ÁLGEBRA Y GEOMETRÍA

Grado en ingeniería en Electrónica Industrial y Automática
Grado en ingeniería Eléctrica
Universidad de Cantabria

Práctica 5 (SAGE)

Aspectos generales de las prácticas de laboratorio:

- Abrir un navegador (recomendado Firefox o Chrome) y en la barra de dirección escribir

  193.146.75.191:8080

  Entrar con vuestras credenciales: Username y Password.
- Crear la carpeta *practicas_algebra* en el escritorio y guardar todos los archivos realizados durante la clase en esa carpeta.
- Al finalizar la clase subir estos archivos a vuestro directorio en moodle y eliminar la carpeta *practicas_algebra* del escritorio.

En muchas ocasiones conocemos el valor de una función $f(x)$ en un conjunto de puntos $x_0, x_1, \ldots, x_n$, pero no tenemos una exprexión de $f(x)$ que nos permita saber el valor en otros puntos distintos. Por ejemplo, los valores $f(x_i)$ pueden provenir de medidas físicas o de algún cálculo que no puede expresarse en una forma simple analíticamente. En esta práctica analizaremos, solamente, el caso en el que la función $f(x)$ es un polinomio, denominado el problema de interpolación polinómica. Joseph-Louis de Lagrange publicó en 1795 una forma de presentar el polinomio que interpola un conjunto de puntos dado.[1]

Esta práctica tiene un doble objetivo, por un lado resolver teóricamente y computacionalmente este problema, usando los conceptos y resultados vistos en clase de teoría y, por otro lado entender y analizar el siguiente programa de Sage, el cual implementa el célebre Polinomio Interpolación de Lagrange:

```
-----------------------------------------------------------------------------------------------------

def PolVan(KK,x,X,Y):
    #Matriz de Vandermonde
    V = matrix(KK, len(X), [[z**i for i in range(len(X))] for z in X])
    #Resolver sistema
    p=V.solve_right(vector(Y))*vector([x^i for i in range(len(X))])
    return p

-----------------------------------------------------------------------------------------------------
```

La tarea está dividida en cuatro etapas, a saber:

1. Presentaremos una breve introducción a la programación con Sage, entre otros motivos, para poder manipular los comandos utilizados en el programa de arriba.
2. Resolveremos el problema para un caso concreto de un polinomio de grado 3.
3. Resolveremos el problema en el caso general.
4. Finalmente, analizaremos la función $PolVan$ del programa.

[1] Más adelante resolveremos, también, con técnicas de álgebra lineal un problema relacionado, el denominado método de los mínimos cuadrados.

1. Introducción a la programación en SAGE

El lenguaje de programación de SAGE por defecto es PYTHON. Se trata de un lenguaje de programación interpretado que soporta orientación a objetos, programación imperativa y funcional.

- **Variables**

  Una variable es un identificador en el que se guarda (almacena) un *dato* o más generalmente un *objeto* (colección de datos). No es necesario decir a priori qué tipo de objeto es lo que vamos a guardar en la variable.

```
--------------------------------------------------------------------------------
b2=12.3;b2
--------------------------------------------------------------------------------
```

  SAGE maneja las variables de modo dinámico, por lo que, si queremos podemos guardar otro objeto en la misma variable:

```
--------------------------------------------------------------------------------
b2="hola mundo"; b2
--------------------------------------------------------------------------------
```

  Podemos operar con las variables, de la misma forma que con el objeto guardado:

```
--------------------------------------------------------------------------------
b2="hola mundo" + "y adios" ; b2
--------------------------------------------------------------------------------
```

  Si queremos borrar una variable, podemos hacerlo con el comando DEL:

```
--------------------------------------------------------------------------------
del b2; b2
--------------------------------------------------------------------------------
```

  También podemos asignar varias variables al mismo tiempo:

```
--------------------------------------------------------------------------------
a,b=1,2; a,b
--------------------------------------------------------------------------------
```

- **Variables en sentido matemático**

  La única que cumple esto es la letra "x".

```
--------------------------------------------------------------------------------
(x-2)**2+5*x^5-1/4
--------------------------------------------------------------------------------
```

  Si necesitamos más variables, utilizamos el comando VAR:

```
--------------------------------------------------------------------------------
var('y'); x-y**3+x*y-6
--------------------------------------------------------------------------------
```

  También podemos almacenar expresiones matemáticas:

```
--------------------------------------------------------------------------------
f= x-y**3+x-5*y+2; p
--------------------------------------------------------------------------------
```

  Podemos evaluar la función matemática f

```
--------------------------------------------------------------------------------
f(x=1,y=0)
--------------------------------------------------------------------------------
```

  También podemos operar simbólicamente con ellas:

```
--------------------------------------------------------------------------------
f.diff(y)
--------------------------------------------------------------------------------
```

- **Programación funcional**

  Para definir un programa como función en SAGE/PYTHON se usa el comando DEF, seguido del nombre de la función, los argumentos (variables) de entrada entres paréntesis y dos puntos. En la siguientes líneas se implementa la función. La identación (sangrado) es muy importante: todo lo que este con el mismo sangrado es el cuerpo de la función. Para terminar, usamos el comando RETURN para devolver el valor de la función.

```
-------------------------------------------------------------------------
def suma_cuadrados(x,y):
        x=x+1; y=y-1
        s= x**2+y**2
        return s
-------------------------------------------------------------------------
```

  Podemos usar la función de forma similar a cualquier función matemática:

```
-------------------------------------------------------------------------
suma_cuadrados(2,4)
-------------------------------------------------------------------------
```

- **Listas**

  Una *lista* es una colección ordenada de objetos. Para crear una lista se pone entre corchetes la lista de los elementos, separados por comas.

```
-------------------------------------------------------------------------
L=[1,2,[3,5.4], 1/7, 8,-1]
-------------------------------------------------------------------------
```

  Igual que el objeto *vector* estudiado en la Práctica 2, para acceder a un elemento se pone el nombre de la lista, seguido de corchetes y entre corchetes el número de elemento al que queremos acceder. El primer elemento se numera con 0.

```
-------------------------------------------------------------------------
L[2]
-------------------------------------------------------------------------
```

  También podemos comenzar contando desde le último elemento:

```
-------------------------------------------------------------------------
L[-2]
-------------------------------------------------------------------------
```

  Podemos seleccionar un segmento de la lista con el operador: nombreDeLista [primerElemento:ultimoElemento].

```
-------------------------------------------------------------------------
L[1:3]
-------------------------------------------------------------------------
```

  Podemos modificar los elementos:

```
-------------------------------------------------------------------------
L[1]="hola"; L
-------------------------------------------------------------------------
```

  Podemos añadir un elemento al final de la lista:

```
-------------------------------------------------------------------------
L.append(9999); L
-------------------------------------------------------------------------
```

  El comando/función LEN aplicado a una lista, devuelve el número de elementos de la lista:

```
-------------------------------------------------------------------------
len(L)
-------------------------------------------------------------------------
```

- **Tuplas**

  Una *tupla*, también, es una colección ordenada de objetos. Para crear una tupla se pone entre paréntesis la lista de los elementos, separados por comas. O simplemente se escriben los elementos separados por comas.

```
-------------------------------------------------------------------------------
F=1,-3.5, "hola", [6,2],(3,5),2/9; F
-------------------------------------------------------------------------------
```

Como en las listas, también, podemos acceder a un elemento. Se pone el nombre de la tupla, seguido de corchetes y entre corchetes el número de elemento al que queremos acceder. El primer elemento se numera con 0. También, el comando LEN aplicado a una tupla devuelve el numero de elementos.

```
-------------------------------------------------------------------------------
F[4], len(F)
-------------------------------------------------------------------------------
```

Una tupla es una lista inmutable. Una tupla no puede modificarse de ningún modo después de su creación. Podemos decir, que el objeto *vector* es una tupla, donde podemos modificar sus elementos/componentes, pero no el número de ellos.

- **Creación de listas**

  SAGE permite crear de modo sencillo una lista de elementos, también para trabajar de modo simbólico.

```
-------------------------------------------------------------------------------
Li1= [x^2,3*x^2/2..2*4*x^2]; Li1
-------------------------------------------------------------------------------
```

El comando/función RANGE evaluada en un natural n devuelve la lista de los n primeros naturales, desde el 0 hasta el $n-1$:

```
-------------------------------------------------------------------------------
range(6)
-------------------------------------------------------------------------------
```

- **Operando con listas**

  SAGE soporta varias operaciones sobre listas. Sumar los elementos de una lista, agrupar dos listas en una uniendo el elemento k de la primera con el k de la segunda, y por último el comando MAP que aplica una función a cada elemento de una lista.

```
-------------------------------------------------------------------------------
oper1=sum([1..10])
oper2=zip([1,2,3,4],['a','b','c','d'] )
oper3=map( cos, [0, pi/4, pi/2, 3*pi/4, pi] )
oper1;oper2; oper3
-------------------------------------------------------------------------------
```

- **Estructuras de control**

  Las estructura de control IF permite seleccionar el código o sentencia a ejecutar y las WHILE o FOR hacen que una sentencia se ejecute repetidamente (iteración).

```
-------------------------------------------------------------------------------
n=33;
if n%3 == 0:
  print n/3
else:
   print (n+1)/3
-------------------------------------------------------------------------------
```

Todas las sentencias que queremos que se ejecuten han de tener el mismo sangrado.

Para repetir una sentencia un número indeterminado de veces, podemos usar un bucle WHILE, con sintaxis:

```
-------------------------------------------------------------------------------
i=-1;
while i < 3:
   i=i*i-1;
   print i
-------------------------------------------------------------------------------
```

El bucle FOR repite una instrucción para cada elemento de una lista:

```
-------------------------------------------------------------------------------
for i in range(4):
   print i^2
-------------------------------------------------------------------------------
```

Finalmente, PYTHON permite crear listas rapidamente a partir de otras.

```
[k**2 for k in range(6)]
```

2. El polinomio de interpolación de grado 3

Supongamos que queremos encontrar un polinomio $p(x)$ de grado menor o igual que tres:

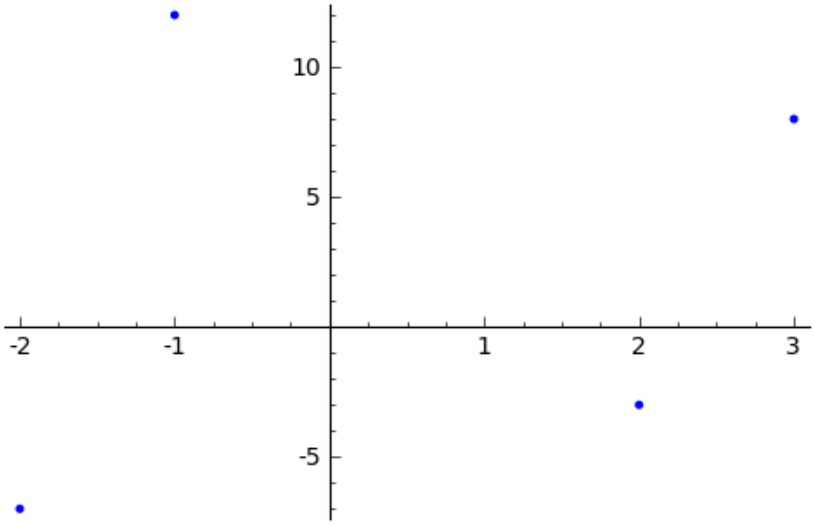
$$p(x) = c_3x^3 + c_2x^2 + c_1x + c_0$$

que interpole los puntos siguientes puntos $(-2,-7),(-1,12),(2,-3),(3,8)$, es decir,

$$p(-2) = -7, \quad p(-1) = 12, \quad p(2) = -3, \quad p(3) = 8$$

Utilizamos los comandos PLOT y SHOW para mostrar los puntos en el plano:

```
puntos = [[-2,-7],[-1,12],[2,-3],[3,8]]
pp = list_plot(puntos)
show(pp)
```



Para computar los coeficientes c_3, c_2, c_1, c_0 del polinomio $p(x) = c_3x^3 + c_2x^2 + c_1x + c_0$, planteamos el siguiente sistema de 4 ecuaciones lineales con las cuatro incognitas c_3, c_2, c_1, c_0:

$$\left\{\begin{array}{rcl} c_0 + (-2)c_1 + (-2)^2c_2 + (-2)^3c_3 &=& -7 \\ c_0 + (-1)c_1 + (-1)^2c_2 + (-1)^3c_3 &=& 12 \\ c_0 + (+2)c_1 + (+2)^2c_2 + (+2)^3c_3 &=& -3 \\ c_0 + (+3)c_1 + (+3)^2c_2 + (+3)^3c_3 &=& 8 \end{array}\right. \tag{1}$$

La matriz del sistema es:

$$V = \begin{pmatrix} 1 & -2 & (-2)^2 & (-2)^3 \\ 1 & (-1) & (-1)^2 & (-1)^3 \\ 1 & 2 & 2^2 & 2^3 \\ 1 & 3 & 3^2 & 3^3 \end{pmatrix} = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{pmatrix}$$

Resolvemos el sistema de ecuaciones con SAGE:

```
V=matrix(QQ,4,[[1,-2,4,-8],[1,-1,1,-1],[1,2,4,8],[1,3,9,27]]);
sol=V.solve_right(vector([-7,12,-3,8]))
```

Obteniendo: $sol = (11, -7, -4, 2) = (c_0, c_1, c_2, c_3)$, es decir, el polinomio $p(x)$ que interpola los puntos es

$$p(x) = 2x^3 - 4x^2 - 7x + 11$$

Que lo podemos construir con el siguiente código de SAGE:

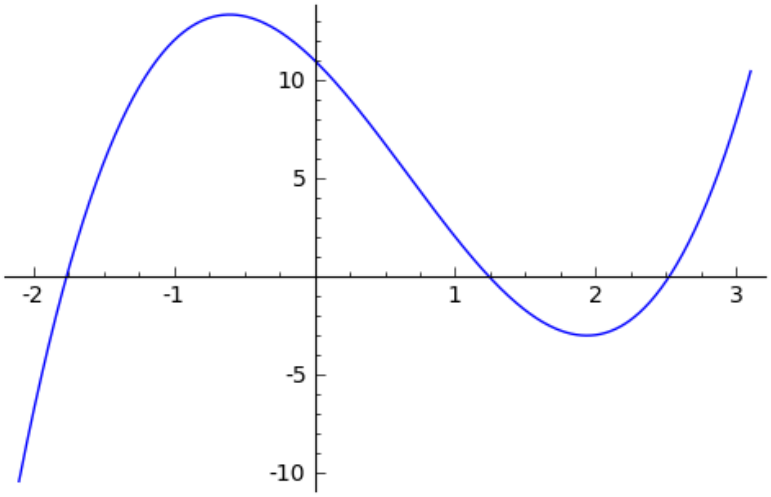
```
-------------------------------------------------
p=sol*vector([x**i for i in range(4)])
-------------------------------------------------
```

Si denotamos $X = (-2, -1, 2, 3)$ e $Y = (-7, 12, -3, 8)$, el polinomio p se construye con el siguiente simple código:

```
-------------------------------------------------
X=(-2,-1,2,3)
Y=(-7,12,-3,8)
V=matrix(QQ,4,[[z^i for i in range(len(X))] for z in X])
p=V.solve_right(vector(Y))*vector([x**i for i in range(len(X))])
-------------------------------------------------
```

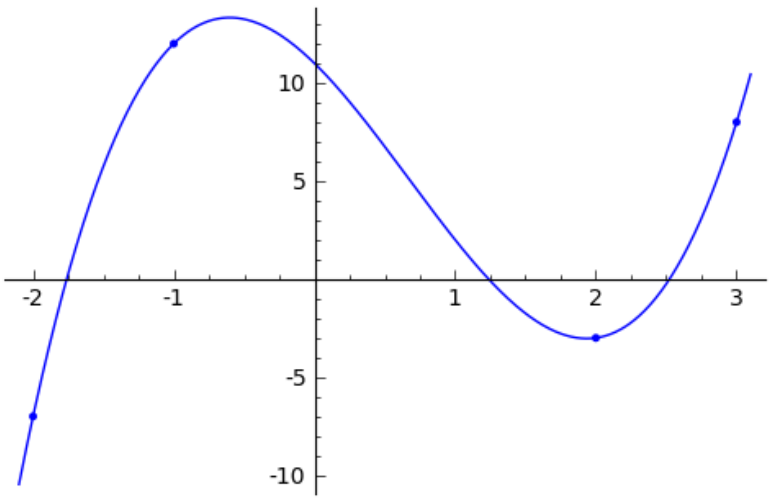
Utilzamos en comando PLOT para dibujar la grafica del polinomio:

```
-------------------------------------------------
grafica=plot(p,-2.1,3.1)
show(grafica)
-------------------------------------------------
```



Ahora mostramos los puntos interpolados y la grafica del polinomio:

```
-------------------------------------------------
show(grafica+pp)
-------------------------------------------------
```



3. **La matriz de Vandermonde y el polinomio de interpolación. El programa** $PolVan$

4. **EJERCICIOS**

   a) Encontrar el polinomio de interpolación para los puntos: $[(3,1),(1,3),(-1,5),(4/5,6),(0,-2)]$

   b) Dada la matriz A de la aplicación lineal $f:\mathbb{R}^5 \to \mathbb{R}^3$, con respecto a las bases canónicas:

$$A = \begin{pmatrix} 5 & -1 & 1 & 2 & -3 \\ 2 & -1 & 0 & 1 & -1 \\ 1 & 1 & 1 & 0 & -1 \end{pmatrix}$$

   - Calcular $f(v)$ donde $v = (2019/2018, -4, 12, -456, -2017)$
   - Bases de $Ker(f)$ e $Im(f)$
   - Encontrar $f^{-1}(1,2,3)$, es decir, el subconjunto de vectores $v = (x,y,z,s,t)$ de $\mathbb{R}^5$ tal que $f(v) = (1,-2,5)$.