

# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## plot(...) de matplotlib C1\_E1

"Pour effacer une figure, cliquez sur la figure et faire CTRL+W"

```
import numpy as np
import matplotlib.pyplot as plt
```

"Déclaration des paramètres de départs"

```
xi=0
xf=2*np.pi
npts=45
#ou
xi,xf,npts= 0,2*np.pi, 45
```

"Méthode 1"

```
x=np.linspace(xi,xf,npts)
```

"Méthode 2"

```
dx=(xf-xi)/(npts-1)
x2=np.arange(xi,xf+dx,dx)
```

"Est-ce que les 2 méthodes donnent le même résultat?"

```
erreur=abs(x-x2)
```

```
y1=np.cos(x)
y2=np.power(x,3)*0.01
```

```
plt.figure(1)
plt.plot(x,y1, '-r', label=r"$y=\cos(x)$")
plt.plot(x,y2, '+b', label=r"$y=0.01 \cdot x^3$")
```

"Ligne 34 à 39 facultatif et modifie l'apparence de la figure"

```
plt.xlabel('Axe des x')
plt.ylabel('Axe des y')
plt.title('Graphique d\'une fonction')
plt.legend()
plt.grid(axis='x')
plt.grid(axis='y')
```

### Markers

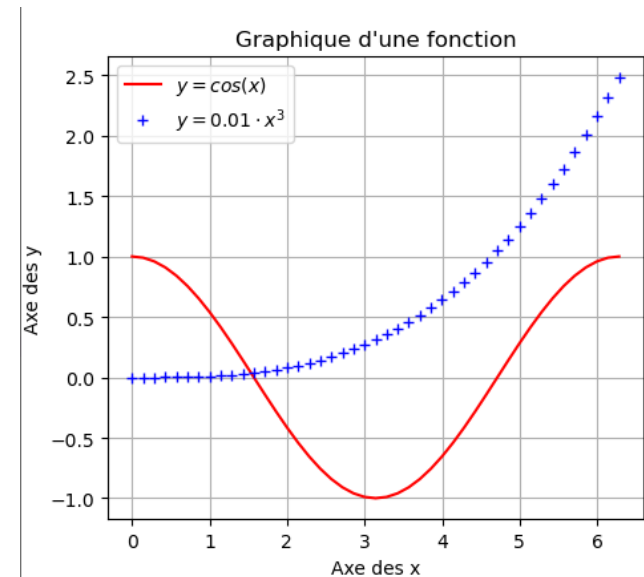
character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker

### Line Styles

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

[Liens](#) pour autre légendes



# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## Courbe paramétrique C1\_E2

```
import matplotlib.pyplot as plt
import numpy as np
```

```
ti=0
tf=2*np.pi
npts=125 #nbr de points
n=4 #paramètre de la rosace
t=np.linspace(ti,tf,npts)
```

"Méthode 1"

```
x=2*np.sin(n*t)*np.cos(t)
y=2*np.sin(n*t)*np.sin(t)
```

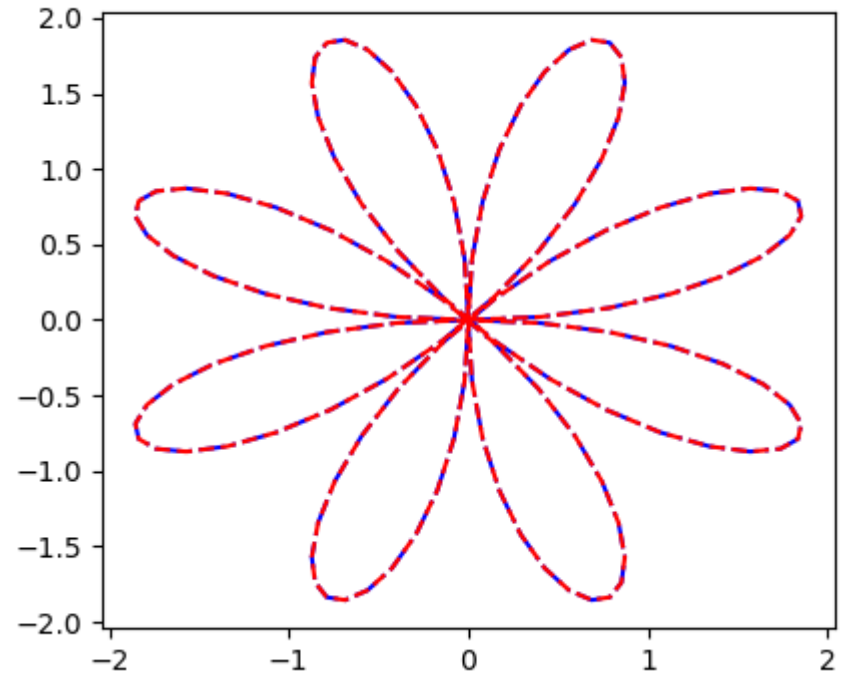
"Méthode 2"

```
x1, y1=np.zeros([len(t)]), np.zeros([len(t)])
for j in range(npts):
    x1[j]=2*np.sin(n*t[j])*np.cos(t[j])
    y1[j]=2*np.sin(n*t[j])*np.sin(t[j])
```

```
plt.plot(x,y,'-.b')
plt.plot(x1,y1,'--r')
```

Méthode traditionnelle,  
s'adapte dans d'autres langages  
de programmation

$$x = 2\sin(n \cdot t)\cos(t)$$
$$y = 2\sin(n \cdot t)\sin(t)$$



# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## Intégration numérique trapèzes composés C1\_E3

```
import numpy as np
import matplotlib.pyplot as plt
```

```
xi=0
xf=4
npts=25
x=np.linspace(xi,xf,npts) #0 à 4 avec 100 pts
```

```
#Méthode 1
y=np.power(x,3)
#Méthode 2
y=x**3
```

```
dx=x[1]-x[0] #pas dx constant dans cet exemple spécifique
```

```
"Méthode d'intégration avec forward integration"
```

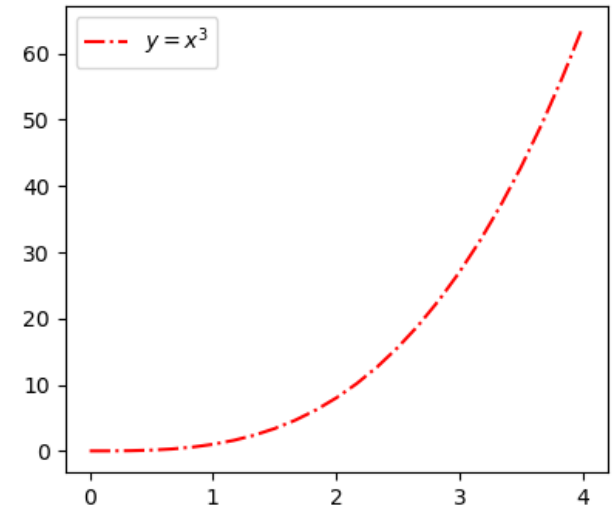
```
A1=0
for j in range(0,npts-1):
    A1= A1 + ( y[j]+y[j+1] )/2*dx
```

```
"Méthode d'intégration avec backward integration"
```

```
A2=0
for i in range(1,npts):
    A2=A2 + ( y[i]+ y[i-1] )/2*dx
```

```
print("L'intégration avec la méthode forward et backward sont :%g et %g" % (A1,A2))
plt.plot(x,y,'-.r',label=r"$y=x^3$")
plt.legend()
```

```
In [45]: runfile('C:/Users/richa/Desktop/MEC1315/2021_A/Python Matlab/Cours/C1_E3.py',
MEC1315/2021_A/Python Matlab/Cours')
L'intégration avec la méthode forward et backward sont :64.1111 et 64.1111
```



$$I = \int_0^4 x^3 dx = \left[ \frac{x^4}{4} \right]_0^4 = 64$$



# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## Dérivation numérique C1\_E4

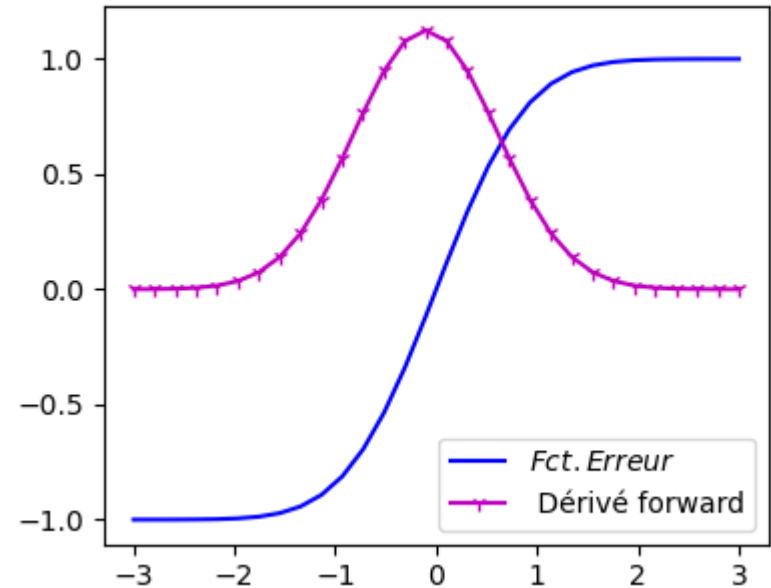
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import special #usage pour appeler la fonction erreur

xi,xf,npts=-3,3,30
x=np.linspace(xi,xf,npts)
y=special.erf(x)

dy_dx=np.empty(npts)
"Forward Dérivation "
for i in range(0,npts-1):
    dy_dx[i]= ( y[i+1]-y[i] ) / ( x[i+1] - x[i] )
"Pour le dernier points on applique la pente du point précédent"
dy_dx[npts-1]=dy_dx[i]

"Que faire si on applique un backward dérivation? "

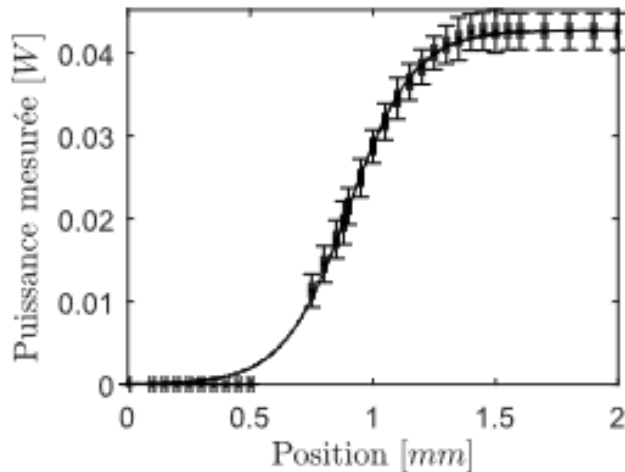
plt.plot(x,y,'-b',label=r"$Fct. Erreur$")
plt.plot(x,dy_dx,'-1m',label=r" Dérivé forward")
plt.legend()
```



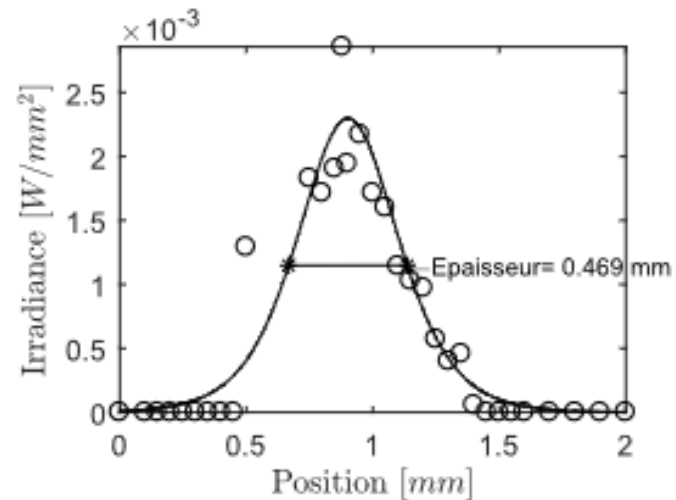
# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## Dérivation numérique Exemple d'application

Mesure de l'épaisseur d'une nappe laser pour la PIV



(c)



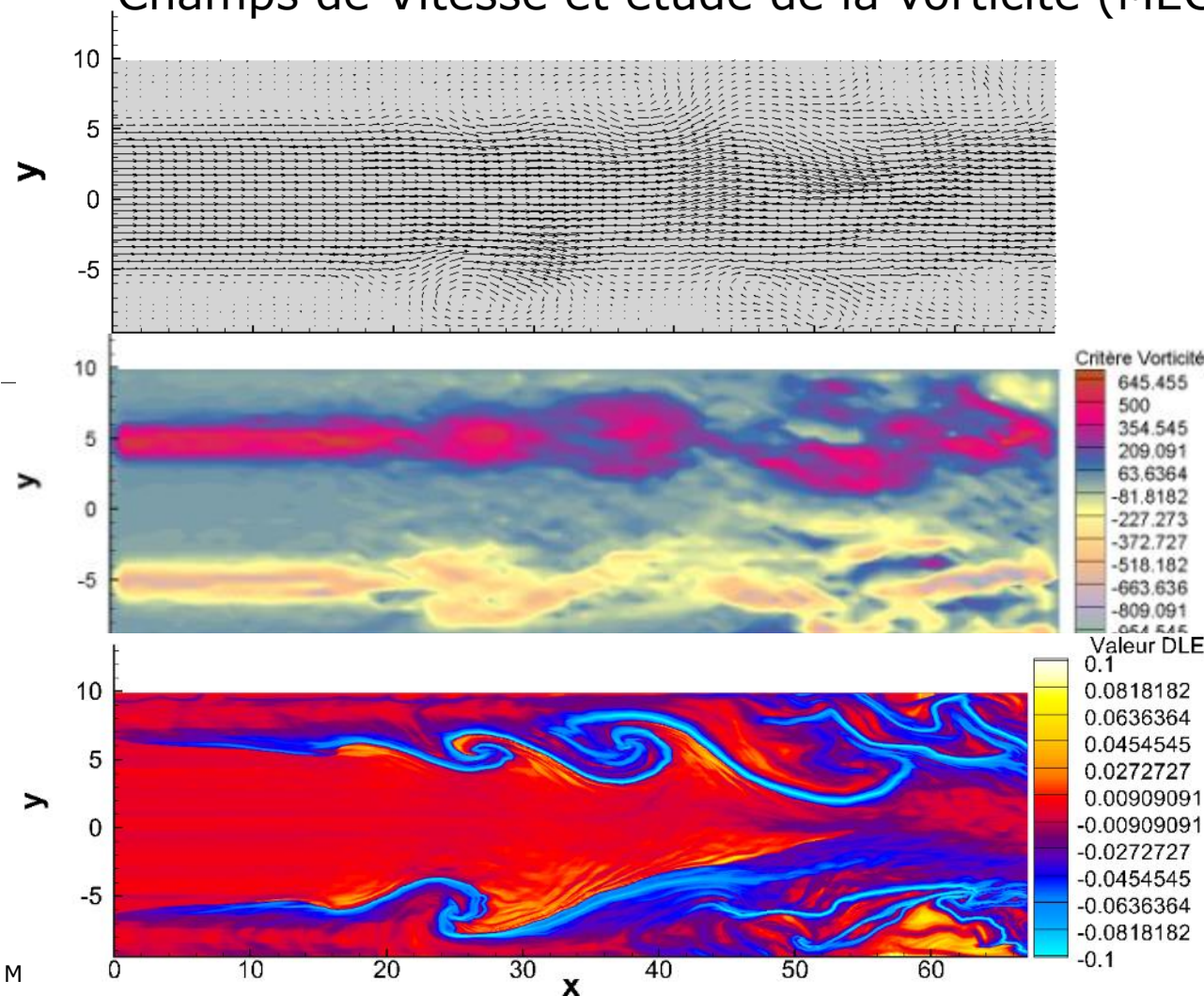
(d)



# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## Dérivation numérique Exemple d'application

Champs de Vitesse et étude de la vorticit  (MEC6617 J. V tel)



$$\omega = \nabla \times (\vec{u}) = \frac{\partial u_j}{\partial x_i} \epsilon_{ijk} (\vec{e})_k$$

$$A = \begin{pmatrix} \frac{\partial x_1}{\partial x_{20}} & \frac{\partial x_1}{\partial y_{20}} \\ \frac{\partial y_1}{\partial x_{20}} & \frac{\partial y_1}{\partial y_{20}} \end{pmatrix}$$

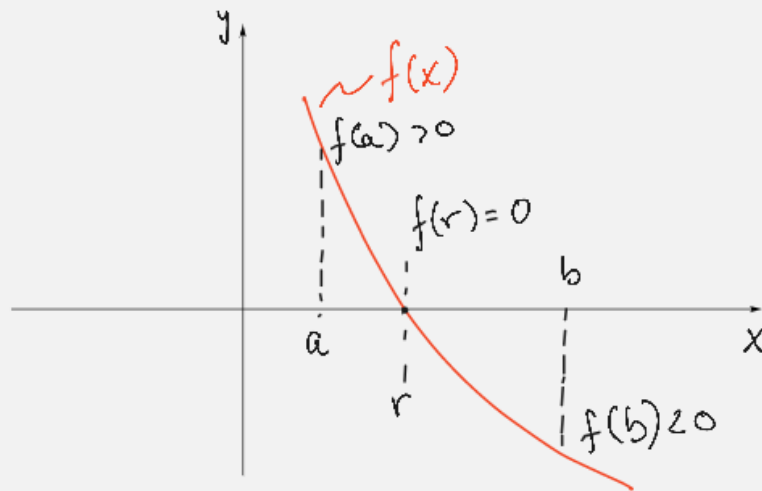


# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## Méthode de la bisection C1\_E5

### Algorithme de calcul

- Choisir un domaine qui contient seulement 1 racine  $r \in [a; b]$
- La fonction doit absolument croiser l'axe des  $x$ , à la racine
- Si  $f(x)$  croise l'axe des  $x$  à  $r$ , la fonction change de signe  $f(a) \cdot f(b) < 0$



# COURS C1 : GRAPHISME AVEC MATPLOTLIB DE PYTHON

## Méthode de la bisection C1\_E5

"Appel de la fonction bisection dans un autre fichier"

```
from my_fzeros import my_fzeros
import numpy as np
import matplotlib.pyplot as plt
```

"Déclaration d'une fonction mathématique à étudier"

```
def F(x):
    import numpy as np
    y=np.power(x,3)-x
    #y=np.exp(-x)*(3.2*np.sin(x)-0.5*np.cos(x))
    return y
```

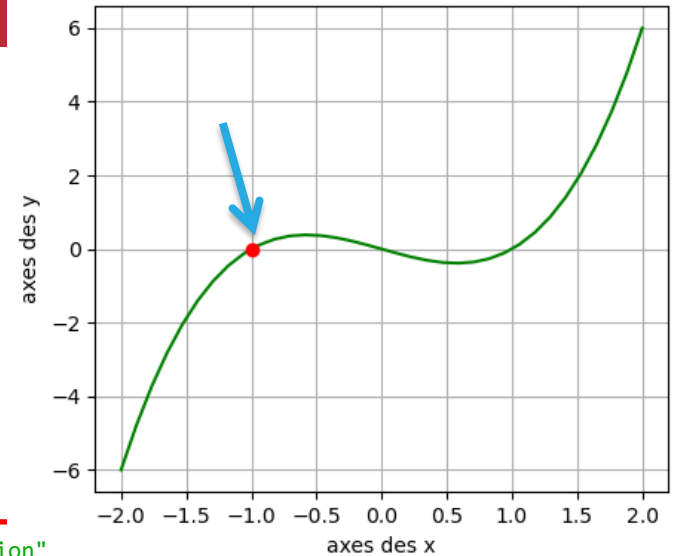
"Tracage d'une fonction"

```
x=np.linspace(-2,2,35)
y=F(x)
```

```
t=my_fzeros(F, 3.2, 3.4)
```

```
plt.figure(1)
plt.plot(x,y,'-g')
plt.plot(t,F(t),'or')
plt.xlabel('axes des x')
plt.ylabel('axes des y')
plt.grid('on')
```

Fichier python s'appellant my\_fzeros.py se trouvant au même niveau de repertoire du script C1\_E5



"Algorithme de Bisection"

```
def my_fzeros(f,xg,xd):
    fxg=f(xg)
    fxd=f(xd)
    "Évaluer l'algorithme si un 0 existe"
    if fxg*fxd>=0:
        print('Erreur, la fonction ne contient pas de zéros dans cet interval')
        r=[]
    else:
        r=(xg+xd)/2
        "En fonction de quel bord le zéro, les bornes s'ajuste"
        while abs(xg-xd)/2>1e-5:
            r=(xg+xd)/2
            fr=f(r)
            if fxg*fr<0:
                xd=r
            else:
                xg=r
        return r
```