizando el algoritmo directo:

s=a[[0]]; For[i=1,i<=n,i++,s=s+a[[i]]*x^i]

y el algoritmo de Horner dado por :

s=a[[n]]; For[i=n-1,i>=0,i--,s=a[[i]]+s*x].

Calcula el número de operaciones en cada uno de ellos. En la evaluación de polinomios de grados n=1.000, 1.500, 2.000, 2.500 y 3.000, se obtienen los siguientes tiempos de ejecución para el algoritmo directo {2.25,4.51,6.76,9.12,11.59} y para el de Horner {0.94,1.37,1.86,2.36,2.75}, respectivamente.

Halla polinomios de grados apropiados para el ajuste por mínimos cuadrados de los datos anteriores. Los coeficientes del polinomio pueden tomarse **a[[i]]=1/i, i=1,...,n** y el punto de evaluación x=1/3.

Ejercicio: Considera la determinación de la constante k de un muelle en la ley de Hooke: f(x) = k x.
Los datos tomados son: $(x_i, f(x_i))$=(2,1.7), (3, 5), (4,4.1), (5,6), (6,7), (8,9.1), (10,10.6). Determina la aproximación por mínimos cuadrados a la constante k.

Ejemplo: Sea el problema de interpolación polinómica a la función f(x)=sen(x), en los puntos 0, 1/2, y 1.
Halla el polinomio de interpolación de grado <=2.
Dibuja la función y su polinomio de interpolación.
Calcula el error de interpolación y halla una cota superior del mismo.

```
Clear[p2,x,a0,a1,a2]
p2[x_]=a0 + a1*(x-0) + a2*(x-0)*(x-1/2);
eq1= p2[0]-Sin[0]==0
eq2= p2[1/2]-Sin[1/2]==0
eq3= p2[1]-Sin[1]==0
Solve[{eq1,eq2,eq3},{a0,a1,a2}]//N
```

$$a0 == 0$$

$$a0 + \frac{a1}{2} - \text{Sin}\left[\frac{1}{2}\right] == 0$$

$$a0 + a1 + \frac{a2}{2} - \text{Sin}[1] == 0$$

$$\{\{a0 \to 0,\ a1 \to 0.958851,\ a2 \to -0.23476\}\}$$

por tanto el polinomio es 0.958851 * x -0.23476*x*(x-1/2)

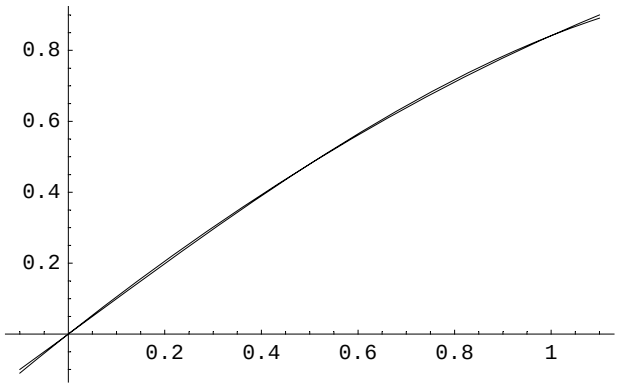
```
p2[x_]=0.958851 * x -0.23476*x*(x-1/2);
```

Lo anterior podia haberse hecho con las sentencias de *Mathematica* :

```
lista = {{0, Sin[0]}, {0.5, Sin[0.5]}, {1., Sin[1.]}};
p2[x_] = InterpolatingPolynomial[lista, x]
```

$$(0.958851 - 0.23476\ (-0.5 + x))\ x$$

```
Plot[{Sin[x], p2[x]}, {x, -0.1, 1.1}];
```



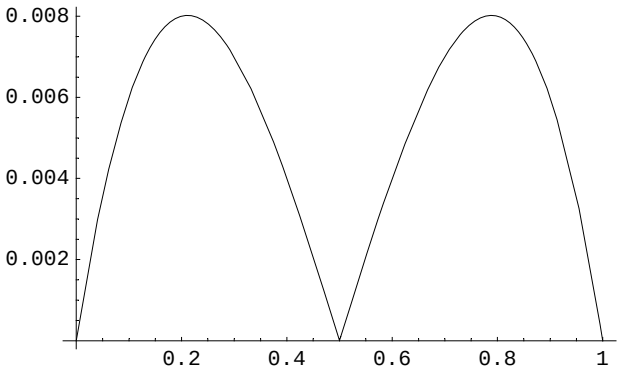
puede apreciarse que el polinomio de interpolación es bastante bueno. Por otra parte, el error de interpolación viene dado por
err_n(x) = $f^{(n+1)}(\eta) / (n+1)! (x - x_0) .... (x - x_n)$,
que en nuestro caso concreto es

```
err= D[Sin[y],{y,3}]/3!*(x-0)*(x-1/2)*(x-1)
```

$$-\frac{1}{6}\,(-1+x)\left(-\frac{1}{2}+x\right)x\,\text{Cos}[y]$$

y como | cos(y)|≤1, ∀ y, tenemos que | err | ≤ | (x-0) (x-1/2) (x-1) |/6.

```
errabs = Abs[(x - 0) * (x - 1 / 2) * (x - 1)] / 6;
Plot[errabs, {x, 0, 1.}];
```



luego el máximo error cometido por esta interpolación cuadrática es aproximadamente 0.008.
Ejercicio: Hacer lo mismo que el ejemplo anterior para la función f(x) = sen(exp(x)) en los puntos x_i = `0, 1 / 2, 1, 3 / 2, 2.`

Ejercicio: Interpola la función f(x)=1/(1 + 20x^2) por polinomios basados en puntos equidistantes en el intervalo [-1,1]. Dibuja los polinomios de interpolación y la función observando lo que ocurre. ¿Hay convergencia de los polinomios a la función cuando n crece?

Ejemplo: En este ejemplo se trata de determinar algunas fórmulas de cuadratura usuales. Primeramente hallaremos la del trapecio en el intervalo [a,b]. Para ésta, se considera el polinomio de grado <=1 que interpola a f(x) en x_i=a, b y luego se integra en el intervalo [a,b] para aproximar a la integral definida de la función en dicho intervalo.

```
Clear[f, a, b, x]
(* Regla del Trapecio: polinomio de grado<=1 que pasa por (a,f(a)) y (b,f(b))*)
pol = InterpolatingPolynomial[{{a, f[a]}, {b, f[b]}}, x];
trapecio = Integrate[pol, {x, a, b}] // Simplify
```

$$-\frac{1}{2}\,(a-b)\,(f[a]+f[b])$$

el error en la regla del trapecio lo calcularemos en el intervalo [0,h], (sustituir a=0 y b=h) y viene dado por
error=$\int_0^h f(x)\,dx - h/2*(f(0)+f(h))$,
el cual desarrollaremos por la fórmula deTaylor, para cuantificar el grado de aproximación.

```
error = Integrate[f[x],{x,0,h}]- h/2*( f[0] + f[h]);
Series[error,{h,0,3}]
```

$$-\frac{1}{12}\,f''[0]\,h^3+O[h]^4$$

Ejercicio: Hacer lo mismo que el ejemplo anterior para la fórmula de Simpson.

Ejemplo: En este ejemplo se trata de hallar una formula de cuadratura de grado de precisión o exactitud máximo para aproximar

$\int_0^1 f(x)\,dx \simeq b0 * f(c0) + b1 * f(c1)$.

Como hay cuatro parametros, impondremos grado de precisión 3 al menos.

```
eq1 = b0 + b1 == 1;
eq2 = b0 * c0 + b1 * c1 == 1 / 2;
eq3 = b0 * c0^2 + b1 * c1^2 == 1 / 3;;
eq4 = b0 * c0^3 + b1 * c1^3 == 1 / 4;
Solve[{eq1, eq2, eq3, eq4}, {b0, b1, c0, c1}]
N[%]
```

$$\left\{\left\{b0 \to \frac{1}{2}, b1 \to \frac{1}{2}, c1 \to \frac{1}{6}\left(3-\sqrt{3}\right), c0 \to \frac{1}{6}\left(3+\sqrt{3}\right)\right\},\right.$$
$$\left.\left\{b0 \to \frac{1}{2}, b1 \to \frac{1}{2}, c1 \to \frac{1}{6}\left(3+\sqrt{3}\right), c0 \to \frac{1}{6}\left(3-\sqrt{3}\right)\right\}\right\}$$

$$\{\{b0 \to 0.5, b1 \to 0.5, c1 \to 0.211325, c0 \to 0.788675\},$$
$$\{b0 \to 0.5, b1 \to 0.5, c1 \to 0.788675, c0 \to 0.211325\}\}$$

Ejemplo: Aplica la fórmula anterior al cálculo de $\int_0^1 sen(cos(x))\,dx$

Ejemplo: ¿Como calcularías la integral doble $\int_0^1 \int_0^1 cos(xy)\,dx\,dy$?