Temario

- Repasos varios
- Graficación usando matplotlib.
- Derivadas numéricas

Jueves 30 Setiembre. 1er Parcial.

Para rendir el 1er parcial deben enviar email a prog.fis.unne@gmail.com con Nombre y DNI. Hasta Martes 28.

Variables acumuladores

$$y = \sum_{i}^{N} i^2$$

En formato secuencial esto es:

$$y_N = y_{N-1} + i^2|_{i=N}$$

En términos computacionales:

```
yacum=0.0
for i in range(N+1):
    yacum = yacum + i**2
```

En el caso del factorial $N!$:

```
facum=1.0
for i in range(2,N+1):
    facum = facum * i
```

Funciones como argumentos de entrada de funciones

Supongamos que tenemos una rutina de integración, que queremos nos sirva para integrar cualquier función matemática.
Los lenguajes (incluido python) permiten que se pueda ingresar como argumento de entrada al nombre de una función:

```
def integracion(fn,a,b,n):
```

donde a, b son los límites inferior y superior. n la cantidad de evaluaciones y fn es el nombre de la función a evaluar.
Las funciones que tienen como argumento otras funciones son funciones de alto-orden.

Funciones como argumentos de entrada de funciones

```
def integracion(fn,a,b,n):
    delta=(b-a)/n
    x=a
    integ=0.0
    for i in range(n):
       integ=integ+ fn(x) * delta
       x=x+delta
    return integ
```

Se llama con

```
def absin(x):
    return m.sin(abs(x))
def abcos(x):
    return m.cos(abs(x))
integracion(absin,-1.,1.,100) # funcion definida por nost
integracion(abcos,-1.,1.,100)
integracion(m.cos,-1.,1.,100) # funcion del math
```

Graficación en python

La librería de graficación se llama matplotlib. Dentro de esta tenemos dos opciones: pylab y pyplot.

- pylab. Interface de graficación inspirada en Matlab.
- pyplot. Librería de graficación basada en objetos.

Por el momento usemos la opción mas sencilla pylab.
El que este interesado en hacer gráficos mas “cool” (personalizados) es conveniente la pyplot pero de todas maneras los conceptos son similares.

Graficación en python

plot es para graficar curvas 2D.

show para mostrar la figura. OJO NO usar en el servidor

```
import pylab
import numpy as np

x = np.linspace(0, 20, 1000)
y = np.sin(x)

pylab.plot(x, y)
pylab.show()
```

Límites de los ejes

xlim(xmin,xmax), ylim(ymin,ymax)

En el caso anterior el plot acomoda los ejes a los máximos. Definamos nosotros lo límites de los ejes.

```
pylab.plot(x, y)
pylab.xlim(5, 15)
pylab.ylim(-1.2, 1.2)
```

Títulos de gráficos

xlabel('x'), ylabel('y'), title('Titulo')

```
pylab.plot(x, y)


pylab.xlabel('this is x!')
pylab.ylabel('this is y!')
pylab.title('My First Plot')
```

Entiende latex para escribir fórmulas:

```
y = np.sin(2 * np.pi * x)
pylab.plot(x, y)
pylab.title(r'$\sin(2 \pi x)$')
```

Tipos de curvas/líneas

```
pylab.plot(x, y, '-r')
```

Colores disponibles:

'r' = red - rojo

'g' = green - verde

'b' = blue - azul

'c' = cyan - celeste

'm' = magenta - violeta

'y' = yellow - amarillo

'k' = black - negro

'w' = white - blanco

Las líneas pueden tener distintos estilos:

'-' = línea continua

'–' = línea a trazos

':' = línea punteada

'-.' = punteada a trazos

'.' = puntos

'o' = círculos

'^' = triángulos

Múltiple curvas y leyendas

```
x = np.linspace(0, 20, 1000)
y1 = np.sin(x)
y2 = np.cos(x)

pylab.plot(x, y1, '-b', label='sine')
pylab.plot(x, y2, '-r', label='cosine')
pylab.legend(loc='upper right')
pylab.ylim(-1.5, 2.0)
```

Múltiple plots

subplot(filas, columnas, nro de plot)

El nro empieza en 1, y sigue la numeración en orden de lectura (izq a der arriba a abajo).

```
>>> subplot(2, 2, 1)
>>> plot(x, sin(x))
>>> subplot(2, 2, 2)
>>> plot(x, cos(x))
>>> subplot(2, 1, 2)
>>> plot(x, x**2-x)
```

Guardar el gráfico en un archivo

Para guardar la figura en un archivo de imagen tenemos el comando:
savefig(fname, dpi=None, facecolor='w', edgecolor='w',
orientation='portrait', papertype=None, format=None):
Ejemplo:

```
savefig('fig05.png', dpi=80)
```

Formatos recomendados: png o eps (conservan la resolución de fuentes y líneas cunado se cambia el tamaño).

Esta es la opción recomendada en el servidor (en lugar del show).

Gráficos de densidades

Si lo que queremos graficar es una “imagen” o gráfico de densidad 2d se utiliza el comando:

```
imshow(data)
show()
```

Para cambiar el origin: `origin="lower"`
Si se quiere el gráfico en blanco y negro en lugar del “heatmap” se utiliza:

```
imshow(data)
gray()
show()
```

Para cambiar la escala del gráfico:

```
imshow(data,extend=[0,10,0,5])
```

Se queremos cambiar el aspecto (la razón de distancia entre x e y)

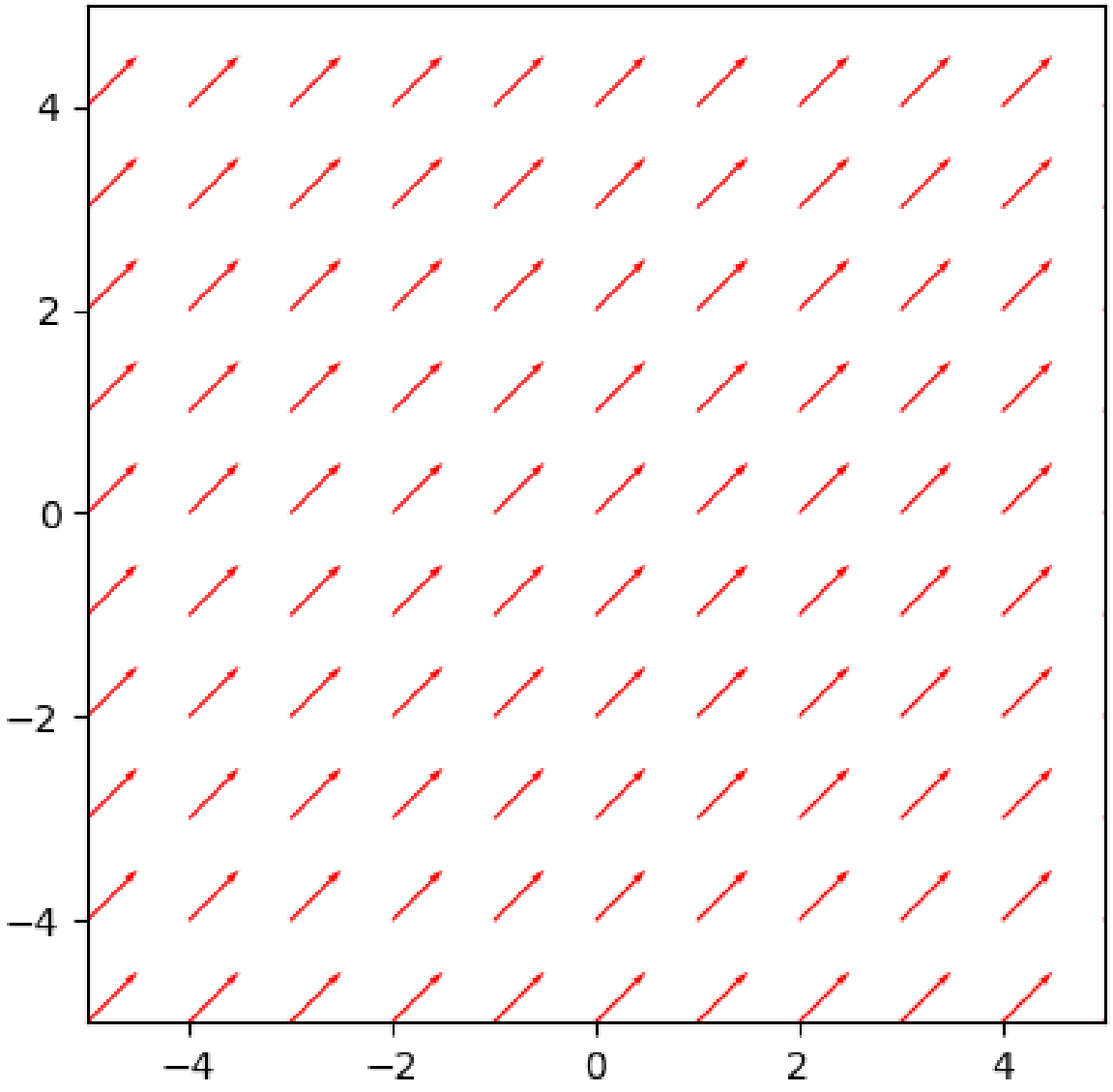
`aspect=2`

Campos de vectores

Función `quiver`. pyplot!

```
import matplotlib.pyplot as plt
import numpy as np
X, Y = np.meshgrid(np.arange(-10, 10, 1), np.arange(-10, 10, 1))
x_shape = X.shape
U=np.ones(x_shape)
V=np.ones(x_shape)
fig, ax = plt.subplots()
q = ax.quiver(X, Y, U, V, units='xy' ,scale=2, color='red')
ax.set_aspect('equal')
ax.set_xlim(-5,5)
ax.set_ylim(-5,5)
plt.show()
```

Campos de vectores



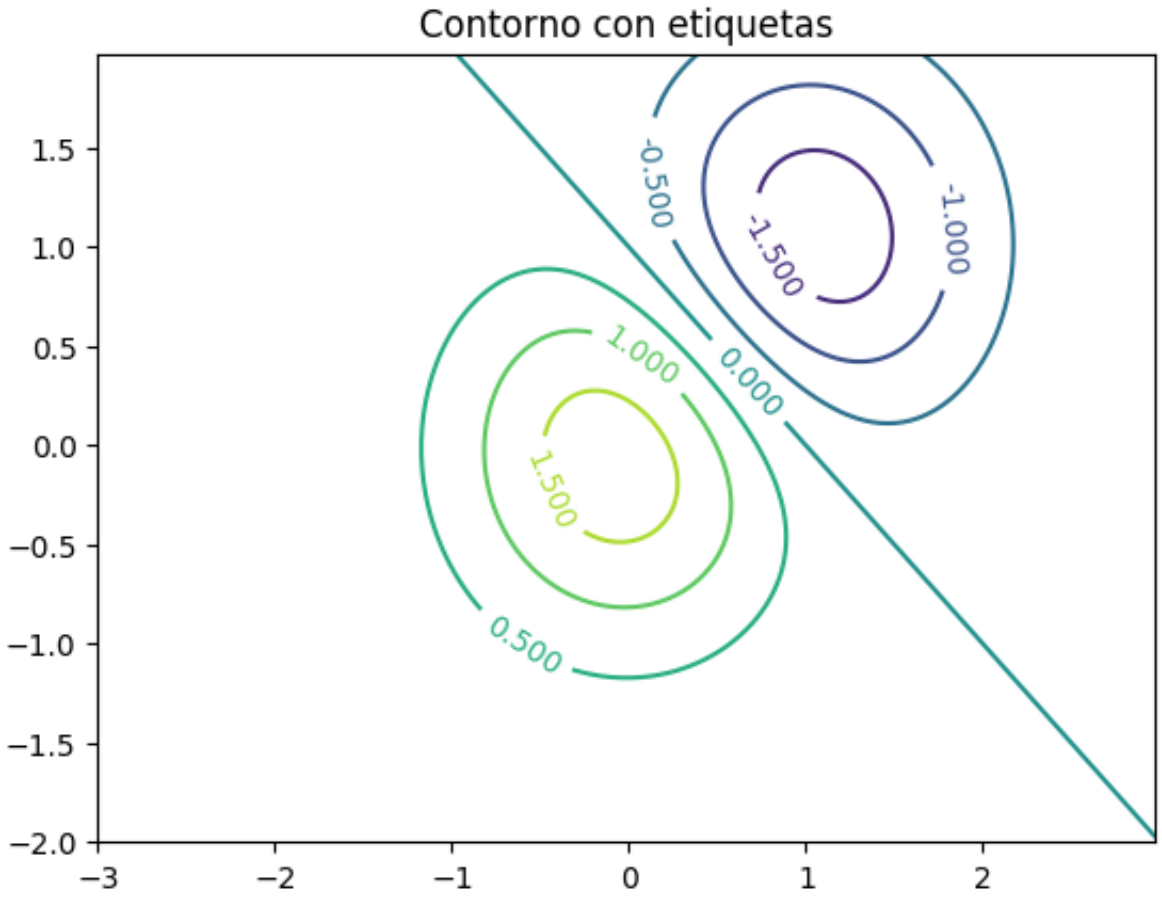
Contornos

Función `contour`. pyplot!

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
CS = ax.contour(X, Y, Z)
ax.clabel(CS, inline=True, fontsize=10)
ax.set_title('Contorno con etiquetas')
plt.savefig('contorno.png', bbox_inches='tight')
plt.show()
plt.close()
```

Contorno


Contorno con etiquetas
-0.500
-1.000
-1.500
0.000
1.000
1.500
0.500
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−3
−2
−1
0
1
2

Mas alla!

Hay un montón de formas de graficar.

- Se pueden hacer animaciones. Ver `animation`
- Se pueden hacer gráficos interactivos.

  Widgets adaptativos.

  Muy bueno para cambiar los parametros del gráfico y lo veo

```
from matplotlib.widgets import Slider, Button,
RadioButtons
https://matplotlib.org/3.4.2/
```

Diferenciación numérica

Supongamos que conocemos y podemos evaluar a una función arbitraria $f(x)$.
Esta función puede ser:

- una función matemática de una línea,
- una función/mapa computacional de 100 líneas de código (con for, if etc), que no se puede traducir en una función analítica.

¿Como calculamos computacionalmente la derivada conociendo solo la función?

Serie de Taylor

Aplicamos como en el caso de aproximación de funciones la serie de Taylos alrededor de h,

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \cdots$$

Donde asumimos h pequeño.
Ahora no conocemos $f'(x)$ y lo queremos calcular a través de la función.
Despejando:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2!}f''(x) + \frac{h^2}{3!}f'''(x)$$

Error de la aproximación: $\sigma(h) \propto h$ proporcional a h.

Diferencias centradas

Si miro la serie de Taylor para $-h$:

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \cdots$$

Positiva:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \cdots$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}f'''(x) + \cdots$$

Se eliminó el término de la derivada segunda $\rightarrow$ Aproximación de segundo orden: $\sigma(h) \propto h^2$

Parcial

Jueves 30 de Setiembre. 1er Parcial.

Para rendir el 1er parcial deben enviar email a prog.fis.unne@gmail.com con Nombre y DNI.

Hasta el día Martes 28 de Setiembre.