Facultad de Ciencias Sociales y Jurídicas

Grado en Administración y Dirección de Empresas

Universidad de Jaén

Facultad de Ciencias Sociales y Jurídicas

Trabajo Fin de Grado

PRÁCTICAS CON ORDENADOR DE MATEMÁTICAS CON SOFTWARE ALTERNATIVO A MATHEMATICA EN EL GRADO DE ADE.

Alumno: Baltanás Palomar, Carlos

06-2021

ÍNDICE

1. RESUMEN/ABSTRACT

RESUMEN: Actualmente, en la Universidad de Jaén, las prácticas de la asignatura Matemáticas 1 están realizadas con el software no gratuito Mathematica, accesible a todo el colectivo universitario.

En este contexto, surgen diferentes programas como Jupyter, el cual es gratuito y donde se analizan datos como los ejercicios de Matemáticas 1. Con esta herramienta se han resuelto todos los ejercicios virtualmente y se ha visto que podemos llegar a las mismas conclusiones mediante un lenguaje matemático y gráfico.

Además, con las referencias bibliográficas y la ayuda de los anexos, hemos entendido la derivación e integración de funciones, matrices, sistemas lineales, entre otros.

Primeramente, he explicado cómo instalar y usar el programa para continuar desarrollando las primeras operaciones con éste. Luego, en cada práctica, he resuelto los ejercicios con las librerías de Sympy y math y representar las funciones que he realizado con MathPlot.

ABSTRACT: Nowadays, in the University of Jaén, the practices of the subject Mathematics 1 are made with the software *Mathematica,* which is non free but accessible by teachers and students.

In this context, different programmes come up like *Jupyter,* which is free and where datums are analysed as the exercises of *Mathematics 1.* With this tool, it has been resolved all the exercises like the report shows virtually, through which it has shown we can arrive to the same conclusions by a mathematic and graphical language.

Moreover, with bibliographic references and annexes' help, it has been understood: derivation and incorporation of functions, matrices, linear systems…

Firstly, I explained how to install and use the programme to continue expounding on the first calculations with it. Then, in each practice, I resolved the exercises with the Sympy and Math libraries and to display as a graph the functions I have worked with MathPlot.

2.INTRODUCCIÓN

Este trabajo se sustenta en las prácticas de la asignatura Matemáticas I del grado en ADE que actualmente se realizan con Mathematica. En el TFG se abordan con un software

alternativo (Jupyter, que es gratuito) para comparar con el que actualmente se utiliza y observar posibles ventajas e inconvenientes.

Una de las posibles ventajas de este programa como hemos comentado anteriormente sería la de que es gratuito y la rapidez con la que realiza los cálculos y los inconvenientes serían que no es un programa fácil de utilizar si no tienes un conocimiento inicial de Jupyter y que hemos encontrado algunas limitaciones a la hora de encontrar la solución a algunos ejercicios.

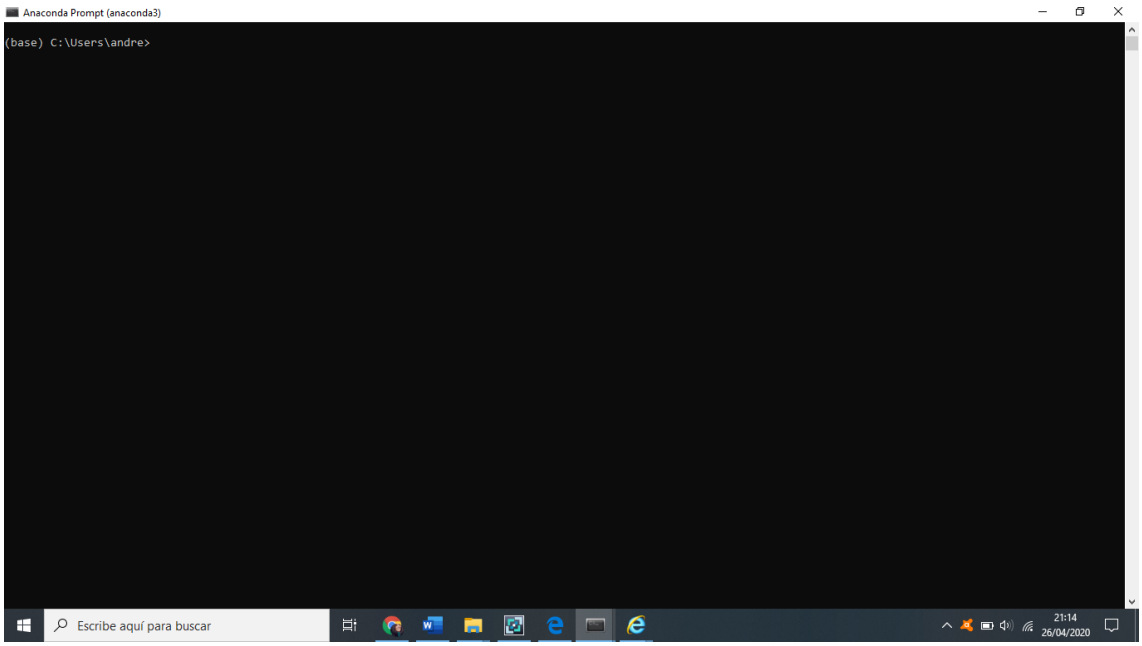
3. PRÁCTICA 0 “INTRODUCCIÓN A JUPYTER”

3.1¿QUÉ ES JUPYTER?

Jupyter es un software gratuito, que se utiliza con un lenguaje de programación llamado IPython [1], [2], [3] ,aunque también soporta más lenguajes, fue creado por Fernando Pérez, el cual lo creó con la intención de ser utilizado por matemáticos e ingenieros [4].

3.2¿CÓMO SE INSTALA?

Para instalar Jupyter lo primero que tenemos que hacer es instalar el programa Anaconda, poniendo en Google instalar Anaconda, hacemos click en https://www.anaconda.com/products/individual [5]. Una vez dentro de Anaconda bajamos hasta el final de la página, donde nos pondrá instaladores de Anaconda.
En la opción de ventanas nos vamos a Python 3.7 e instalamos el gráfico correspondiente al ordenador de cada persona. Este es el lenguaje necesario para utilizar Jupyter.
Ya instalado Python, para comenzar a utilizar Jupyter, nos tenemos que ir a Inicio y poner en el buscador Anaconda, hacemos click y nos saldrá una ventana en negro como esta:

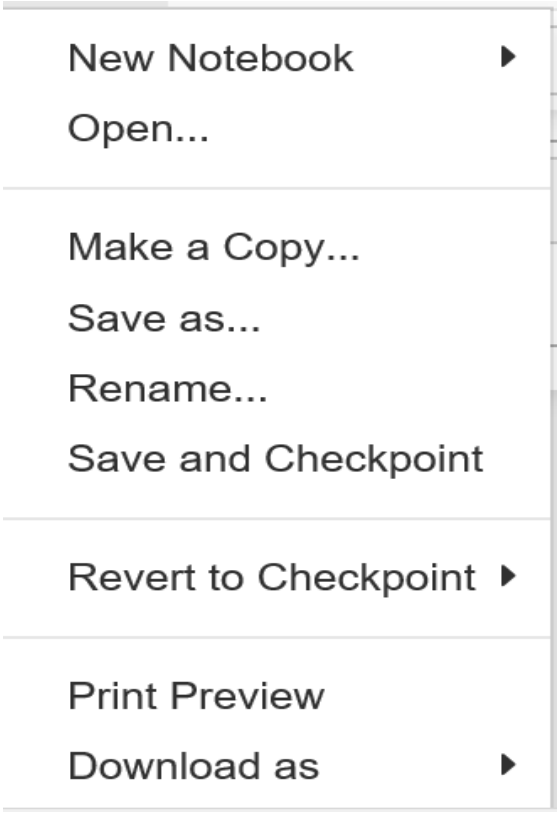


Aquí escribimos Jupyter notebook y pulsamos Enter, nos esperamos un poco y ya nos sale el navegador de Jupyter listo para ser utilizado.

3.3.TRABAJANDO CON JUPYTER

Una vez abierto Jupyter, podemos ver que hay una barra de herramientas con las siguientes opciones: File, Edit, View, Insert, Cell, Kernel y Help [6]. Comentamos las opciones más destacadas.

3.3.1.File



Lo primero que nos aparece es la opción New Notebook, que nos servirá para abrir un nuevo documento en blanco.

La opción Open es para abrir un documento ya creado anteriormente.

- **Make a Copy:** para hacer una copia del documento que se está usando.
- **Save as:** guardar el documento.
- **Rename:** cambiar de título.
- **Print Preview:** imprimir el documento.
- **Download as:** descargar el archivo con otro formato distinto.

3.3.2.Edit

Aquí los más utilizados son:

- **Cut Cells:** para cortar la celda seleccionada.
- **Copy Cells:** para copiar la celda seleccionada.
- **Paste Cells** : para pegar la celda seleccionada.
- **Delete Cells:** borrar la celda seleccionada.
- **Undo:** Deshacer la última operación realizada.
- **Move Cell Up:** Mover la celda hacia arriba.
- **Move Cell Down:** Mover la celda hacia abajo.

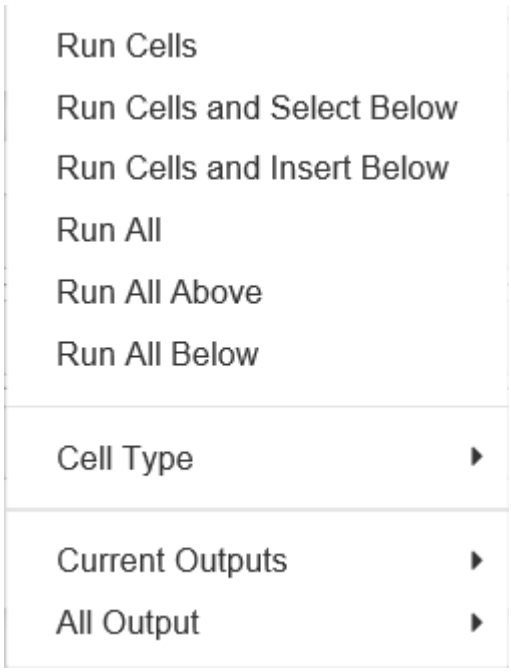
3.3.3. View

Se utiliza para esconder la barra de tareas o no.

3.3.4. Insert

Inserta el número de celdas que quieras.

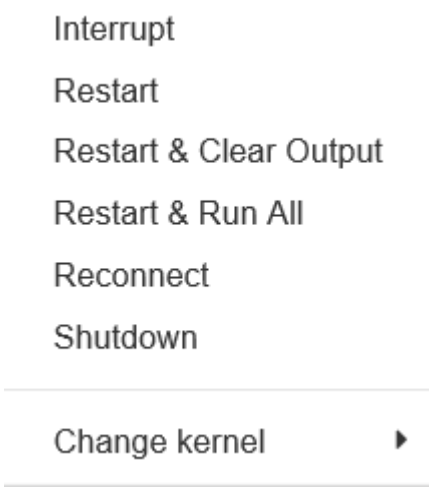
3.3.5. Cell



Las más utilizadas son:

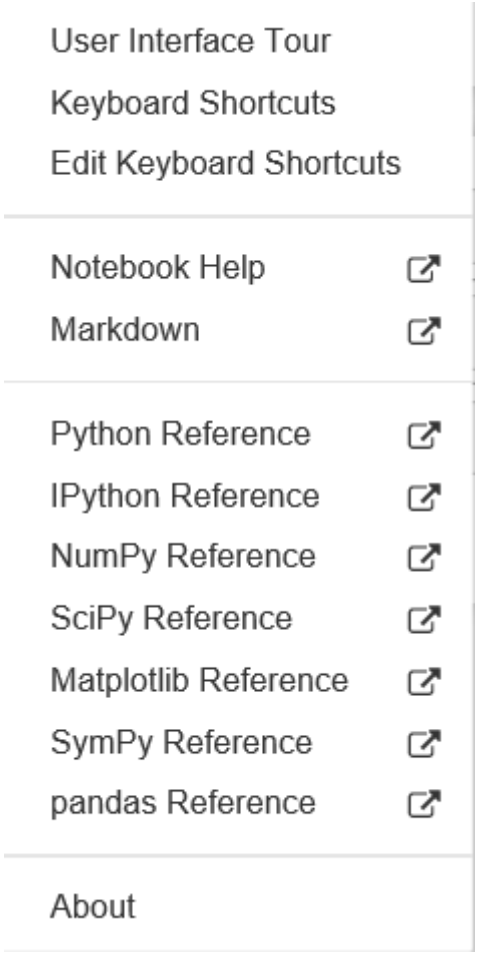
- **Run Cells:** para calcular la celda seleccionada, aunque también se puede calcular la celda en el teclado, con las teclas, ctrl + Enter.
- **Run All:** para calcular todas las celdas.
- **Cell Type :** para cambiar el tipo de celda, si lo queremos en formato texto o código.

3.3.6. Kernel



Esta opción sirve para reiniciar, interrumpir el programa o cambiar de lenguaje de programación, nosotros vamos a utilizar Python 3.

3.3.7. Help



Aquí podemos observar las distintas ayudas que nos ofrece el programa, tanto para los distintos lenguajes que se pueden utilizar, como para el programa en sí, en la pestaña Notebook Help.

También nos ofrece el programa un Tour para que nos familiaricemos con él.

3.4.“PRIMERAS OPERACIONES CON JUPYTER”.

En primer lugar, para realizar los ejercicios de esta práctica hay que insertar la función: from math import *, para que el programa reconozca los símbolos de las funciones exponenciales, logarítmicas, trigonométricas etc.[2], [3].

Para que Jupyter nos proporcione el resultado de una operación con un número determinado de dígitos hay que introducir la función: “{0:.número de dígitos que queramos f}”, por ejemplo, si queremos que nos dé, el resultado de una operación con 3 dígitos sería: “{0:.3f}”

Jupyter tiene incorporado la mayoría de las funciones matemáticas que son las siguientes:

- Funciones trigonométricas
  Sin(x), Cos(x), Tan(x)
- Funciones exponenciales y logarítmicas
  Log(x)
  Exp(e)
- Funciones radicales
  Sqrt(x): Raiz cuadrada de x
- Polinomios y funciones racionales
  x** para elevar un número a un exponente.
  un ejemplo sería x**2+3x/2

Otra cuestión importante es a la hora de definir una función a través del comando def en el cual ponemos nombre a la función, f(x) y a continuación return.

3.5.EJERCICIOS INTRODUCCIÓN A JUPYTER: PRÁCTICA 0

In [1]: from math import *

EJERCICIO 1

Calcula la raíz cuadrada de 2, 3, 4 y 5, con dos, tres, cuatro y cinco decimales, respectivamente.

In [2]: a = "{0:.2f}".format(sqrt(2))

b = "{0:.3f}".format(sqrt(3))

c = "{0:.4f}".format(sqrt(4))

d = "{0:.5f}".format(sqrt(5))

print(a,b,c,d)

Out[2]:1.41 1.732 2.0000 2.23607

EJERCICIO 2

Calcula el valor de 3 más la exponencial de la raíz cuadrada de *e*, con 10 decimales.

In [3]: "{0:.10f}".format(3+exp(sqrt(e)))

Out[3]: '8.2003257648'

EJERCICIO 3

Calcula con una aproximación de 30 dígitos las expresiones:

A) sen 45°

In [4]: "{0:.30f}".format(sin(radians(45)))

Out[4]: '0.707106781186547572737310929369'

B) $(\log_2 3)^7$.

In [5]: "{0:.30f}".format(log(3,2)**7)

Out[5]: '25.126554557199280992563217296265'

C) $\sqrt{\frac{2}{sen\frac{7\pi}{3}}}$

In [6]: "{0:.30f}".format(sqrt(2/sin(7*pi/3)))

Out[6]: '1.519671371303185303247573756380'

EJERCICIO 4

Calcula el coseno y seno, con 5 cifras decimales, de π/7

In [7]: a = "{0:.5f}".format(cos(pi/7))

b = "{0:.5f}".format(sin(pi/7))

print(a,b)

Out[7]:0.90097 0.43388

EJERCICIO 5

Evalúa las siguientes funciones en 2 y 0.

A) cos(x) sen(x) – $\sqrt{x}$.

In [8]: x = 2

a = cos(x)*sin(x)-sqrt(x)

a

Out[8]: -1.7926148100270594

In [9]: x = 0

a = cos(x)*sin(x)-sqrt(x)

a

Out[9]: 0.0

B) $x + \frac{\cos(x)}{2}$

In [10]: x = 2

b = x+cos(x)/2

b

Out[10]: 1.7919265817264287

In [11]: x = 0

b = x+cos(x)/2

b

Out[11]: 0.5

C) $1 + e^{x+2}$

In [12]: x = 2

c = 1+e**(x+2)

c

Out[12]: 55.59815003314423

In [13]: x = 0

c = 1+e**(x+2)

c

Out[13]: 8.389056098930649

EJERCICIO 6

Asigna a las variables x e y valores respectivos 3 y 4 y realiza las siguientes operaciones:

A) x/y - 1

In [14]: x = 3

y = 4

x/y-1

Out[14]: -0.25

B) x^2 /(y+2)

In [15]: x = 3

y = 4

x**2/(y+2)

Out[15]: 1.5

C) $(\sqrt{\frac{x}{y} + 2x^2})^{3y}$

In [16]: x = 3

y = 4

(sqrt(x/y+2*x**2)**(3*y))

Out[16]: 43451786.04125982

D) $(2x - y + x y)^3$

In [17]:x = 3

y = 4

(2*x-y+x*y)**3

Out[17]: 2744

EJERCICIO 7

Calcula 7/12[1/13 + 5/32 : (15 – 2/3)]

In [18]: 7/12*(1/13+5/32/(15-2/3))

Out[18]: 0.05123080649970186

4.PRÁCTICA 1 "FUNCIONES"

Para resolver esta práctica hay que insertar los comandos siguientes:

- From math import * que se utiliza por si hubiera que insertar algún raíz o coseno o seno etc..
- From sympy import*, para realizar los ejercicios de límites.
- From matplotlib import pyplot: para representar las funciones gráficamente
  Los límites se resuelven con la función limit(la función, x, hacia donde tiende la x).[7],[8].
  Para representar las funciones lo primero que hay que hacer es introducir los comandos anteriormente mencionados y después definir la función con def y return e introducir los comandos para darle valores a la x e y desde donde hasta donde queremos representar la función, así como poner el color que queramos a los distintos ejes y a la gráfica.[9],[10],[11],[12],[13]
  Para definir una función a trozos hay que utilizar el comando def que ya lo hemos comentado antes y como novedad el comando if para decirle al programa las restricciones que tiene la función.
  Para saber las raíces de una función utilizamos la función from scipy.optimize import fsolve y el comando linspace para darle los valores a la función.
  Por último, para resolver una ecuación utilizamos la función solve.

4.1.EJERCICIOS PRÁCTICA 1: FUNCIONES

In [1]: from math import *

```
from sympy import*

from sympy.interactive import printing

x = Symbol('x')
```

EJERCICIO 1 Calcula los siguientes límites.

A) $\lim_{x\to 0+} 5^{1/x}$

```
In [2]: limit(5**(1/x),x,0)
```

Out[2]:∞

B) $\lim_{x\to 0-} 5^{1/x}$

```
In [3]: limit(5**(1/x), x, 0,dir='-')
```

Out[3]:0

C) $\lim_{x\to 2} \frac{x^2-4}{x-2}$

```
In [4]: limit((x**2-4)/(x-2),x,2)
```

Out[4]:4

D) $\lim_{x\to +\infty} \sqrt{x^2-x}-(x+1)$

```
In [5]: limit((x**2-x)**(1/2)-(x+1),x,S.Infinity)
```

Out[5]:−1.5

E) $\lim_{x\to +\infty} \left(1+\frac{2^{-x}}{x}\right)^x$

```
In [6]: limit((1+2**-x/x)**x,x,S.Infinity)
```

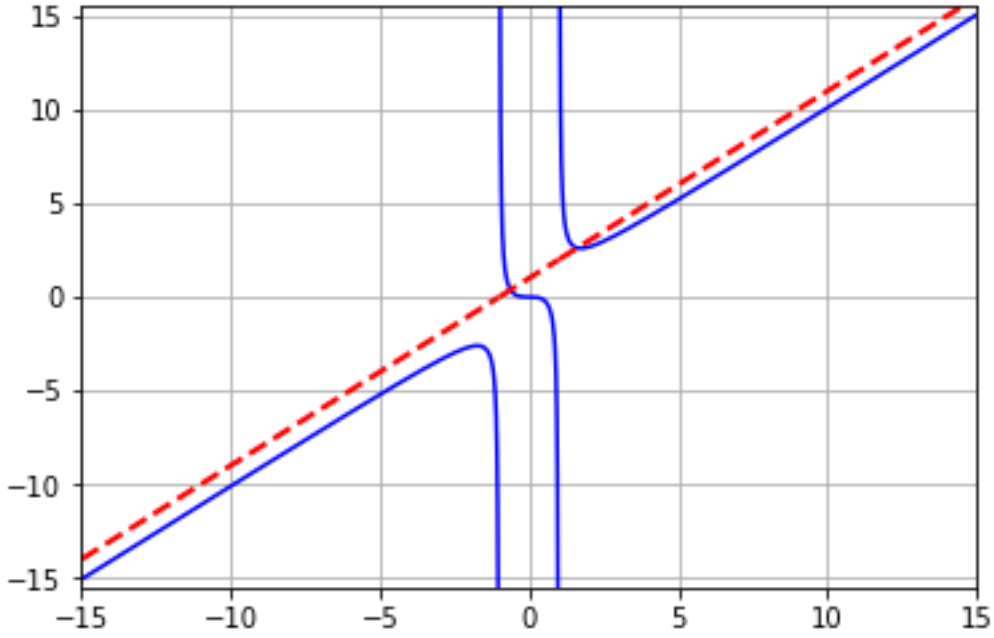
Out[6]:1

EJERCICIO 2 Representa gráficamente las siguientes funciones:

A) $f(x)=\frac{x^3}{x^2-1}$

```
In [7]: %matplotlib inline
import matplotlib.pyplot as  plt
```

```
import numpy as np
f = lambda x: pow(x, 3) / (pow(x, 2) - 1)
fig, ax = plt.subplots()
x = np.linspace(-15.0, 15.0, 10000)
pos = np.where(np.abs(np.diff(f(x))) >= 10.0)[0]
x = np.insert(x, pos, np.nan)
ax.axis([x[0], x[-1], -15.5, 15.5])
ax.plot(x, f(x), color='b', linestyle='-', lw=1.5)
ax.plot(x, x + 1.0, color='r', linestyle='--', lw=2.0)
ax.grid('on')
plt.show()
```



B) $\boldsymbol{g(x) = \frac{x}{\sqrt{x^2+2}}}$

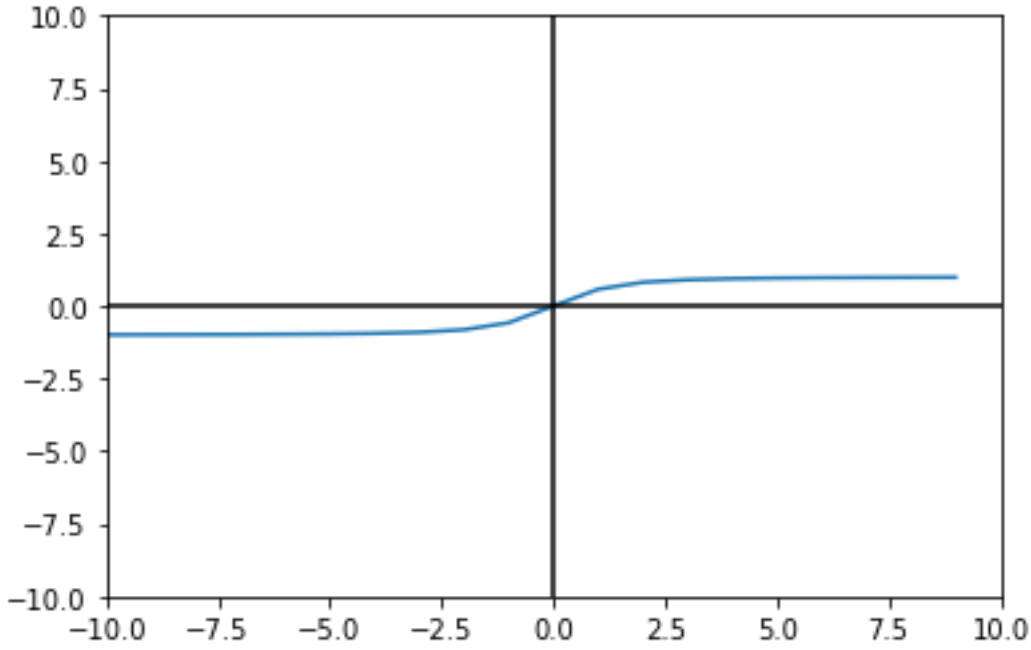
```
In [8]: def f(x):
  return x/(sqrt(x**2+2))
x = range(-10,10)
pyplot.plot(x,[f(i) for i in x])
pyplot.axhline(0,color="black")
```

pyplot.axvline(0,color="black")

pyplot.xlim(-10,10)

pyplot.ylim(-10,10)

(-10, 10)

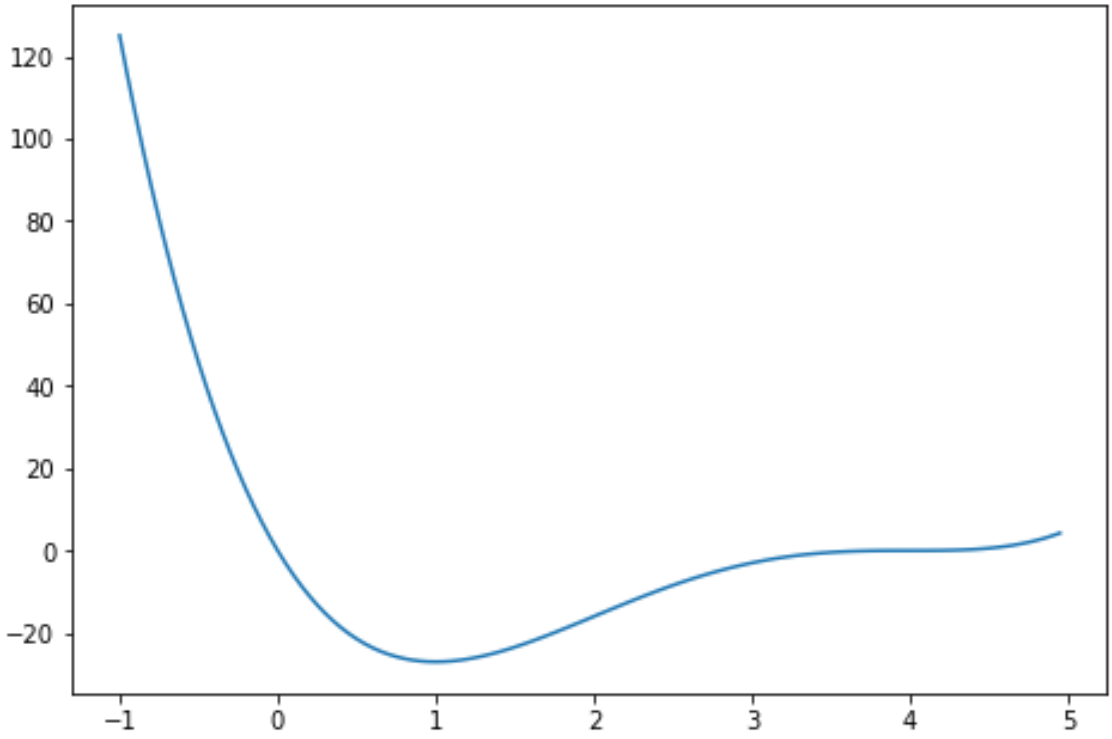


C) $h(x) = x^4 - 12x^3 + 48x^2 - 64x$

```
In [9]: from matplotlib import pyplot as plt
import numpy as np
import math
%matplotlib inline
def f(x):
 return x**4-12*x**3+48*x**2-64*x
x=np.arange(-1, 5,0.05)
y=f(x)
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(x,y)
```

Out[9]:

[<matplotlib.lines.Line2D at 0x18954d45e80>]



EJERCICIO 3 Sea $f: \mathbb{R} \to \mathbb{R}$ la función definida por

$$f(x) = \begin{cases} -x^2 + 2x & x < 3 \\ -x & 3 \leq x \leq 5 \\ 1 & 5 < x \end{cases}$$

A) Calcula $f(-1), f(5), f(6)$.

In [10]: def f(x):

```
    if x<3:
        return -x**2+2*x
    if 3<=x<=5:
        return -x
    if 5<x:
        return 1
```

In[11]: f(-1)

Out[11]: -3

In[12]: f(5)

Out[12]: -5

In[13]: f(6)

Out[13]: 1

B) Estudia la continuidad de f.

In [14]: f(3)

Out[14]: -3

In [15]: limit(-x**2+2*x,x,3)

Out[15]: −3

In [16]: limit(-x,x,3)

Out[16]: −3

Para x=3 la función f(x) es continua.

In [17]: f(5)

Out[17]: -5

In [18]: limit(-x,x,5)

Out[18]: −5

In [19]: limit(1,x,5)

Out[19]: 1

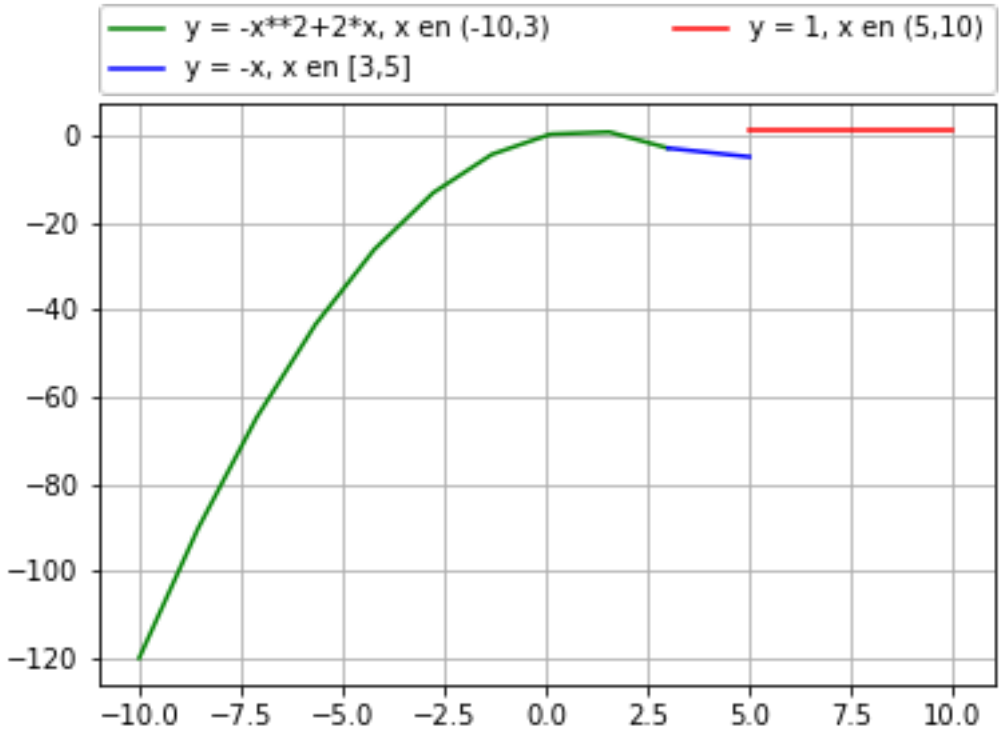
Para x=5 la función f(x) no es continua. Por lo tanto, f(x) es continua en toda la recta salvo en x=5.

C) Representa gráficamente la función.

In [20]: import numpy as np
import matplotlib.pyplot as plt

```
x1 = np.linspace(-10, 3, num=10)
x2 = np.linspace(3, 5, num=3)
x3 = np.linspace(5, 10, num=10)
plt.plot(x1, [-x**2+2*x for x in x1], 'g',  label='y = -x**2+2*x, x en (-10,3)')
plt.plot(x2, [-x for x in x2], 'b',  label='y = -x, x en [3,5]')
plt.plot(x3, [1 for x in x3], 'r',  label='y = 1, x en (5,10)')
plt.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower left',
           ncol=2, mode="expand", borderaxespad=0.)

plt.grid()
plt.show()
```

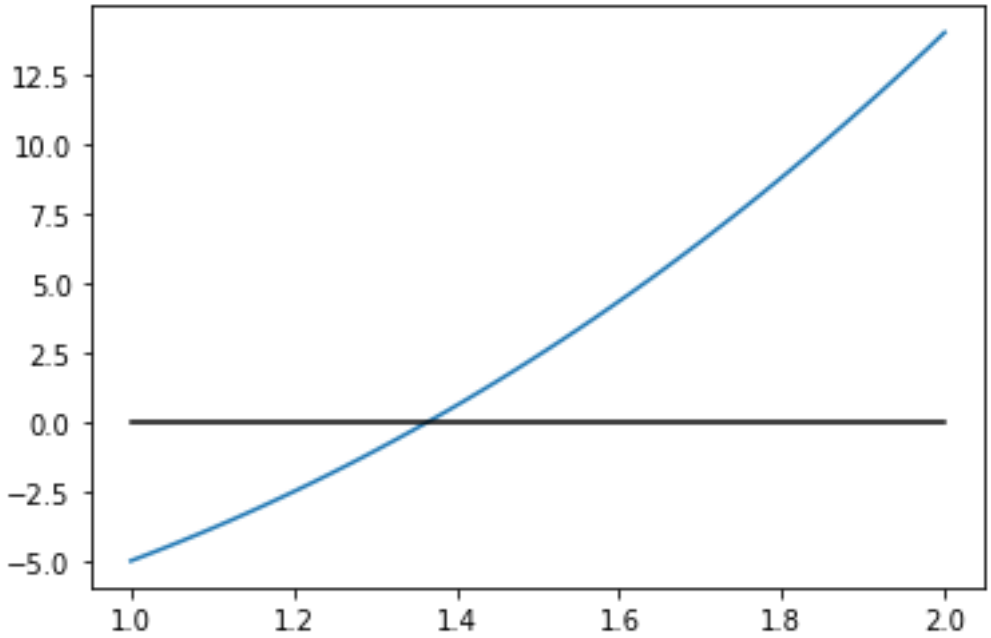


EJERCICIO 4 Comprueba que las siguientes ecuaciones tienen una raíz y calcúlala:

A) $x^3 + 4x^2 = 10$ en el intervalo [1,2].

```
In [21]: from scipy.optimize import fsolve
def f(x):
    return x**3+4*x**2-10
x = np.linspace(1,2)
```

```
f = lambda x : x**3+4*x**2-10
OX = 0*x
plt.plot(x,f(x))
plt.plot(x,OX,'k-')
plt.show()
```

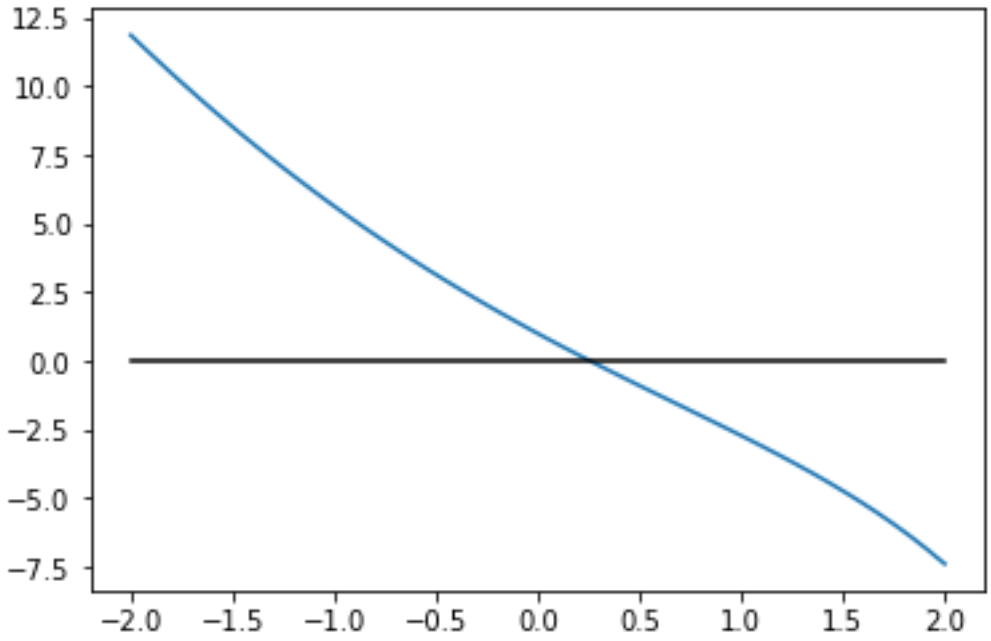



```
In [22]: root = fsolve(f, [1, 2])
root
```

Out[22]: array([1.36523001, 1.36523001])

B) $3x = 2 + x^2 - e^x$ en el intervalo [-2,2].

```
In [23]: x = np.linspace(-2,2)
f = lambda x : x**2-3*x-e**x+2
OX = 0*x
plt.plot(x,f(x))
plt.plot(x,OX,'k-')
plt.show()
```




```
In [24]: def f(x):
    return x**2-3*x-e**x+2
root = fsolve(f,[-2,2])
root
```

Out[24]: array([0.25753029, 0.25753029])

EJERCICIO 5 Sea $f: \mathbb{R} \to \mathbb{R}$ la función definida por

$$f(x) = \begin{cases} x + 2 & x < 1 \\ -2x - 1 & 1 \leq x \end{cases}$$

¿Satisface f las hipótesis del teorema de Bolzano en el intervalo [0,2]?

```
In [25]: def f(x):
    if x<1:
        return x+2
    if 1<=x:
        return -2*x-1
f(0)
```

Out[25]: 2

In [26]: f(2)

Out[26]: -5

Los signos de f en 0 y 2 son distintos. Sin embargo, como vemos, los límites de f a la izquierda y a la derecha de 1 son distintos:

In [27]: x = Symbol('x')

In [28]: limit(x+2,x,1)

Out[28]: 3

In [29]: limit(-2*x-1,x,1)

Out[29]: -3

con lo que f no es continua en x=1, por tanto, no es continua en el intervalo [0,2], y no se satisfacen las hipótesis del Teorema de Bolzano.

EJERCICIO 6 Sean *f*, *g*: $\mathbb{R} \to \mathbb{R}$ las funciones definidas, respectivamente, por

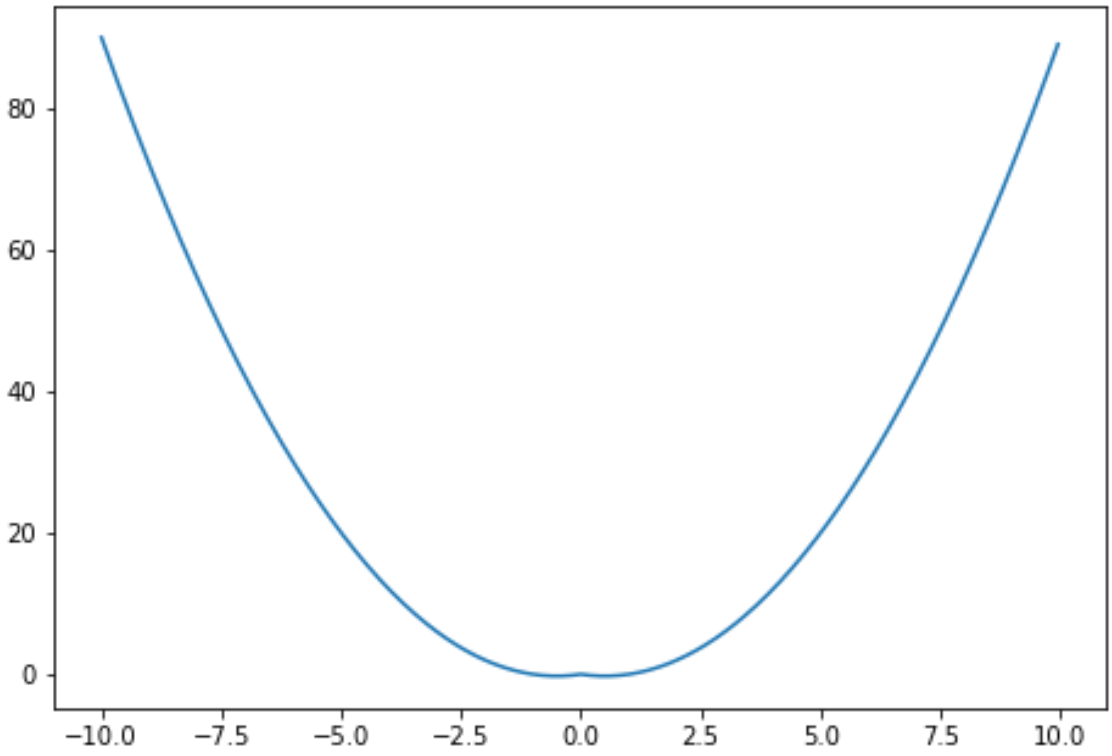
$$f(x) = x(x-1) \quad \text{y} \quad g(x) = |x|$$

A) Halla la función *p*: $\mathbb{R} \to \mathbb{R}$ dada por $p(x) = (f \circ g)(x)$ y represéntala gráficamente

```
In [30]: def p(x):
 return abs(x)*(abs(x)-1)
x=np.arange(-10, 10,0.05)
y=p(x)
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(x,y)
```

Out[30]:

[<matplotlib.lines.Line2D at 0x2289d9848b0>]

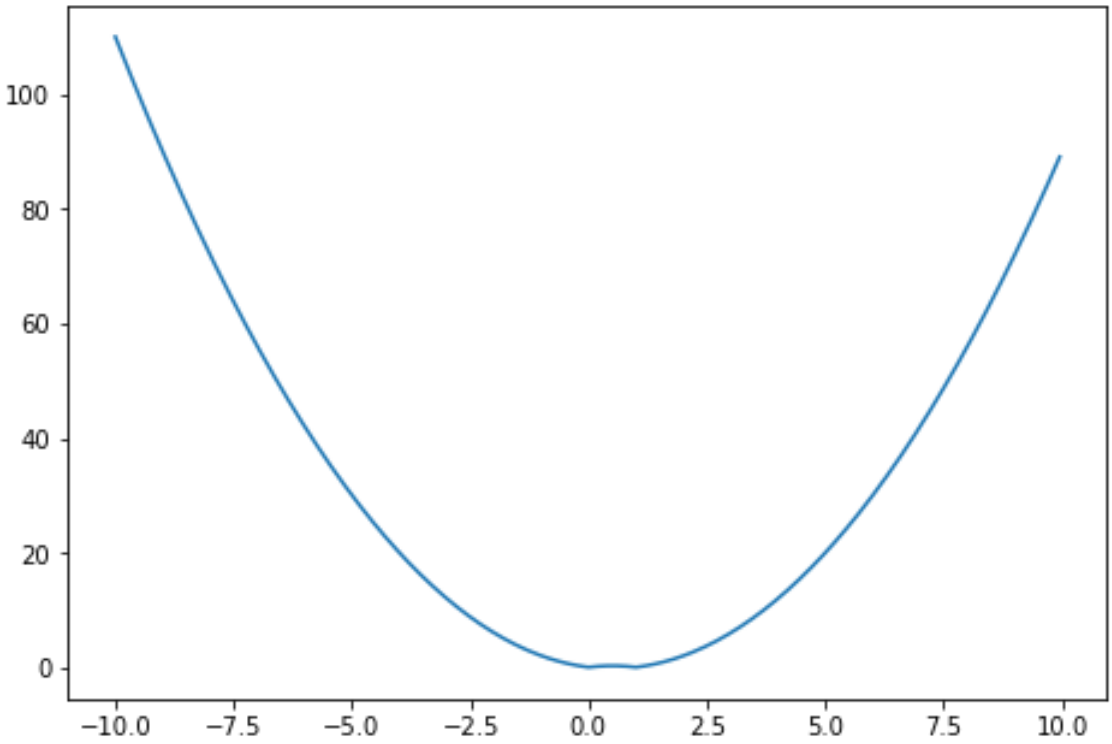


B) Halla la función q : $\mathbb{R} \to \mathbb{R}$ dada por $q(x) = (g\ o\ f\ )\ (x)$ y represéntala gráficamente.

```
In [31]: def q(x):
 return abs(x*(x-1))
x=np.arange(-10, 10,0.05)
y=q(x)
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(x,y)
```

Out[31]:

[<matplotlib.lines.Line2D at 0x2289d9da580>]



EJERCICIO 7 Las ganancias de una empresa, en millones de euros, se ajustan a la función:

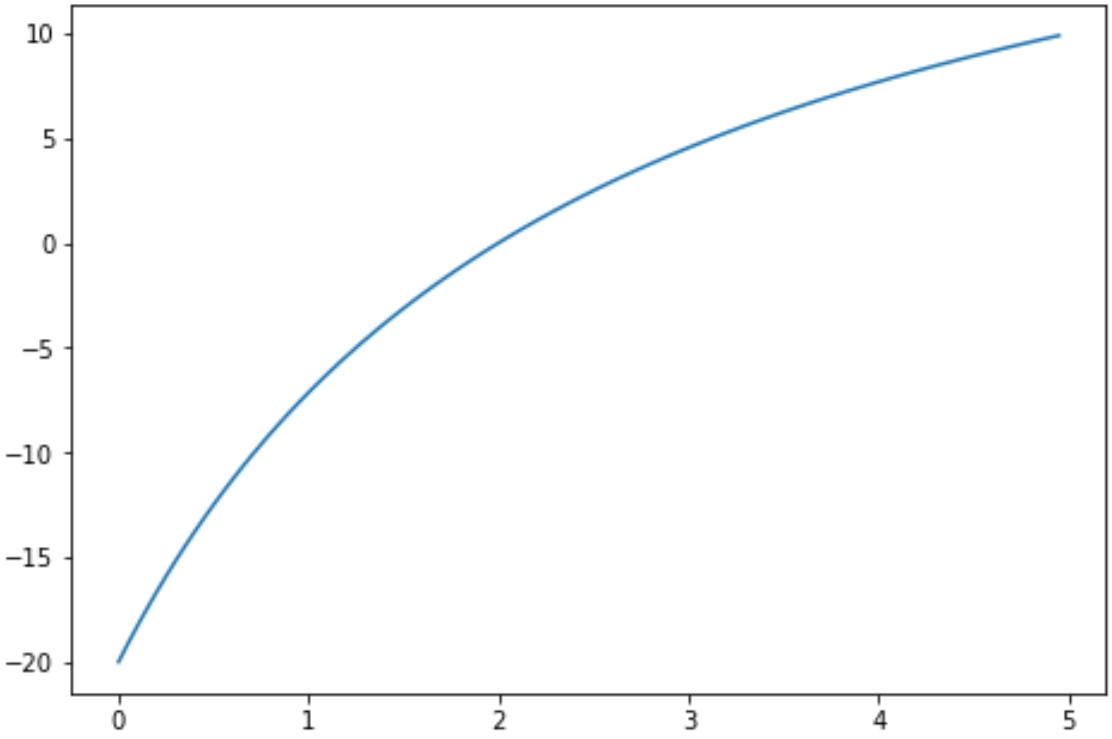
$$f(x) = \frac{50x-100}{2x+5}$$

donde x representa los años de vida de la empresa cuando $x \geq 0$.

A) Representa gráficamente la función f .

```
In [32]: def q(x):
 return (50*x-100)/(2*x+5)
x=np.arange(0, 5,0.05)
y=q(x)
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(x,y)
```

Out[32]:



B) ¿A partir de qué año la empresa deja de tener pérdidas?

In [33]: solve([50*x-100],[x])

La empresa dejará de tener pérdidas a partir del segundo año.

Out[33]:{x: 2}

C) A medida que transcurre el tiempo, ¿están limitados sus beneficios? En caso afirmativo, ¿cuál es su límite?

In [34]: limit((50*x-100)/(2*x+5),x,S.Infinity)

Sus beneficios se estabilizan a partir de los 25 millones de euros.

Out[34]:25

5.PRÁCTICA 2: "DERIVACIÓN DE FUNCIONES"

En esta práctica vamos a estudiar con Jupyter la derivación de funciones [15], a partir de comandos que nos resuelven las derivadas de forma directa, y a través de su representación en una gráfica para calcular sus máximos y sus mínimos, así como los puntos de corte y la recta tangente.

Lo primero que hay que hacer es insertar la librería de Jupyter para realizar derivadas, la cual se realiza a través de, from sympy import, con esto insertamos los comandos que vamos a utilizar para resolver las derivadas, los más comunes son:

- Derivative: comando para resolver las derivadas
- Diff: similar al anterior, también para resolver directamente las derivadas
- Simplify: para simplificar la derivada
- Symbols: introducir símbolos, como por ejemplo la incógnita x, el numero pi etc..

También en esta práctica hemos representado funciones [9], [10], [11], [12], [13], como en las anteriores, en las cuales hemos insertado la librería matplot, la cual sirve en Jupyter para representar funciones, los comandos más usados a través de esta librería son:

- Range: desde donde hasta donde queremos representar la función.
- Pyplot.plot: representar la función en cuestión
- Pyplot.axhline: ponerle el color que queramos al eje de la X
- Pyplot.axvline: ponerle el color que queramos al eje de la Y
- Pyplot.xlim: darle valores al eje de la X
- Pyplot.ylim: darle valores al eje de la Y
- Show(): ver la gráfica de la función

5.1 EJERCICIOS PRÁCTICA 2

EJERCICIO 1 Calcula, a partir de la definición de derivada, las funciones derivadas de

A) $f(x) = \ln(x)$.

In [1]: import sympy

```
from sympy import*
from math import*
x, y, h = sympy.symbols('x h y')
sympy.init_printing(use_unicode=True)
sympy.limit((sympy.ln(x+h)-(sympy.ln(x)))/(h),h,0)
```

Out[1]: 1/x

B) $g(x) = \cos(x)$.

```
In [2]: sympy.limit((sympy.cos(x+h)-sympy.cos(x))/(h),h,0)
```

Out[2]: −sin(x)

EJERCICIO 2 Estudia la derivabilidad de las funciones:

A) $\begin{cases} 1 - x^2 & -3 < x \leq 2 \\ -x - 1 & x > 2 \end{cases}$

```
In [3]: def f(x):
    if -3<x<=2:
        return 1-x**2
    if x>2:
        return -x-1
sympy.limit(1-x**2,x,2)
```

Out[3]: −3

```
In [4]: sympy.limit(-x-1,x,2)
```

Out[4]: −3

ahora vemos el valor de la función en el punto 2

```
In [5]: def f(x):
    return 1-x**2
f(2)
```

f es continua veamos si es derivable

Out[5]: −3

```
In [6]: sympy.diff(1-x**2, x)
```

Out[6]: −2x

```
In [7]: sympy.limit(-2*x,x,2)
```

Out[7]: −4

In [8]: sympy.diff(-x-1,x)

Out[8]: −1

In [9]: sympy.limit(-1,x,2)

la función es derivable en todo su dominio excepto en x=2.

Out[9]: −1

B) $\begin{cases} x^2 sen\frac{1}{x} & , x \neq 0 \\ 0 & , x = 0 \end{cases}$

```
In [10]: def f(x):
    if x!=0:
        return x**2*sin(1/x)
    if x==0:
        return 0

 f(0)
```

Out[10]: 0

In [11]: sympy.limit(x**2*sin(1/x),x,0)

g es continua ahora estudiamos su derivabilidad.

Out[11]: 0

In [12]: sympy.diff(x**2*sympy.sin(1/x),x)

Out[12]: 2xsin(1/x)−cos(1/x)

In [13]: sympy.limit(2*x*sympy.sin(1/x)-sympy.cos(1/x),x,0)

Out[13]: ⟨−1,1⟩

In [14]: limit((f(x)-f(0))/(x-0),x,0,dir='+')

Out[14]: 0

In [15]: limit((f(x)-f(0))/(x-0),x,0,dir='-')

Out[15]: 0

la función es derivable en x = 0

EJERCICIO 3 Considera la función $f: \mathbb{R} - \pi/2, \pi/2) \rightarrow \mathbb{R}$ definida por $f(x) = ln\sqrt{\frac{1+Senx}{1-senx}}$

A) Comprueba si la función f es de clase C1.

In [16]: solve(1-sin(x),x)

Out[16]: [π/2]

In [17]: solve(1+sin(x),x)

La función es continua en todo su dominio.

Out[17]: [−π/2, 3π/2]

In [18]: x = Symbol('x')

fx = ln(sympy.sqrt(1+sympy.sin(x)/(1-sympy.sin(x))))

dx = Derivative(fx, x).doit()

dx

Out[18]:
$$\frac{\frac{\cos(x)}{2(1-\sin(x))} + \frac{\sin(x)\cos(x)}{2(1-\sin(x))^2}}{1 + \frac{\sin(x)}{1-\sin(x)}}$$

In [19]: simplify(diff(fx, x))

Out[19]:
$$\frac{\cos(x)}{2(1-\sin(x))}$$

B) Calcula la derivada de orden 5 de f .

In [20]: simplify(diff(fx, x,5))

Out[20]:
$$-\frac{(\sin(x)+5)\cos(x)}{2(\sin(x)-1)^3}$$

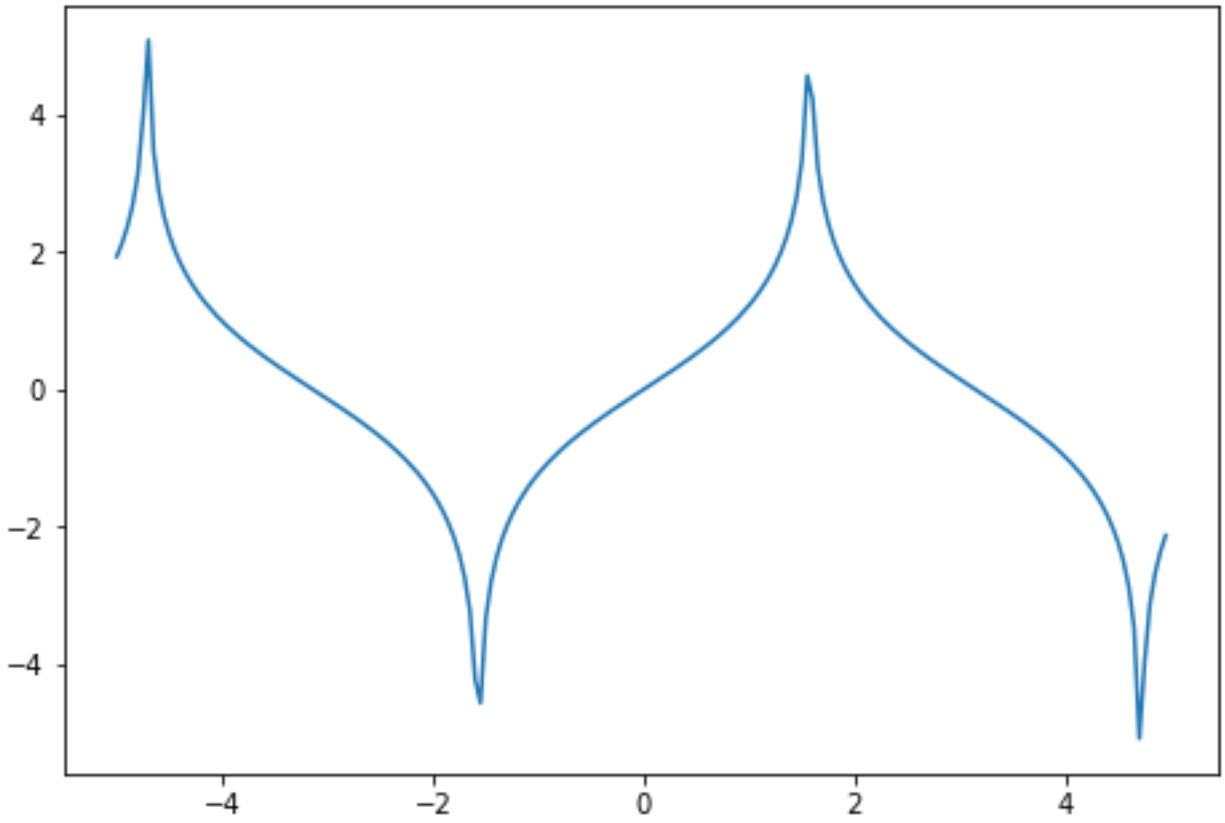
C) Representa la función.

```
In [21]: from matplotlib import pyplot as plt
from sympy import ln
from math import sqrt,sin
import numpy as np
import math
%matplotlib inline
```

```
x=np.arange(-5, 5, 0.05)
y=np.log(np.sqrt((1+np.sin(x))/(1-np.sin(x))))
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(x,y)
```

Out[21]:

[<matplotlib.lines.Line2D at 0x228a45930a0>]



EJERCICIO 4 Calcule y represente la recta tangente:

A) En el punto x = 0.3 de la función $\frac{x^3}{x^2-1}$ en el intervalo [-1,1].

```
In [22]: x = Symbol('x')
def f(x):
    return x**3/(x**2-1)
def TF(x):
    return f(0.3)+diff(f(x), x).subs(x, 0.3)*(x-0.3)
```

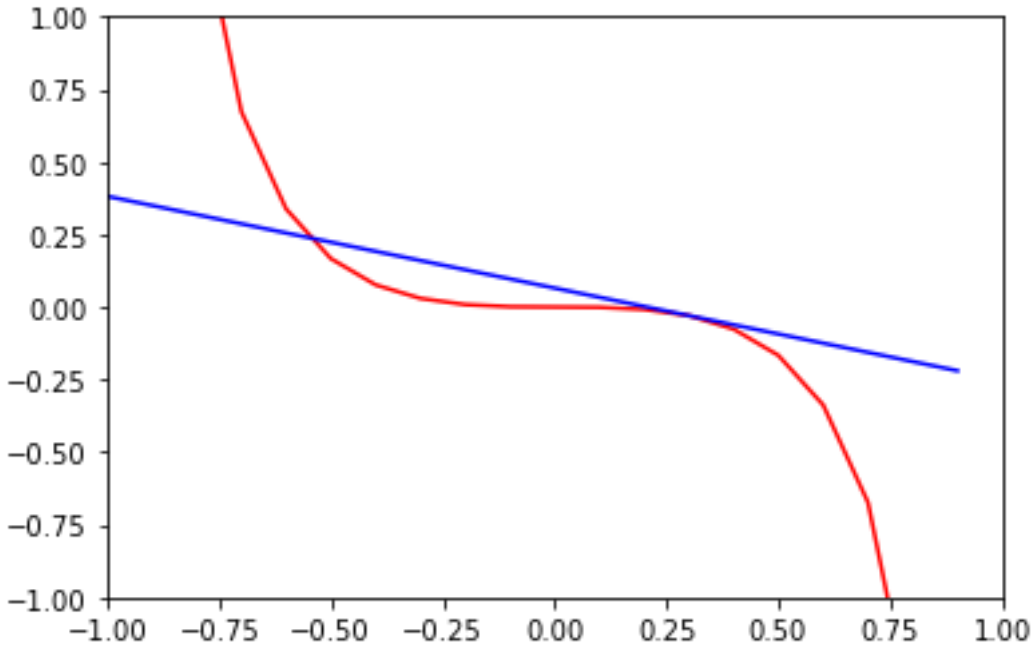
In [23]: TF(x)

Out[23]: 0.0652095157589663−0.316266151430987x

In [24]: **from numpy import** *

```
from matplotlib.pyplot import *
from matplotlib import pyplot
%matplotlib inline
x=arange(-1,1,0.1)
plot(x,x**3/(x**2-1),'-r')
plot(x,0.0652095157589663-0.316266151430987*x,'b')
pyplot.xlim(-1, 1)
pyplot.ylim(-1, 1)
```

Out[24]: (-1.0, 1.0)



B) En el punto x = -1 de la función $x^4 - 12x^3 + 48x^2 - 64x$

```
In [25]: x = Symbol('x')
def f(x):
    return x**4-12*x**3+48*x**2-64*x
def TF(x):
    return f(-1)+diff(f(x), x).subs(x, -1)*(x+1)
```

In [26]: TF(x)

Out[26]: −200x−75

```
In [27]: x=arange(-3,1,0.1)
plot(x,x**4-12*x**3+48*x**2-64*x,'-r')
```

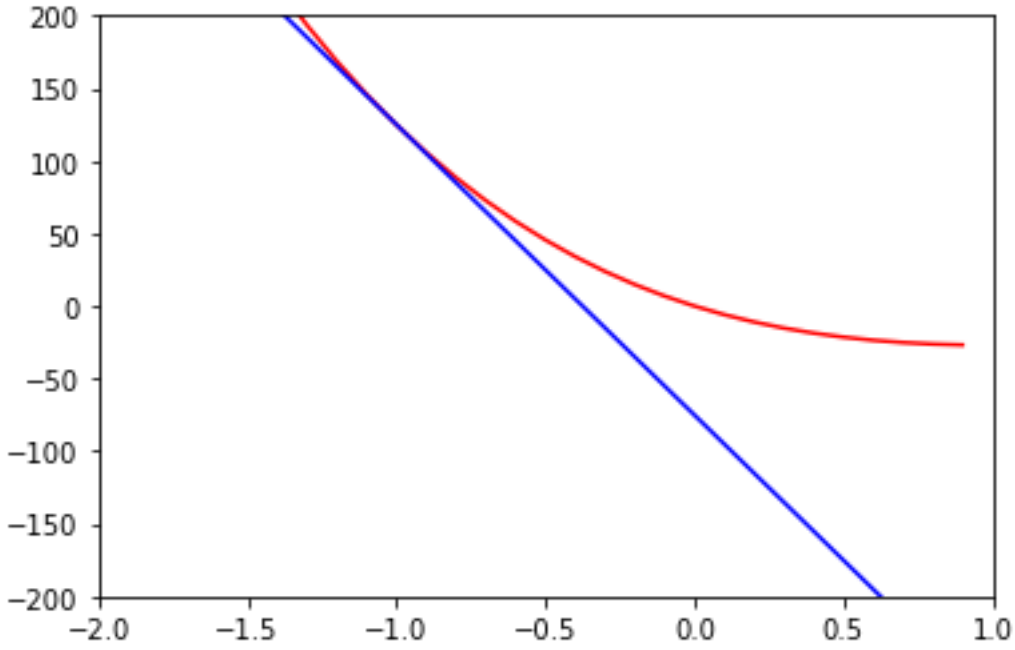
plot(x,-200*x-75,'b')

pyplot.xlim(-2, 1)

pyplot.ylim(-200, 200)

Out[27]:

(-200.0, 200.0)



EJERCICIO 5 Sea la función f : $\mathbb{R}\rightarrow\mathbb{R}$ definida por $f(x) = -x^3 + 3x$

A) Determina los puntos de corte de la gráfica de *f* con el eje de coordenadas.

In [28]: solve(-x**3+3*x,x)

Out[28]: [0, -√3, √3]

Los puntos de corte con el eje de coordenadas son : (0,0) (-√3,0) (√3,0)

In [29]: fx = -x**3+3*x

simplify(diff(fx, x))

Out[29]: 3−3x2

B) Calcula los extremos relativos y los puntos de inflexión de *f* .

In [30]: solve(3-3*x**2,x)

Out[30]: [−1, 1]

Los posibles máximos o mínimos son X = -1 y X = 1

In [31]: Derivative(fx, x, 2).doit()

Out[31]: −6x

In [32]: -6*-1

Out[32]: 6

In [33]: -6*1

Out[33]: −6

Tenemos un mínimo en X = -1 y un máximo en X = 1

In [34]:

CALCULAMOS LOS 0 DE LA SEGUNDA DERIVADA PARA VER LOS PUNTOS DE INFLEXIÓN

solve(-6*x,x)

Out[34]: [0]

In [35]: Derivative(fx, x, 3).doit()

Out[35]: −6

TENEMOS UN PUNTO DE INFLEXIÓN EN X = 0.

C) Representa gráficamente la función.

In [36]: x=np.arange(-2.5, 2, 0.05)

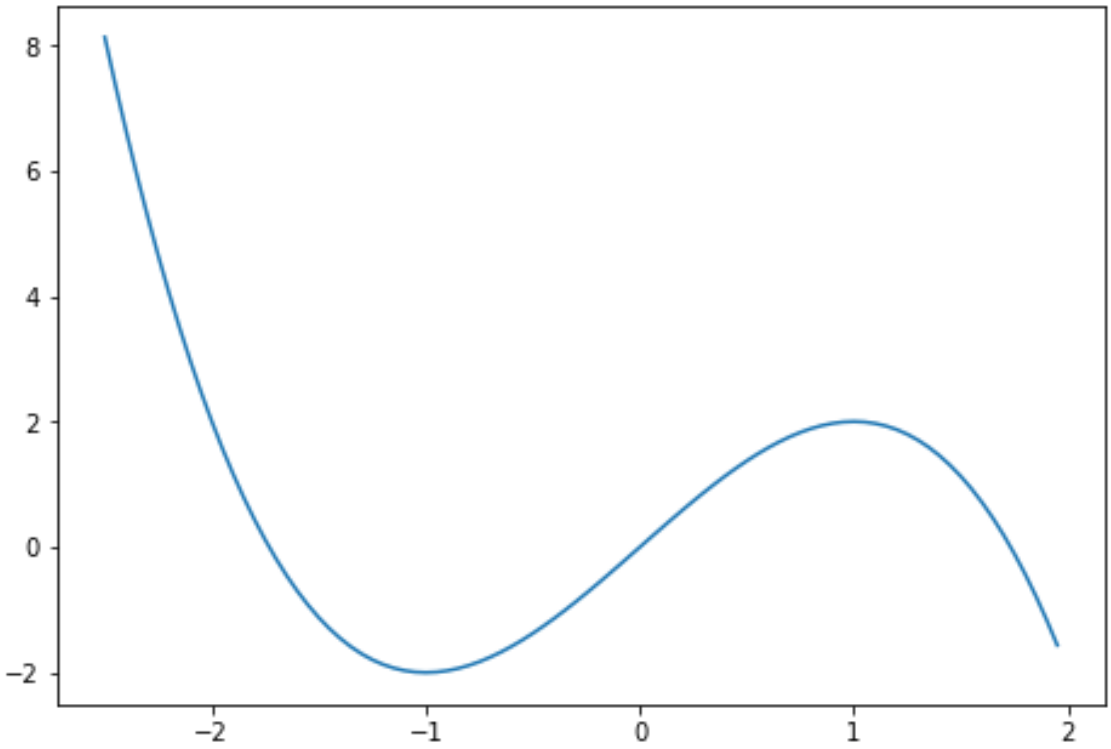
y=-x**3+3*x

fig=plt.figure()

ax=fig.add_axes([0,0,1,1])

ax.plot(x,y)

Out[36]:



EJERCICIO 6 Las funciones de ingresos y costes anuales por la fabricación y venta de q unidades de un determinado producto vienen dados por:

***I*(*q*) = 2000 *q* - 0.04 *q*2 y *C*(*q*) = 1 000 000 + 100 *q* + 0.001 *q*2.**

Halla:

A) La función que da el beneficio anual.

In [37]: ix = -0.04*x**2+2000*x

cx = 0.001*x**2+100*x+1000000

bx = ix-cx

bx

Out[37]: −0.041x2+1900x−1000000

B) ¿Cuántas unidades hay que producir y vender para que el beneficio sea máximo? ¿cuál es ese beneficio?

In [38]: simplify(diff(bx, x))

Out[38]: 1900−0.082x

In [39]: solve(-0.082*x+1900,x)

Out[39]: [23170.7317073171]

In [40]: Derivative(bx, x, 2).doit()

Out[40]: −0.082

ES MÁXIMO EL BENEFICIO CUANDO EL NÚMERO DE UNIDADES SEA 23170,7

In [41]: **def** b(x):

  return -0.041*x**2+1900*x-1000000

b(23170.7)

Out[41]: 21012195.12191

EL BENEFICIO QUE SE PRODUCE ES 21012195,12191.

6.PRÁCTICA 3: "INTEGRACIÓN DE FUNCIONES"

En esta práctica vamos a aprender a realizar integrales [15], [16] a través de las librerías sympy y matplot, las cuales son necesarias para la realización de estos ejercicios.

Vamos a importar para resolver las integrales los siguientes comandos:

- Integrate: Para resolver una integral. Sería integrate(función ,variable sobre la que se quiere integrar) y si es una integral definida lo mismo que la anterior pero después de poner la variable sobre la que se quiere integrar hay que poner los extremos del intervalo sobre los que se quiere integrar.
- Importamos los símbolos necesarios, como la x, el logaritmo neperiano de e , cos, pi etc.. a través del comando Symbol como en anteriores prácticas.
- Simplify, para simplificar las integrales.
- Para representar funciones como ya vimos en anteriores prácticas hay que utilizar el comando plot, también es necesario range(para darles valores a la x).

Para hallar y representar la recta tangente tendremos que realizar la derivada de la función y utilizar comandos como los ya mencionados plot, show, range.

6.1. EJERCICIOS PRÁCTICA 3

EJERCICIO 1 Calcula las siguientes integrales:

A) $\int \frac{e^{2^x}}{\sqrt{e^x+1}}\,dx$

In [1]: from sympy import*

integrate(e**(2*x)/sqrt(e**x+1),x)

Out[1]:

0.666666666666667·2.71828182845905^x$\sqrt{2.71828182845905\text{^}x + 1}$ −1.3333333333 $\sqrt{2.71828182845905^{x} + 1}$

B) $\int_{0}^{\pi/4} \frac{x}{cos^2 x}\, dx$

```
In [6]: integrate(x/cos(x)**2,(x,0,pi/4))
```

Out[6]:

$$\frac{\log\left(\left(-1+\sqrt{2}\right)^2+1\right)}{-1+\left(-1+\sqrt{2}\right)^2}+\frac{\left(-1+\sqrt{2}\right)^2\log\left(\sqrt{2}\right)}{-1+\left(-1+\sqrt{2}\right)^2}-\frac{\left(-1+\sqrt{2}\right)^2\log\left(\left(-1+\sqrt{2}\right)^2+1\right)}{-1+\left(-1+\sqrt{2}\right)^2}-\frac{\log\left(\sqrt{2}\right)}{-1+\left(-1+\sqrt{2}\right)^2}-\frac{\pi\left(-1+\sqrt{2}\right)}{2\left(-1+\left(-1+\sqrt{2}\right)^2\right)}$$
$$-i\pi+\frac{\left(-1+\sqrt{2}\right)^2\left(\log\left(2-\sqrt{2}\right)+i\pi\right)}{-1+\left(-1+\sqrt{2}\right)^2}-\frac{\log\left(2-\sqrt{2}\right)+i\pi}{-1+\left(-1+\sqrt{2}\right)^2}$$

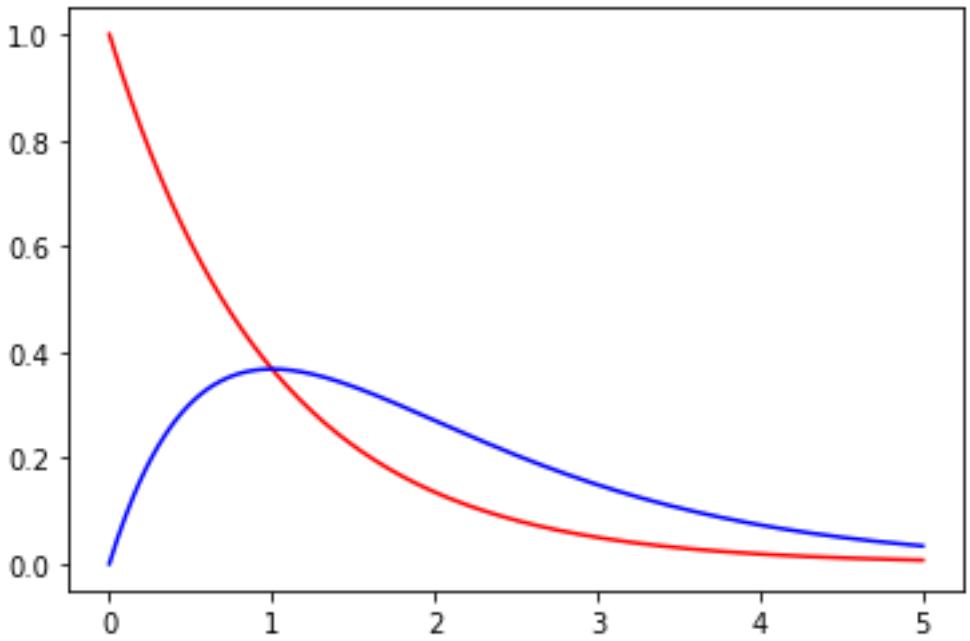
EJERCICIO 2 Calcula el área del recinto limitado por el eje de ordenadas y las curvas de ecuaciones $y = e^{-x}$, $y = x\, e^{-x}$

In [3]: from numpy import *

import math

import matplotlib.pyplot as plt

```
t = linspace(0, 5, 400)
a = e**(-x)
b = x*e**(-x)
plt.plot(t, a, 'r')
plt.plot(t, b, 'b')
plt.show()
```



In[4]: **from math import***

from matplotlib import pyplot

def f1(x):

  return e**-x

def f2(x):

  return x*e**-x

x = Symbol('x')

solve(f1(x)-f2(x),x)

Out[4]: [1.00000000000000]

In [5]: f1(0.5)

Out[5]: 0.6065306597126334

In [6]: f2(0.5)

Out[6]: 0.3032653298563167

In [7]: integrate(f1(x)-f2(x), (x, 0, 1))

Out[7]: 0.367879441171442

El área es 0.3679 unidades cuadradas.

EJERCICIO 3 Sea $f: \mathbb{R} \to \mathbb{R}$ la función definida por

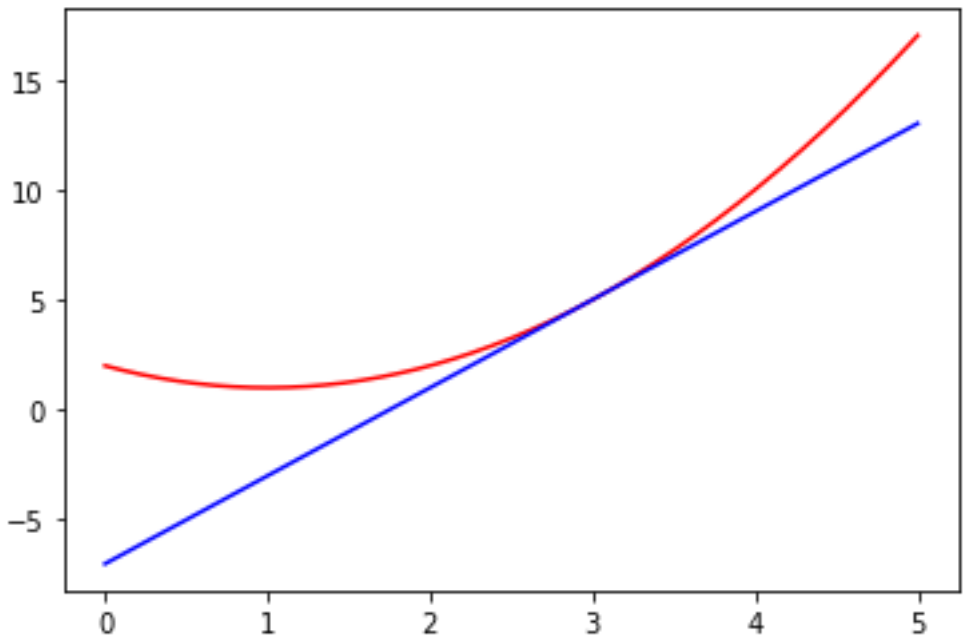
$$f(x) = x^2 - 2x + 2$$

A) Halla la ecuación de la recta tangente a la gráfica de *f* en el punto de abscisa *x* = 3.

```
In [8]: def f(x):
    return x**2-2*x+2
def TF(x):
    return f(3)+diff(f(x), x).subs(x, 3)*(x-3)
```

```
In [9]: TF(x)
Out[9]: 4x−7
```

```
In [10]: from numpy import *
import math
import matplotlib.pyplot as plt
x = linspace(0, 5, 400)
a = x**2-2*x+2
b = 4*x-7
plt.plot(x, a, 'r')
plt.plot(x, b, 'b')
plt.show()
```



B) Calcula el área del recinto limitado por la gráfica de *f* , la recta tangente obtenida y el eje OY.

```
In [11]: from sympy import*
x = Symbol('x')
solve(f(x)-TF(x),x)
```

Out[11]: [3]

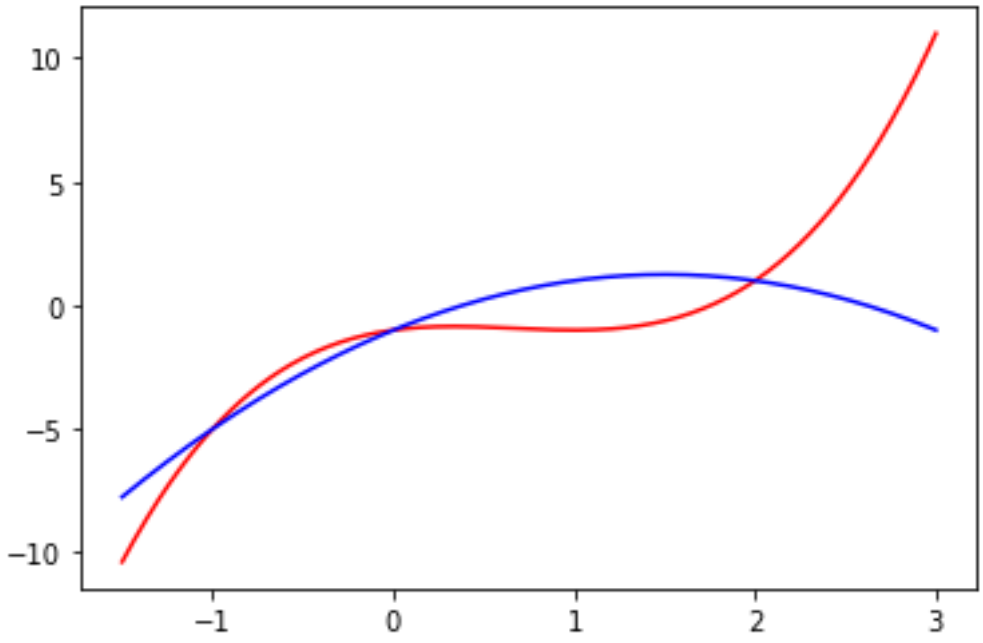
```
In [12]: integrate(f(x)-TF(x),(x,0,3))
```

Out[12]: 9

El área es 9 unidades cuadradas.

EJERCICIO 4 Calcula la suma de las áreas de los recintos acotados limitados por las curvas $y = x^3 - 2x^2 + x - 1$ e $y = -x^2 + 3x - 1$

```
In [13]: x = linspace(-1.5, 3, 400)
a = x**3-2*x**2+x-1
b = -x**2+3*x-1
plt.plot(x, a, 'r')
plt.plot(x, b, 'b')
plt.show()
```




```
In [14]: from sympy import*
def f(x):
    return x**3-2*x**2+x-1
def g(x):
    return -x**2+3*x-1
```

```
x = Symbol('x')
solve(f(x)-g(x),x)
```

Out[14]: [-1, 0, 2]

```
In [15]: integrate(f(x)-g(x),(x,-1,0))+integrate(g(x)-f(x),(x,0,2))
```

Out[15]: 37/12

El área pedida es 37/12 unidades cuadradas.

EJERCICIO 5 Dada la siguiente función definida a trozos

$$f(x) = \begin{cases} x+1 & x < 0 \\ x^2+1 & 0 \leq x < 1 \\ ln(x)+2 & x \geq 1 \end{cases}$$

A) Calcular $\int_1^5 f(x)\,dx$, $\int_{-1}^1 f(x)\,dx$ y $\int_0^5 f(x)\,dx$

```
In [14]: integrate(ln(x)+2,(x,1,5))
```

Out[14]: 4+5log(5)

```
In [15]: integrate(x+1,(x,-1,0))+integrate(x**2+1,(x,0,1))
```

Out[15]: 11/6

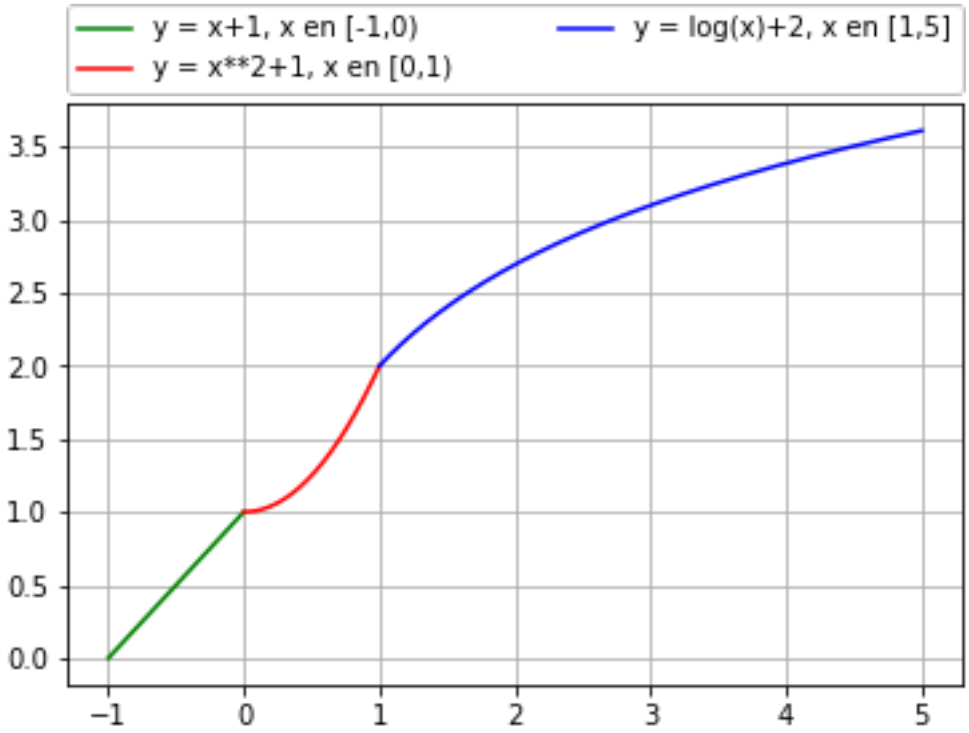
```
In [16]: integrate(x**2+1,(x,0,1))+integrate(ln(x)+2,(x,1,5))
```

Out[16]: 16/3+5log(5)

B) Calcular el área encerrada bajo la función y el eje de abscisas en el intervalo [-1,5].

```
In [17]: import numpy as np
import matplotlib.pyplot as plt
from math import*
x1 = np.linspace(-1, 0, 400)
x2 = np.linspace(0, 1, 400)
x3 = np.linspace(1, 5, 400)
plt.plot(x1, [x+1 for x in x1], 'g',  label='y = x+1, x en [-1,0)')
plt.plot(x2, [x**2+1 for x in x2], 'r',  label='y = x**2+1, x en [0,1)')
plt.plot(x3, [log(x)+2 for x in x3], 'b',  label='y = log(x)+2, x en [1,5]')
plt.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc='lower left',
           ncol=2, mode="expand", borderaxespad=0.)
plt.grid()
```

plt.show()



In [18]: integrate(x+1,(x,-1,0))+integrate(x**2+1,(x,0,1))+integrate(ln(x)+2,(x,1,5))

Out[18]: 35/6+5log(5)

El área pedida es 35/6+5log(5) unidades cuadradas.

EJERCICIO 6 En cierto proceso productivo la función de coste marginal es

$$c'(x) = \frac{3}{7000} - xe^x - 12x^2e^{-x} + 203 \text{ € /unidad}$$

Calcula lo que cuesta fabricar las 5 primeras piezas ¿y las 10 primeras?

```
In [19]: from math import*
def c(x):
    return 3/7000*(x*e**x)-12*x**2*e**-x+203
integrate(c(x),(x,0,5))
```

Out[19]: 994.246499597484

Lo que cuesta fabricar las 5 primeras piezas es 994,25€.

In [20]: integrate(c(x),(x,0,10))

Out[20]: 2091.02611927714

Lo que cuesta fabricar las 10 primeras piezas es 2091,03€.

EJERCICIO 7 La tasa de natalidad (=nacimientos/año) de cierta región viene dada por la función

$$T(t) = t^2 e^t 100$$

donde *t* indica el tiempo transcurrido expresado en años. Si la población inicial (*t* = 0) es de 350000 individuos, calcula la población que habrá al cabo de 50 años. Calcula asimismo la tasa de natalidad media durante esos 50 años.

```
In [21]: t = Symbol('t')
def T(t):
    return t**2*e**t*100
350000+integrate(T(t),(t,0,50))
```

Out[21]: $1.24536626796661 \cdot 10^{27}$

La población que habrá al cabo de 50 años es $1.24536626796661 \cdot 10^{27}$

```
In [22]: 1/50*integrate(T(t),(t,0,50))
```

Out[22]: $2.49073253593322 \cdot 10^{25}$

La tasa de natalidad media durante esos 50 años es de $2.49073253593322 \cdot 10^{25}$

EJERCICIO 8 Invertimos 1000 € en un fondo de inversión que ofrece interés variable compuesto continuamente dado, en tanto por uno, por I(t)=0.03+0.05t. Calcula el capital que recibimos al cabo de 5 años.

```
In [23]: def i(t):
    return 0.03+0.05*t
1000*e**(integrate(i(t),(t,0,5)))
```

Out[23]: 2170.59212718344

EL CAPITAL TRANSCURRIDO 5 AÑOS SERÁ DE 2170,59€

7.PRÁCTICA 4: "MATRICES"

Para manejar matrices en Jupyter utilizaremos from sympy import* y el comando Matrix para darle forma a la matriz.[17], [18], [19].

- Para calcular la transpuesta de una matriz se introduce el nombre que le hayamos puesto a la matriz anteriormente más .T. Un ejemplo sería si la matriz se llamara A, pues A.T
- Para el cálculo del determinante hay que poner la función: .det()
- Para el cálculo de la inversa podemos elevar la función a -1 y nos haría la inversa.
- Para calcular el rango de una matriz, basta con definir la matriz a través del comando Matrix y a continuación poner rank(nombre de la función definida)

Para hacer operaciones comunes de matrices como por ejemplo una suma primero tendríamos que definir las dos matrices dándole un nombre y con el comando Matrix y después con el símbolo +. Por ejemplo A = Matrix([[1,1]]),[1,1] B= Matrix([[1,1],[1,1]])

A+B. [19]

7.1. EJERCICIOS PRÁCTICA 4

EJERCICIO 1 Dadas las matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \mathbf{y} \quad \mathbf{B} = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

```
In [1]: from sympy import*
init_printing (use_unicode = True)
A = Matrix([[1,1,1],[1,0,1],[1,1,1]])
B = Matrix([[0,0,0],[2,0,0],[1,2,1]])
```

Calcula:

A) 3 A + 2 B.

```
In [2]:
3*A+2*B
```

```
Out[2]:
array([[3, 3, 3],
       [7, 0, 3],
       [5, 7, 5]])
```

B) $A * B^t$.

In [3]:

```
W = A*(B.T)
W
```

$$\begin{bmatrix} 0 & 2 & 4 \\ 0 & 2 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

Out[3]:

C) Det($A * B^t$).

In [4]: W.det()

Out[4]: 0

D) A^{127}

```
A**127
```

Out[3]:

$$\begin{bmatrix} 10713765734882642631807029119157159770066439306160898048 & 78430208583246046436540694886351003449164043396610785 28 \\ 78430208583246046436540694886351003449164043396610785 28 & 57414897531160759763059192610441188503000699329996390 40 \\ 10713765734882642631807029119157159770066439306160898048 & 78430208583246046436540694886351003449164043396610785 28 \end{bmatrix}$$

EJERCICIO 2 Demuestra, utilizando las matrices del ejercicio anterior, que no es cierta la propiedad conmutativa para el producto de matrices cuadradas.

In [5]:

```
A*B
```

Out[5]:

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

In [6]:

```
B*A
Como vemos no es cierta la propiedad conmutativa pues A*B es distinto de B*A
```

Out[6]:

$$\begin{bmatrix} 0 & 0 & 0 \\ 2 & 2 & 2 \\ 4 & 2 & 4 \end{bmatrix}$$

EJERCICIO 3 Siendo *A* y *B* las matrices del ejercicio 1, resuelve la ecuación matricial.

$$\mathbf{A} * \mathbf{X} + \boldsymbol{B}^{t} * \mathbf{X} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{2} \end{pmatrix}$$

```
C = Matrix([[1,1,1],[0,0,1],[1,1,2]])
X = (A+B.T)**-1*C
X
```

Out[6]:

$$\begin{bmatrix} 3 & 3 & \frac{11}{2} \\ 0 & 0 & -\frac{1}{2} \\ -1 & -1 & -\frac{3}{2} \end{bmatrix}$$

EJERCICIO 5 Calcula el rango de la matriz

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

In [5]: A = Matrix([[1,-1,-2],[0,1,1],[2,0,-2],[1,0,-1],[1,2,1]])

A.rank()

Out[5]: 2

EJERCICIO 6 Calcula *a* y *b* para que sea 2 el rango de la matriz

```
A = Matrix([[a,1,2],[1,2,0],[2,1,1],[-2,b,1]])
A
```

Out[2]:

$$\begin{bmatrix} a & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 1 & 1 \\ -2 & b & 1 \end{bmatrix}$$

In [6]: from sympy import*

a = Symbol('a')

b = Symbol('b')

B = Matrix([[a,1,2],[1,2,0],[2,1,1]])

B.det()

Out[6]: 2a−7

In [7]: C = Matrix([[a,1,2],[1,2,0],[-2,b,1]])

C.det()

Out[7]: 2a+2b+7

In [8]: D = Matrix([[a,1,2],[2,1,1],[-2,b,1]])

D.det()

Out[8]: −ab+a+4b

In [9]: E = Matrix([[1,2,0],[2,1,1],[-2,b,1]])

E.det()

Out[9]: −b−7

In [10]: solve(-7+2*a)

a = 7/2

Out[10]: [7/2]

In [11]: solve(-7-b)

b = -7

Out[11]: [-7]

In [12]: **from sympy import***

A = Matrix([[7/2,1,2],[1,2,0],[2,1,1],[-2,-7,1]])

A.rank()

Out[12]: 2

Para los valores a = 7/2 y b = -7 la matriz A tiene rango 2.

EJERCICIO 7. Considera las matrices

$$\mathbf{A}=\begin{pmatrix}\mathbf{1} & \mathbf{0} & \mathbf{-1}\\ \mathbf{0} & \boldsymbol{m} & \mathbf{3}\\ \mathbf{4} & \mathbf{1} & \mathbf{-m}\end{pmatrix} \quad \mathbf{B}=\begin{pmatrix}\mathbf{1}\\ \mathbf{-1}\\ \mathbf{3}\end{pmatrix} \quad \mathbf{X}=\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{y}\\ \boldsymbol{z}\end{pmatrix}$$

A) ¿Para que valores de m existe la matriz inversa A^{-1}?

m = Symbol('m')

A = Matrix([[1,0,-1],[0,m,3],[4,1,-m]])

A.det()

Out[11]: −m2+4m−3

In [12]: solve(-m**2+4*m-3)

Out[12]: [1, 3]

Existe la inversa siempre que m sea distinto de 1 y 3

B) Resuelve el sistema $A * X = B$ para $m = -1$.

```
A = Matrix([[1,0,-1],[0,-1,3],[4,1,1]])
B = Matrix([[1],[-1],[3]])
A**-1*B
```

Out[13]:

$$\begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \\ -\frac{1}{4} \end{bmatrix}$$

8.PRÁCTICA 5:SISTEMAS LINEALES.VECTORES

Para la realización de esta práctica, como en la anterior, se utiliza la librería sympy y math, definir las matrices con el comando Matrix y hacer las operaciones correspondientes como el determinante, la transpuesta, la inversa etc...[17]

Para resolver un sistema lineal hay que definir la matriz y la matriz de coeficientes y a continuación insertar el comando solve que nos resolvería el sistema y a partir de ahí sacar las conclusiones correspondientes sobre la solución obtenida.

Hay que resaltar que en Jupyter no existe la instrucción análoga a la que se tiene en Mathematica, Reduce, que permita discutir, de un modo directo, sistemas de ecuaciones lineales con parámetros.

Para estudiar si un vector es linealmente dependiente o independiente, [18], [19], hay que hacer su determinante y si el resultado es 0, los vectores son linealmente dependientes y si no es 0 el determinante, no son dependientes.

8.1 EJERCICIOS PRÁCTICA 5

EJERCICIO 1 Resuelve, invirtiendo la matriz de coeficientes, el sistema de ecuaciones lineales con incógnitas *x*, *y*, *z*:

$$\left.\begin{array}{l} x + y + z = 2 \\ 2x - y + 3z = 1 \\ x + z = 1 \end{array}\right\}$$

```
In [1]: from sympy import*
A = Matrix([[1,1,1],[2,-1,3],[1,0,1]])
B = Matrix([[2],[1],[1]])

In [2]: A.det()
Out[2]: 1

In [5]: A**-1*B
        La solución es X = 1, Y = 1 y Z = 0
Out[5]:
```

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

EJERCICIO 2 Resuelve los siguientes sistemas de ecuaciones lineales con incógnitas *x*, *y*, *z*, *t* :

A)
$$\left.\begin{array}{l} 3x + 2y - z = 5 \\ x - y + 2z = 1 \\ -x + z - t = 0 \\ 5x + y - 2z + 3t = 6 \end{array}\right\}$$

```
In [3]: x = Symbol('x')
y = Symbol('y')
z = Symbol('z')
t = Symbol('t')
solve([3*x+2*y-z-5,x-y+2*z-1,-x+z-t,5*x+y-2*z+3*t-6],[x,y,z,t])

Out[3]: {x: 0, y: 11/3, z: 7/3, t: 7/3}
```

$$\begin{matrix} x - y + z = 1 \\ x + az = 2 \\ 2x + y + 2z = b \end{matrix}$$ **B)** }

In [4]: a = Symbol('a')

b = Symbol('b')

solve([x-y+z-1,x+a*z-2,2*x+y+2*z-b],[x,y,z])

Out[4]: {x: (a*b + a - 6)/(3*(a - 1)), y: b/3 - 2/3, z: (5 - b)/(3*(a - 1))}

In [5]: solve([x-y+z-1,x+0*z-2,2*x+y+2*z-b],[x,y,z])

Out[5]: {x: 2, y: b/3 - 2/3, z: b/3 - 5/3}

Si a!=1 el sistema es compatible determinado

In [6]: linsolve([x-y+z-1,x+1*z-2,2*x+y+2*z-5],[x,y,z])

Out[6]: {(2−z, 1, z)} {(2−z, 1, z)}

Si a=1 y b=5 el sistema es compatible indeterminado

In [7]: linsolve([x-y+z-1,x+1*z-2,2*x+y+2*z-1],[x,y,z])

Out[7]: ∅

Si a=1 y b!=5 el sistema es incompatible

EJERCICIO 3 Discute y resuelve, en los casos en que sea posible, los sistemas de ecuaciones lineales con incógnitas *x*, *y*, *z* :

$$\begin{matrix} x + y + z = 1 \\ x + az = 2 \\ 2x + y + 2z = b \end{matrix}$$ **A)** }

In [8]: solve([x+y+z-1,x+a*z-2,2*x+y+2*z-b],[x,y,z])

Out[8]: {x: (a*b - a - 2)/(a - 1), y: 2 - b, z: (3 - b)/(a - 1)}

In [9]: solve([x+y+z-1,x+0*z-2,2*x+y+2*z-b],[x,y,z])

Out[9]: {x: 2, y: 2 - b, z: b - 3}

Si a!=1 el sistema es compatible determinado

In [10]: linsolve([x+y+z-1,x+1*z-2,2*x+y+2*z-3],[x,y,z])

Out[10]: {(2−z, −1, z)}

Si a=1 y b=3 el sistema es compatible indeterminado

In [11]: linsolve([x+y+z-1,x+1*z-2,2*x+y+2*z-0],[x,y,z])

Out[11]: ∅

Si a=1 y b!=3 el sistema es incompatible

$$cx + ay = 1 - ca^2$$

B) $$ax + cy = 1 - ac^2$$ }

$$acx + acy = -a^2c^2$$

```
In [12]: from sympy import*
init_printing()
a,b,c,x,y,z = symbols('a b c x y z')
```

```
In [1]: from sympy import*
x,y,z,a,b,c = symbols('x,y,z,a,b,c', real=True)
system = [c*x+a*y+c*a**2-1,a*x+c*y+a*c**2-1,a*c*x+a*c*y+a**2*c**2]
nonlinsolve(system, [x,y])
```

Out[1]:

$$\left\{\left(-\frac{ac^2 - \frac{c\left(a^2c+ac^2-1\right)}{a+c} - 1}{a},\ -\frac{a^2c+ac^2-1}{a+c}\right)\right\}$$

In[13]: solve([1*x+0*y+1*0**2-1,0*x+1*y+0*1**2-1,0*1*x+0*1*y+0**2*1**2],[x,y]

Out[13]: {x: 1, y: 1}

Si a = 0 y c!=0 el sistema es compatible determinado.

In [14]:

solve([0*x+1*y+0*1**2-1,1*x+0*y+1*0**2-1,1*0*x+1*0*y+1**2*0**2],[x,y])

Out[14]: {x: 1, y: 1}

Si a!= 0 y c = 0 el sistema es compatible determinado.

In [15]:

linsolve([2*x+3*y+2*3**2-1,3*x+2*y+3*2**2-1,3*2*x+3*2*y+3**2*2**2],[x,y])

Out[15]: ∅

para todos los demás valores el sistema es incompatible.

EJERCICIO 4 Estudia la dependencia o independencia lineal de los vec tores:

$\overrightarrow{v_1} = (1,0,1,1), \overrightarrow{v_2} = (0,1,1,0), \vec{v}_3 = (1,1,-1,1)$ y $\vec{v}_4 = (0,0,3,0)$

Calcula una base del subespacio de $\mathbb{R}^4$generado por ellos.

```
In [16]: from sympy import*
M = Matrix([[1,0,1,0],[0,1,1,0],[1,1,-1,3],[1,0,1,0]])
M.det()
```

Out[16]: 0

ya que su determinante es igual a 0, los vectores son linealmente dependientes.

In [17]: M.rank()

Out[17]: 3

In [18]: Q = Matrix([[1,0,1,0],[0,1,1,0],[1,1,-1,3]])

Q.rank()

Out[18]: 3

Puesto que el rango de la matriz formada, por filas, por v1,v2,v3,v4 era 3 y, al calcular ahora el rango de la matriz formada por filas, por v1,v2,v3, sigue siendo 3, lo que tenemos es que v1,v2,v3 generan el mismo subespacio que v1,v2,v3,v4. Además son linealmente independientes al ser 3 el rango con lo que {v1,v2,v3} es una base del subespacio de R^4 generado por v1,v2,v3,v4.

9.PRÁCTICA 6: DIAGONALIZACIÓN DE MATRICES

En esta práctica sobre diagonalización de matrices vamos a ver si un vector es vector propio de una matriz, el cual se hace multiplicando la matriz por el vector y si el resultado que nos sale es múltiplo del vector es vector propio y si no es múltiplo, no lo es.

Para calcular todos los valores propios de una matriz se utiliza la función eigenvals() y para calcular todos los vectores propios, la función eigenvects().

Otra cuestión que vamos a estudiar en esta práctica es ver si una matriz es diagonalizable o no, para ello lo que tenemos que hacer es ver sus valores y vectores propios con las anteriores funciones y ver si la multiplicidad algebraica y la multiplicidad geométrica de los valores propios coinciden, si coinciden es diagonalizable si no coinciden no lo es. [19]

9.1 EJERCICIOS PRÁCTICA 6

EJERCICIO 1 Dada la matriz $A = \begin{bmatrix} 1 & 3 & -1 \\ 3 & 0 & 3 \\ -1 & 3 & 2 \end{bmatrix}$, estudia si el vector (0, 0, 1) es vector propio de A.

In [1]: from sympy import*

```
A = Matrix([[1,3,-1],[3,0,3],[-1,3,2]])
B = Matrix([[0],[0],[1]])
A*B
```

Out[1]:

$$\begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix}$$

El vector (0,0,1) no es vector propio de la matriz, ya que el resultado no es múltiplo del vector.

EJERCICIO 2 Calcular todos los valores y vectores propios de la matriz $\begin{pmatrix} 8 & 1 & 1 \\ 1 & 6 & 2 \\ 1 & 3 & 7 \end{pmatrix}$.

```
In [2]: A = Matrix([[8,1,1],[1,6,2],[1,3,7]])
A.eigenvals()
```

Out[2]: {10: 1, 7: 1, 4: 1}

```
In [3]: A.eigenvects()
```

Out[3]:
```
[(4,
  1,
  [Matrix([
   [ 0],
   [-1],
   [ 1]])]),
 (7,
  1,
  [Matrix([
   [-3/2],
   [ 1/2],
   [   1]])]),
 (10,
```

1,

[Matrix([

[6/7],

[5/7],

[1]])])]

Los valores propios son 10, 7 y 4 y los vectores propios son: (6,5,7), (-3,1,2), (0,-1,1).

EJERCICIO 3 Dada la matriz $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & -5 & 4 \end{bmatrix}$.

A) Estudiar si 2 es valor propio de la matriz *A*.

In [4]: A = Matrix([[0,1,0],[0,0,1],[2,-5,4]])

I = Matrix([[1,0,0],[0,1,0],[0,0,1]])

R = A-2*I

R

Out[4]: $\begin{bmatrix} -2 & 1 & 0 \\ 0 & -2 & 1 \\ 2 & -5 & 2 \end{bmatrix}$

In [5]: R.det()

Out[5]: 0

In [6]: A.eigenvals()

Out[6]: {2: 1, 1: 2}

2 es valor propio de A.

B) ¿Son (1, 1, 1) y (0, 0, 1) vectores propios de la matriz *A*? En caso afirmativo calcula el valor propio asociado.

In [7]: v1 = Matrix([[1],[1],[1]])

v2 = Matrix([[0],[0],[1]])

A*v1

Out[7]: $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

El vector que hemos llamado v1 es vector propio de la matriz ya que el resultado es múltiplo de v1. 1*v1 y el valor propio asociado es 1.

In [8]: A*v2

Out[8]: $\begin{bmatrix} 0 \\ 1 \\ 4 \end{bmatrix}$

El vector v2 no es vector propio de la matriz.

C) ¿Es diagonalizable la matriz *A*?

```
In [4]: A.eigenvects()

Out[4]: [(1,
          2,
          [Matrix([
           [1],
           [1],
           [1]])]),
         (2,
          1,
          [Matrix([
           [1/4],
           [1/2],
           [ 1]])])]

In [ ]: La matriz no es diagonalizable.porque la multiplicidad algebraica y la multiplicidad geometrica del valor propio 1 no coinciden.
```

EJERCICIO 4 Estudiar si las siguientes matrices son diagonalizables en función del parámetro *a*:

$$\begin{bmatrix} a & 0 & 0 \\ 0 & 1 & a \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} a & 0 & 0 \\ 0 & 2 & 0 \\ a & -1 & 1 \end{bmatrix}$$

In [9]: a = symbols('a')

A = Matrix([[a,0,0],[0,1,a],[0,0,-1]])

B = Matrix([[a,0,0],[0,2,0],[a,-1,1]])

In [10]: A.eigenvals()

Out[10]: {a: 1, 1: 1, -1: 1}

```
In [13]: A.eigenvects()

Out[13]: [(-1,
           1,
           [Matrix([
            [   0],
            [-a/2],
            [   1]])]),
          (1,
           1,
           [Matrix([
            [0],
            [1],
            [0]])]),
          (a,
           1,
           [Matrix([
            [1],
            [0],
            [0]])])]
```

Para cualquier valor de a la matriz es diagonalizable porque la multiplicidad algebraica y geométrica coinciden para cada valor propio.

In [11]: B.eigenvals()

Out[11]: {a: 1, 2: 1, 1: 1}

```
In [15]: B.eigenvects()

Out[15]: [(1,
           1,
           [Matrix([
            [0],
            [0],
            [1]])]),
          (2,
           1,
           [Matrix([
            [ 0],
            [-1],
            [ 1]])]),
          (a,
           1,
           [Matrix([
            [(a - 1)/a],
            [        0],
            [        1]])])]
```

```
In [ ]: estudiamos el caso a = 0
```

```
In [16]: C = Matrix([[0,0,0],[0,2,0],[0,-1,1]])
         C.eigenvects()
```

```
Out[16]: [(0,
           1,
           [Matrix([
            [1],
            [0],
            [0]])]),
          (1,
           1,
           [Matrix([
            [0],
            [0],
            [1]])]),
          (2,
           1,
           [Matrix([
            [ 0],
            [-1],
            [ 1]])])]
```

```
In [ ]: Cuando a = 0 la matriz es diagonalizable
```

```
In [24]: D = Matrix([[1,0,0],[0,2,0],[1,-1,1]])
         D.eigenvects()
```

```
Out[24]: [(1,
           2,
           [Matrix([
            [0],
            [0],
            [1]])]),
          (2,
           1,
           [Matrix([
            [ 0],
            [-1],
            [ 1]])])]
```

Para a = 1 la matriz no es diagonalizable ya que en este caso, para el valor propio 1, la multiplicidad algebraica es 2 y la multiplicidad geométrica, 1.

Para el resto de casos de a la matriz es diagonalizable porque al tener 3 valores propios distintos la multiplicidad algebraica y geométrica siempre va a ser 1.

EJERCICIO 5 Dada la matriz $A = \begin{bmatrix} 5 & 0 & 1 \\ 0 & 2 & 0 \\ 5 & -1 & 1 \end{bmatrix}$ calcular A^{20} a partir de la diagonalización de la matriz A.

In [12]: A = Matrix([[5,0,1],[0,2,0],[5,-1,1]])

A.eigenvals()

Out[12]: {6: 1, 2: 1, 0: 1}

In [35]:
```
A.eigenvects()
```

Out[35]:
```
[(0,
  1,
  [Matrix([
   [-1/5],
   [   0],
   [   1]])]),
 (2,
  1,
  [Matrix([
   [-1/3],
   [-8/3],
   [   1]])]),
 (6,
  1,
  [Matrix([
   [1],
   [0],
   [1]])])]
```

In [5]:
```
P,D = A.diagonalize()
P
```

Out[5]:
$$\begin{bmatrix} -1 & -1 & 1 \\ 0 & -8 & 0 \\ 5 & 3 & 1 \end{bmatrix}$$

In [49]:
```
D
```

Out[49]:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

In [50]:
```
P*D*P**-1
```

Out[50]:
$$\begin{bmatrix} 5 & 0 & 1 \\ 0 & 2 & 0 \\ 5 & -1 & 1 \end{bmatrix}$$

In [51]:
```
P*D*P**-1 == A
```

Out[51]:
```
True
```

In [ ]:
```
La matriz A es diagonalizable
```

In [10]:
```
diagmat = D**20
diagmat
```

Out[10]:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1048576 & 0 \\ 0 & 0 & 3656158440062976 \end{bmatrix}$$

In [11]:
```
P*diagmat*P**-1
```

Out[11]:
$$\begin{bmatrix} 3046798700052480 & -152339934871552 & 609359740010496 \\ 0 & 1048576 & 0 \\ 3046798700052480 & -152339935395840 & 609359740010496 \end{bmatrix}$$

In [12]:
```
P*diagmat*P**-1 == A**20
```

Out[12]:
```
True
```

10.CONCLUSIONES

Como conclusión hemos sacado que este software llamado Jupyter puede abarcar y resolver todo tipo de funciones tanto matemáticas como investigaciones científicas, no es difícil de utilizar si se le dedica tiempo a la comprensión del programa pero una vez comprendido es fácil y rápido de utilizar.

De cada ejercicio se puede sacar una enseñanza ya que nos da los resultados de una forma exacta y fácil de comprender para el lector, es una herramienta muy completa que nos permite representar funciones y a la vez también es gratuita por lo que se puede utilizar simplemente teniendo un ordenador en casa.

Por otro lado, desde mi punto de vista, creo que la Universidad de Jaén podría estudiar la alternativa de añadir este software a su plataforma como alternativa al Mathematica, tanto para que el alumnado aprenda nuevas herramientas virtuales como para el aprendizaje de estos, al que le ayudaría, creo, a entender mejor la asignatura de matemáticas y ya no solamente a ellos sino al profesorado.

11.BIBLIOGRAFÍA

Los enlaces de todas las páginas web recogidas en esta bibliografía están vigentes a fecha 30/05/2021:

[1].https://www.python.org/

[2].https://www.learnpython.org/es/

[3].https://entrenamiento-python-basico.readthedocs.io/es/latest/

[4].https://www.paradigmadigital.com/dev/jupyter-data-science-aplicada/

[5].https://www.anaconda.com/products/individual

[6]https://jupyter-notebook.readthedocs.io/en/stable/

[7] https://relopezbriega.github.io/blog/2015/12/02/introduccion-al-calculo-con-python/

[8]https://ernestocrespo13.wordpress.com/2015/02/21/calculo-de-limites-con-la-libreria-sympy/

[9]https://python-para-impacientes.blogspot.com/2014/08/graficos-en-ipython.html

[10]https://recursospython.com/codigos-de-fuente/graficar-funciones-matplotlib/

[11]https://medium.com/@Med1um1/using-matplotlib-in-jupyter-notebooks-comparing-methods-and-some-tips-python-c38e85b40ba1

[12]https://www.lawebdelprogramador.com/foros/Python/1736806-Graficar-Funciones-a-trozos.html

[13]https://stackoverflow.com/questions/22276066/how-to-plot-multiple-functions-on-the-same-figure-in-matplotlib

[14]https://pybonacci.org/2012/04/30/como-calcular-limites-derivadas-series-e-integrales-en-python-con-sympy/

[15]https://numython.github.io/posts/integrales-con-sympy/

[16]https://www.diegocalvo.es/matrices-en-python/

[17]https://relopezbriega.github.io/blog/2016/02/10/mas-algebra-lineal-con-python/

[18]https://relopezbriega.github.io/blog/2015/06/14/algebra-lineal-con-python/

[19]https://ichi.pro/es/diagonalizacion-de-la-matriz-valor-propio-vector-propio-122839708524010

[20]https://mail.google.com/mail/u/2?ui=2&ik=45f50cf367&attid=0.1&permmsgid=msg-f:1662167624324767219&th=171134a5d5085df3&view=att&disp=safe&realattid=f_k87r88ed0 (Apuntes del mathematica)

[21]A.J. López, J. Jódar. Apuntes de la asignatura Matemáticas I para el grado en Administración y Dirección de Empresas. Universidad de Jaén (2020).