matplotlib et les graphes

1. Outils graphiques	
Représentation graphique d'un nuage de points.	Utiliser les fonctions de base de la bibliothèque **matplotlib** pour représenter un nuage de points.
Représentation graphique d'une fonction.	Utiliser les fonctions de base de la bibliothèque **matplotlib** pour tracer la courbe représentative d'une fonction.
Courbes planes paramétrées.	Utiliser les fonctions de base de la bibliothèque **matplotlib** pour tracer une courbe plane paramétrée.

Mémento

Syntaxe	Rôle
	importation des modules
import numpy as np	importe le module *numpy* et crée l'alias np sur les fictions de ce module
from math import *	importe l'ensemble du module math sans alias, non recommandé
import matplotlib.pyplot as plt	importe le sous module pyplot contenant les fonctions graphiques
	tracé d'un nuage de points
plt.plot(xi, yi, '+r')	affiche le nuage de points de coordonnées xi, yi avec des croix "+" rouges non-reliées
	tracé d'une fonction
> def f(x): # Attention à l'indentation	définition de la fonction à tracer (ici $x \to \sin(x^2) - 1$)
`>  return np.sin(x**2)-1`	
xi = np.linspace(-5, 5, 10**3)	création d'une liste de 1000 valeurs équiréparties sur $[-5;5]$
plt.plot(xi, f(xi), '-b')	Tracé des points de coordonnées *xi,f(xi)* avec une ligne bleue
	tracé d'une courbe paramétrée
ti = np.linspace(0, 10, 10**3)	liste des valeurs de dates t_i sur l'intervalle $[0;10]$
xi = np.cos(2*ti) yi = np.sin(3*ti)	calcul des coordonnées *xi,yi* des points pour toutes les valeurs du paramètre t_i.
plt.plot(xi, f(xi), '-k')	tracé de la courbe paramétrée,les points étant reliés par un trait noir
	complément : mise en forme
plt.title("Titre du graphe")	Titre du graphe
plt.xlabel("Titre de l'axe X")	Titre de l'axe horizontal
`plt.ylabel(r"Grandeur Y : $\alpha\$ (rad)")`	Utilisation de LaTeX (symbole grec de la lettre alpha) dans du texte Python

1 Importation des modules

matplotlib.pyplot est une collection de fonctions qui nous sont utiles pour effectuer des représentations graphiques en Python.

Ces fonctions ne sont pas accessibles par défaut, il est nécessaire des les importer.

Il existe trois syntaxes pour **importer un module** en Python.

Supposons que l'on souhaite utiliser la fonction `sin` du module *math* pour calculer $sin(x)$ avec $x = 2.0$.

Méthode 1:

```
from math import * # importe l'ensemble des fonctions du module math
```

On fera ensuite `sin(2)`

Méthode 2:

```
import math # importe le module math
```

On fera ensuite `math.sin(2)`

Méthode 3:

```
import math as mt # importe le module math en créant un alias
```

On fera ensuite `mt.sin(2)`. *mt* est un **alias** permettant d'accéder aux fonction du module *math*.

[5]:
```
import matplotlib.pyplot as plt # création de l'alias plt qui désigne la collection des fonctions de pyplot

import numpy as np # création de l'alias np qui désigne la collection des fonctions Numpy
```

2 Tracé d'une fonction

Soit la fonction $f : x \mapsto f(x) = sin(x)$ dont on souhaite tracer la représentation graphique sur l'intervalle $I = [-2;2]$.

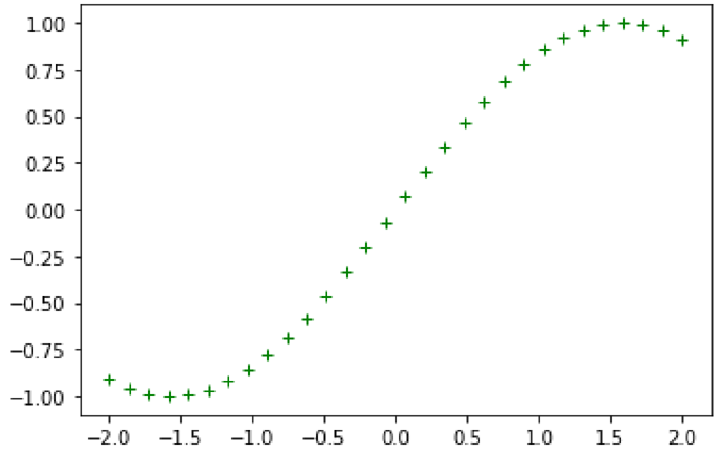
Le tracé se fait en trois étapes: > (1) Création de la liste des valeurs de la variable x, représentation les abscisses.

(2) Calcul de la liste des ordonnées y pour chaque valeur de x sous la forme $y = f(x)$.

(3) Appel de la fonction `plot(xi,yi)` de pyplot permettant de tracer l'ensemble des points M_i dont les coordonnées sont les couples (x_i, y_i).

[16]:
```
xi = np.linspace(-2,2,30) # la fonction linspace renvoie 30 valeurs équidistribuées en -2 et +2 (bornes incluses)

yi = np.sin(xi) # calcul de la liste des ordonnées yi correspondant à chaque xi

plt.plot(xi,yi,"+g") # les symboles sont des croix vertes (g = green)
```

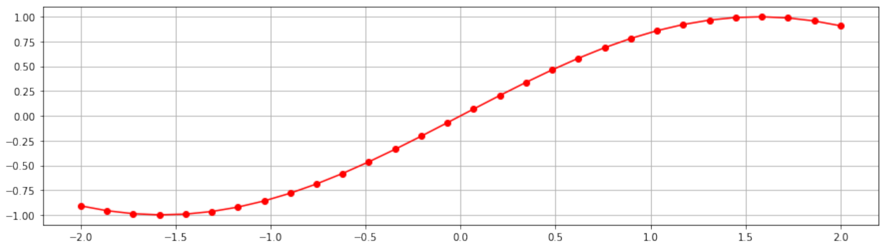
[16]:
```
[<matplotlib.lines.Line2D at 0x26b2480fb88>]
```

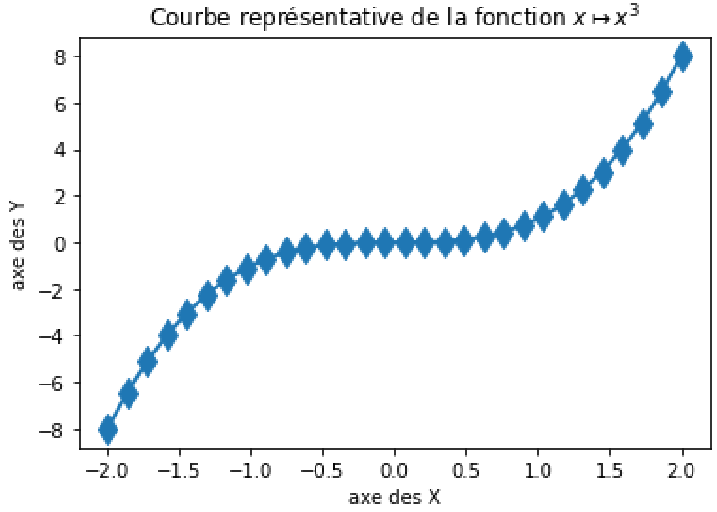


Améliorations

Il est possible : - de relier les points, - de changer les symboles, - de changer la taille de la figure, - de changer la taille du trait, - d'ajouter une grille, - d'ajouter un titre aux axes.

[37]:
```
# 1er graphe
plt.figure(figsize = (15,4)) # 15 pouces de largeur, 4 pouces de hauteur
plt.plot(xi,np.sin(xi),'-or') # symboles o = ronds, - reliés, r = red
plt.grid() # ajout d'une grille
# 2ème graphe
plt.figure() # création d'un nouvelle figure
plt.plot(xi,xi**3,'-d',lw = 2, ms = 11) # symboles d = diamant, lw = linewidth, ms = markersize
plt.xlabel('axe des X') # titre de l'axe horizontal
plt.ylabel('axe des Y') # titre de l'axe vertical
plt.title(r'Courbe représentative de la fonction $x\mapsto x^3$') # titre du graphe, r = mise en forme
plt.show() # pour afficher le graphique, non nécessaire avec le notebook Jupyter
```



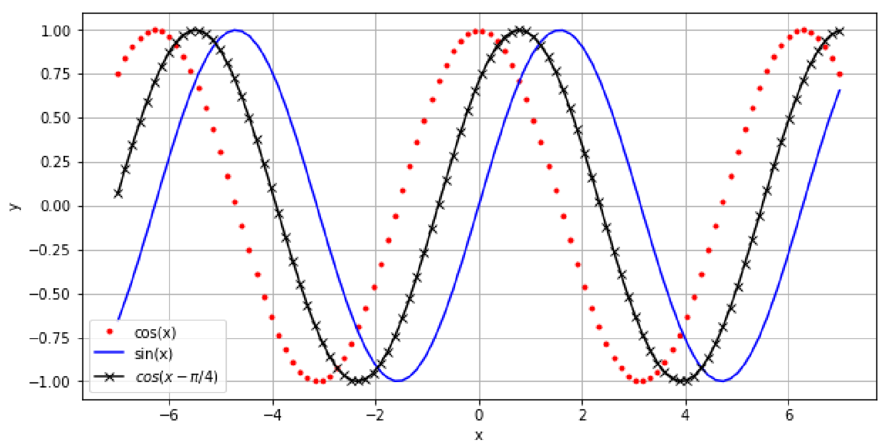


2.1 Tracé de plusieurs courbes sur le même graphe

Le paramètre *label* de la fonction `plot` permet d'ajouter une légende à chaque courbe.

[60]:
```
xi = np.linspace(-7,7,100) # intervalle de 100 valeurs équiréparties en -7 et 7
y1 = np.cos(xi) # cosinus
y2 = np.sin(xi) # sinus
y3 = np.cos(xi-np.pi/4) # cos(x-pi/4) = déphasage de 45°
plt.figure(figsize=(10,5))
plt.plot(xi,y1,'.r',label='cos(x)') # .r = points rouges ; label = renseigne la légende
plt.plot(xi,y2,'-b',label='sin(x)') # trait bleu
plt.plot(xi,y3,'x-k',label=r'$cos(x-\pi/4)$') # croix noires reliées; r devant le texte = formate le LaTeX
plt.grid()      # ajout d'une grille
plt.xlabel('x') # titre de l'axe des x
plt.ylabel('y') # titre de l'axe des y
plt.legend()    # ajoute la légende
```

[60]:
```
<matplotlib.legend.Legend at 0x26b28c4ec48>
```



2.1.1 Exercice N1 n°1 : tracé de fonctions

a) Compléter ci-dessous les trois instructions permettant de tracer, sur l'intervalle $[-5;5]$, le graphe de la fonction:

$$x \mapsto \cos(10x)e^{-x^2}$$

[ ]:
```
xi = # listes des valeurs d'abscisses sur l'intervalle [-5;5]

yi = # calcul des valeurs d'ordonnées yi correspondant à chaque xi

 # appel à la fonction plot du module pyplot
```

b) Compléter les instructions précédentes afin d'ajouter la représentation graphique des fonctions:

- $f_2 : x \mapsto e^{-x^2}$ en rouge
- $f_3 : x \mapsto -e^{-x^2}$ en vert

3 Tracé d'un nuage de points

La matrice 4x2 (4 lignes, 2 colonnes) suivante représente les coordonnées des quatre sommets d'un rectangle.

$$M = \begin{pmatrix} 0 & 0 \\ 2 & 0 \\ 2 & 1 \\ 0 & 1 \end{pmatrix}$$

L'instruction ci-dessous définit la matrice M en Python.

[81]:
```
M = np.array([[0,0], [2,0], [2,1], [0,1]]) # bien noter l'écriture des crochets !
print(M) # affichage de la matrice
```



```
print('type de la variable M  : ', type(M)) # une matrice est un objet de type␣
↪ndarray
```

```
[[0 0]
 [2 0]
 [2 1]
 [0 1]]
type de la variable M  :  <class 'numpy.ndarray'>
```

Chaque ligne donne les coordonnées d'un point:

- Les valeurs des abscisses sont situées dans la première colonne.
- Les valeurs des ordonnées sont situées dans la deuxième colonne.

Comment tracer le rectangle défini par ces quatre points?

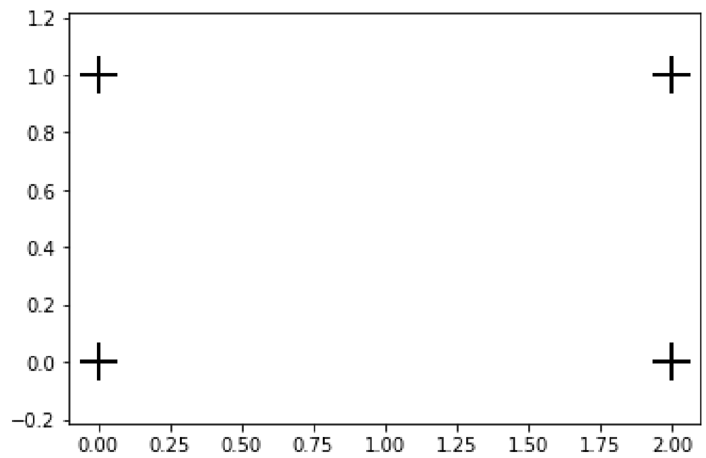
(1) On récupère la liste des abscisses x_i

(2) On récupère la liste des ordonnées y_i

(3) On utilise la fonction plot de pyplot pour afficher le **nuage des points** de coordonnées (x_i, y_i).

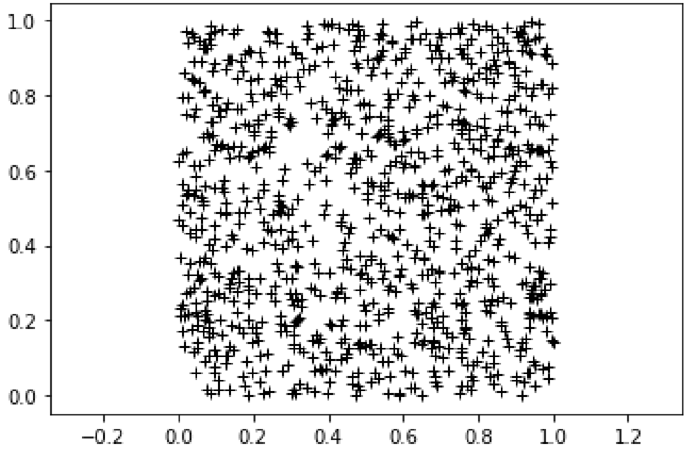
[95]:
```
xi = M[:,0] # extraction de la première colonne, son indice est zéro

yi = M[:,1] # extraction de la deuxième colonne, son indice est un

plt.plot(xi,yi,'+k',ms = 20,mew = 2) # croix = en noir, markersize = 12 ;␣
↪markerEdgeWidth = 2

plt.axis('equal') # impose que l'échelle des x et l'échelle des y soient identiques
```

[95]: (-0.1, 2.1, -0.05, 1.05)



Exemple 1 : nuage de points aléatoires Utilisation de la fonction `np.random.rand()`

[119]:
```python
N = 1000 # nb de points
# création d'une matrice N x 2 dont les valeurs sont uniformément distribuées dans␣
↪l'intervalle [0;1]
M1 = np.random.rand(N,2)
plt.plot(M1[:,0],M1[:,1],'+k') # affiche du nuage de points avec des croix + noires
plt.axis('equal') # impose que l'échelle des x et l'échelle des y soient identiques
plt.show()
```



Exemple 2 : étoile à 5 branches. Principe:

- Les points de l'étoiles sont réparties sur un cercle de rayon unité.
- Les coordonnées des points sont données par $x_i = \cos(\theta_i)$ et $y_i = \sin(\theta_i)$
- Les angles θ_i sont régulièrement distribués sur l'intervalle $[0;2\pi]$. On peut obtenir les θ_i par la relation:

$$\theta_i = \frac{2\pi}{5} \times i, \quad \text{avec } i = 0,1,\ldots,4$$

- On trace l'étoile en reliant un point sur deux.

[167]:
```python
# Etape 1: création de la liste des angles theta_i
thetai = [2*np.pi/5*k for k in range(5)]
print(thetai) # on peut vérifier que le tableau contient 5 valeurs.
```

```
[0.0, 1.2566370614359172, 2.5132741228718345, 3.7699111843077517,
5.026548245743669]
```

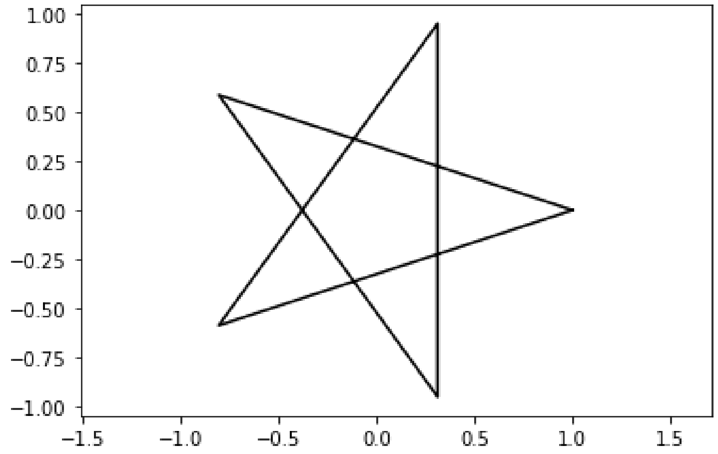
[168]:
```python
# Etape 2: calcul des valeurs des abscisses xi et des ordonnées yi
xi = np.cos(thetai)
yi = np.sin(thetai)
```

[170]:
```python
# Etape 3: on extrait un point sur 2
indices = [0,2,4,1,3,0] # indices des points obtenus en sautant un point sur 2
```



```python
# Etape 4: on relie les points
plt.plot(xi[indices],yi[indices],'-k')
plt.axis('equal')
```

[170]:
```
(-0.8994678440936948,
 1.0904508497187473,
 -1.046162167924669,
 1.0461621679246689)
```



Autre méthode pour créer la liste des indices

Il est possible de rendre systématique la recherche des indices successifs 0,2,4,1,3,0 en réalisant la liste `L=[0,1,2,3,4,0,1,2,3,4,0]` puis en prenant 1 sur 2 des termes de cette liste doublée en réalisant un *slicing* grâce à l'instruction `L[::2]`.

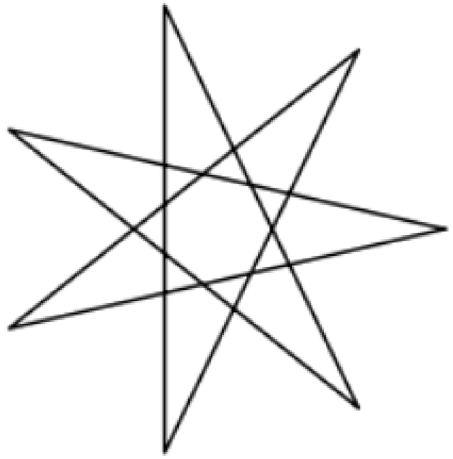
- On crée une liste d'indice de zéro à (N-1) que l'on concatène avec elle-même (opération *2)
- On ne prélève qu'un indice sur deux (1 sur 2) de cette liste grâce à un **slicing** (cf cours d'informatique pour tous).
- On n'oublie pas d'**ajouter le zéro** en fin de liste pour réaliser une boucle fermée.

[1]:
```python
# Etape 3bis: extraction d'un d'indice sur deux grâce à un slicing
indices = [k  for k in range(7) ]*2 +[0] # 3 listes d'indices + zéro
indices = indices[::2] # on extrait un point sur trois de cette liste
print(indices)
```

```
[0, 2, 4, 6, 1, 3, 5, 0]
```

3.1 Exercice N1 n°2 : étoile à 7 branches

a) Sur le même principe, écrire ci-dessous les instructions permettant de tracer une étoile à 7 branches.



On remarquera que l'on doit "sauter" un indice sur trois pour relier les points.

Aide : pour obtenir la liste des indices, on pourra utiliser les instructions suivantes:

[179]:
```
# Etape 3 : extraction d'un d'indice sur trois grâce à un slicing
indices = [k  for k in range(7) ]*3 +[0] # 3 listes d'indices
indices = indices[::3] # on extrait un point sur trois de cette liste
print(indices)
```

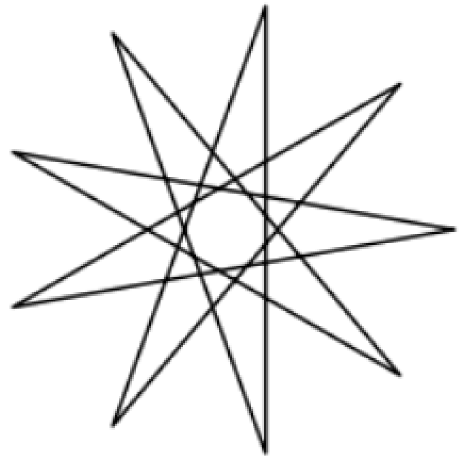
```
[0, 3, 6, 2, 5, 1, 4, 0]
```

[180]:
```
# Etape 1: création de la liste des angles theta_i
# à compléter...

# Etape 2: calcul des valeurs des abscisses xi et des ordonnées yi

# Etape 3 : extraction d'un d'indice sur trois grâce à un slicing
# à compléter...

# Etape 4: affichage du nuage de points
# à compléter...
```

b) Généralisation (difficile !)

Ecrire la suite d'instructions permettant de tracer une étoile à $N = 2k + 1$ branches où k est un entier positif

Par exemple, pour $k = 4$, voici ci-dessous une étoile à neuf branches. On remarquera que l'on "saute" les indices de $k = 4$ en $k = 4$.



3.2 Courbes planes paramétrées

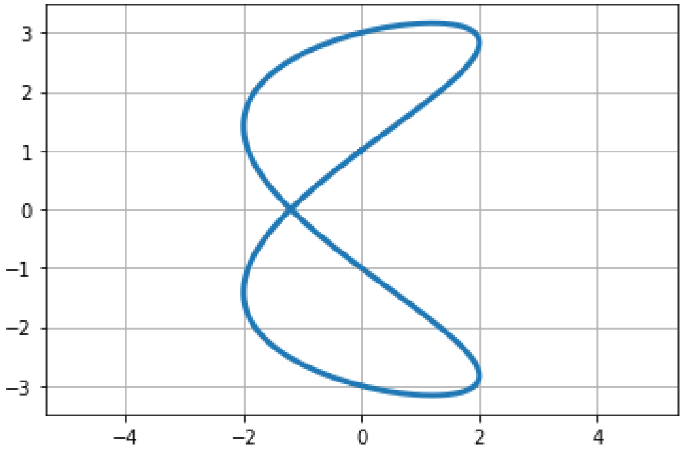
Exemple : On considère l'ensemble de points du plan $M(t)$ dont les coordonnées cartésiennes $(x(t), y(t))$ sont données par les expressions suivantes:

$$\begin{cases} x(t) &= 4\cos(t)\sin(t) \\ y(t) &= 3\sin(t) + \cos(t) \end{cases}$$

Lorsque la grandeur t décrit l'intervalle $[0; 2\pi]$, les coordonnées $x(t)$ et $y(t)$ du point varient et décrivent une courbe du plan.

Les instructions ci-dessous permettent de tracer cette courbe paramétrée.

[268]:
```
# Etape 0 : import des modules
import matplotlib.pyplot as plt
import numpy as np
# Etape 1 : création de la liste des valeurs de la variable t
t = np.linspace(0,2*np.pi,10**3) # 1000 valeurs régulièrement réparties dans␣
 ↪l'intervalle
# Etape 2 : calcul des coordonnées xi,yi des points M_i de la courbe
xi = 4*np.cos(t)*np.sin(t)
yi = 3*np.sin(t)+np.cos(t)
# Etape 3 : appel de la fonction plot de pyplot
plt.plot(xi,yi,lw = 3) # tracé du nuage de points, lineWidth =3
plt.grid()
plt.axis('equal') # axes "carrés"
plt.show()
```



3.3 Exercice N1 n°3 : équation paramétrique d'un cercle de rayon R (exo corrigé)

Compléter les instructions suivantes méthode pour tracer la courbe paramétrée suivante, la variable t variant dans l'intervalle $[0;2\pi]$.

$$\begin{cases} x(t) &= R\cos(t) \\ y(t) &= R\sin(t) \end{cases}$$

On prendre $R = 2.0$

[ ]:
```
R= 2.0 # Valeur du rayon du cercle
# Etape 1 : création de la liste des valeurs de la variable t
t =  # 1000 valeurs régulièrement réparties dans l'intervalle
# Etape 2 : calcul des coordonnées xi,yi des points M_i de la courbe
xi = R*np.cos(t) # valeurs xi
yi =             # valeurs yi
# Etape 3 : appel de la fonction plot de pyplot
plt.plot(...) # tracé du nuage de points, lineWidth =3
plt.grid()
plt.axis('equal') # axes "carrés"
plt.show()
```

Correction N1 n°3

[ ]:
```
R= 2.0 # Valeur du rayon du cercle
# Etape 1 : création de la liste des valeurs de la variable t
t = np.linspace(0,2*np.pi,10**3) # 1000 valeurs régulièrement réparties dans␣
 ↪l'intervalle
# Etape 2 : calcul des coordonnées xi,yi des points M_i de la courbe
xi = R*np.cos(t)
yi = R*np.sin(t)
# Etape 3 : appel de la fonction plot de pyplot
plt.plot(xi,yi,lw = 3) # tracé du nuage de points, lineWidth =3
plt.grid()
plt.axis('equal') # axes "carrés"
```



```
plt.show()
```

3.4 Exercice N1 n°4: équation paramétrique d'une ellipse (à chercher)

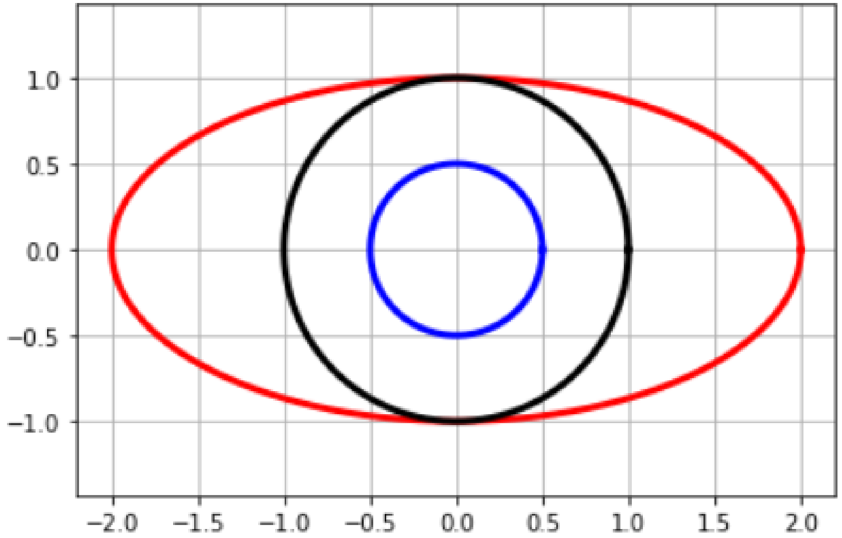
Voici l'équation d'une ellipse dont les axes sont horizontaux et verticaux. La longueur du demi-grand axe est $a = 2.0$, le demi-petit axe a pour longueur $b = 0.8$.

La variable t varie dans l'intervalle $[0;2\pi]$.

$$\begin{cases} x(t) &= a\cos(t) \\ y(t) &= b\sin(t) \end{cases}$$

a) Ecrire le code Python permettant de tracer cette ellipse.

b) **Challenge de l'oeil** (! difficile) : Ecrire la liste d'instructions permettant de réaliser la figure ci-dessous.



3.5 Exercice N1 n°5 : Jet parabolique (exo corrigé)

La trajectoire d'un point M du plan est décrite par l'équation paramétrique suivante:

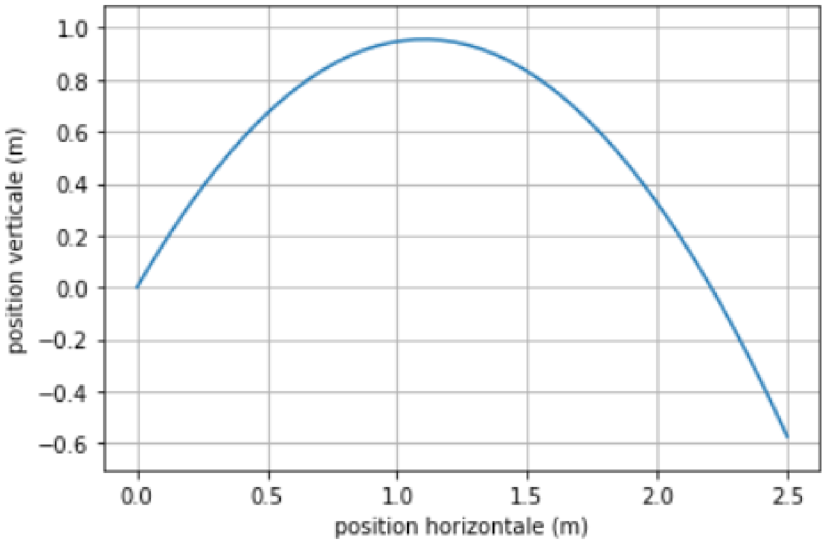
$$\begin{cases} x(t) &= V_0\cos(\alpha)\times t \\ y(t) &= V_0\sin(\alpha)\times t - \frac{1}{2}g\times t^2 \end{cases}$$

On donne $g = 9,81$ m.s^{-2}, $\alpha = 60$, $V_0 = 5,0$ m.s^{-1}.

La date t variera entre zéro et une seconde.

a) Compléter la liste d'instructions ci-dessous permettant de tracer la trajectoire du point M.

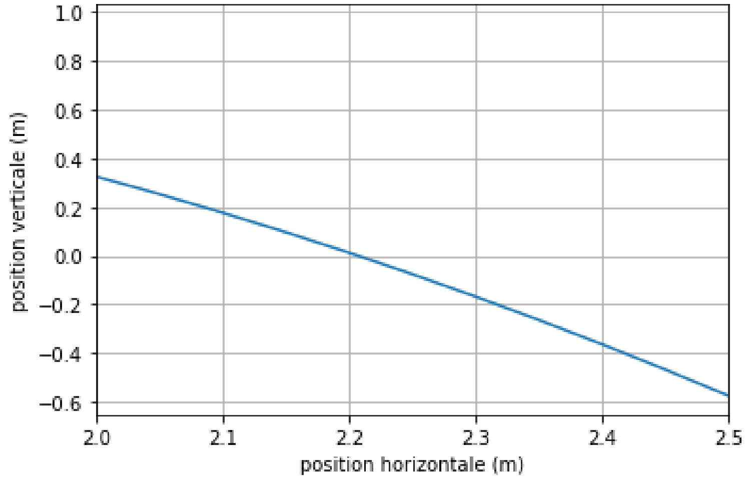
La figure a obtenir est la suivante :



[ ]:
```
## A COMPLETER
V0 = 5.0 # m/s
g = 9.81 # m/s^2
alpha = # valeur de l'angle à compléter (ATTENTION : degrés et radians)
t =  # listes des dates
xi = # listes des abscisses
yi = # listes des abscisses
# affichages : plot, grille, titres des axes
```

Correction N1 n°5

[207]:
```
V0 = 5.0 # m/s
g = 9.81 # m/s^2
alpha = 60*np.pi/180 # à compléter
t = np.linspace(0,1) # listes des dates
xi = V0*np.cos(alpha)*t # listes des abscisses
yi = V0*np.sin(alpha)*t-g/2*t**2 # listes des abscisses
plt.plot(xi,yi)
plt.grid()
plt.axis('equal')
plt.xlabel('position horizontale (m)')
plt.ylabel('position verticale (m)')
plt.show()
```

b) Estimer par lecture graphique la portée du tir, c'est-à-dire la valeur de l'abscisse correspondant à l'annulation de l'ordonnée.

Note: on pourra utiliser la fonction `plt.xlim([a,b])` permettant de faire un zoom sur les abscisses (penser à supprimer la fonction `plt.axis('equal')`

[ ]:
```
```

3.6 EXERCICES D'ENTRAINEMENT

3.7 N1 n°6 Représentation graphique d'une fonction (niveau facile)

a) Tracer la représentation graphique de la fonction numérique

$$f : x \mapsto x(1-x)$$

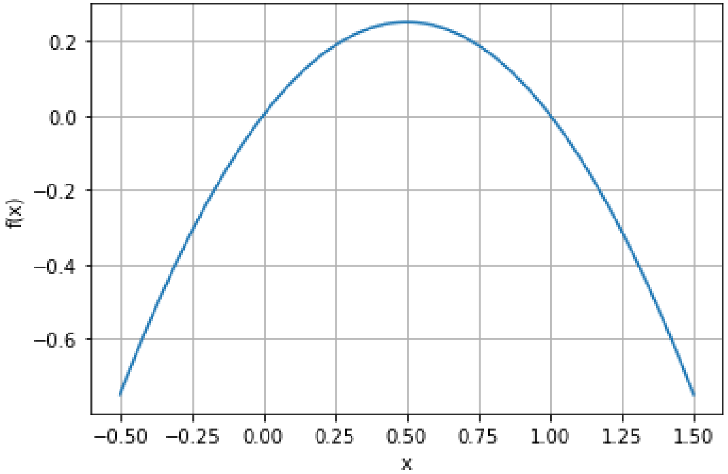
pour x compris entre -0,5 et 1,5.

b) Déterminer graphiquement pour quelle valeur de x la fonction f est maximale. Quelle est la valeur de ce maximum?

c) Retrouver les résultats de la question b) par le calcul.

[213]:
```
# Etape 0 : import des modules
import matplotlib.pyplot as plt
import numpy as np
# Etape 1 : création de la liste des valeurs de la variable x
x = np.linspace(-0.5,1.5,10**3) # 1000 valeurs régulièrement réparties dans␣
 ↪l'intervalle
# Etape 2 : calcul des valeur de puissance
y = x*(1-x)
# Etape 3 : appel de la fonction plot de pyplot
plt.plot(x,y) # tracé du nuage de points
plt.xlabel('x')
plt.ylabel('f(x)')
```

```
plt.grid()
```



Par lecture graphique, la fonction est maximale pour $x \approx 0,5$, le maximum vaut $f(0,5) = 0,25$.

On retrouve ce résultat en calcul la dérivée $(uv)' = u'v + v'u$:

$$f'(x) = (1-x) + (-1) \times x = 1 - 2x$$

La dérivée s'annule pour

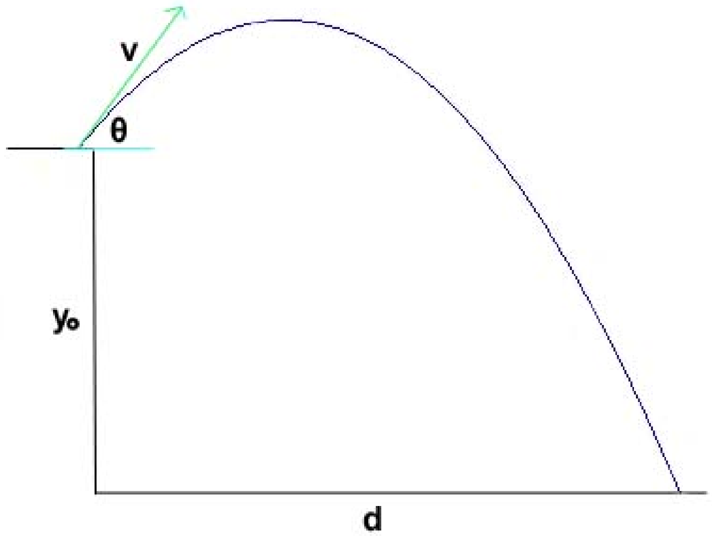
$$f'(x) = 0 \iff x = 1/2$$

Le maximum vaut $f(1/2) = 1/4$.

Remarque : en toute rigueur, pour prouver que la fonction est maximale en $x = 1/2$, il faut vérifier que sa dérivée seconde est posive en $x = 1/2$. Ce qui est bien le cas car $f''(x) = 1 > 0$

3.8 N1 n°7 Représentation graphique d'une fonction (niveau moyen)

La *portée d'un projectile* correspond à la distance horizontale du point où le projectile est lâché par le système lui donnant sa vitesse initiale et la projection horizontale du point de chute du projectile (cf schéma).



Pour un jet sans frottement dans un champ de pesanteur g uniforme, la portée, notée d, est une distance qui dépend:

- de la vitesse initiale v du projectile ,
- de l'angle θ que fait le vecteur vitesse initiale avec la direction horizontale,
- de la hauteur y_0 du projectile par rapport au sol.

L'expression de la portée d est la suivante:

$$d = \frac{v\cos(\theta)}{g}\left(v\sin(\theta) + \sqrt{(v\sin\theta)^2 + 2gy_0}\right)$$

Dans toute la suite, on fixe les valeurs suivantes:

- $v = 10,0$ m.s^{-1} la vitesse initiale,
- $g = 9,81$ m.s^{-2} l'intensité de la pesanteur,
- $y_0 = 5,0$ m la hauteur initiale par rapport au sol.

a) Tracer l'évolution de la distance d en fonction de l'angle θ, pour θ variant de zéro à 90° (on s'aidera du script suivant que l'on complétera).

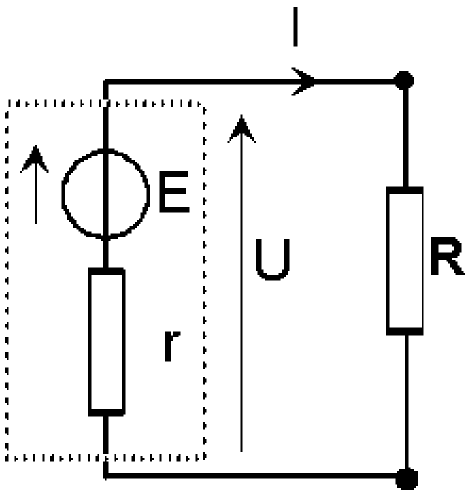
b) En déduire graphiquement la valeur de l'angle qui donne la portée maximale (on estimera la valeur de l'angle à 1 degré près).

[ ]:
```
# script à compléter
v = 10.  # m/s
g = 9.81 # m.s^-2
y0 = 5.  # m
# Etape 0 : import des modules

# Etape 1 : création de la liste des valeurs de la variable theta, en degrés
theta =  # valeurs régulièrement réparties dans l'intervalle
# Etape 2 : calcul de la flèche d pour toutes les valeurs de theta
d =
```

```
# Etape 3 : appel de la fonction plot de pyplot

plt.xlabel('theta (degres)') # titre de l'axe des x
# titre de l'axe des y
plt.grid()
```

3.9 N1 n°8 Représentation graphique d'une fonction (niveau difficile)

On considère le circuit électrique suivant comportant une source idéale de tension E et deux résistors r et R associés en série.



On s'intéresse à la puissance $\mathscr{P}_R$ dissipée par la résistance R lorsque l'on fait **varier la valeur de** R, les grandeurs E et r **étant maintenues constantes**.

L'intensité du courant circulant dans le circuit est

$$I = \frac{E}{R+r}$$

La puissance dissipée par la résistance R est donnée par l'expression:

$$\mathscr{P}_R = RI^2$$

a) Etablir l'expression de la puissance $\mathscr{P}_R$ en fonction des grandeurs E, R et r.

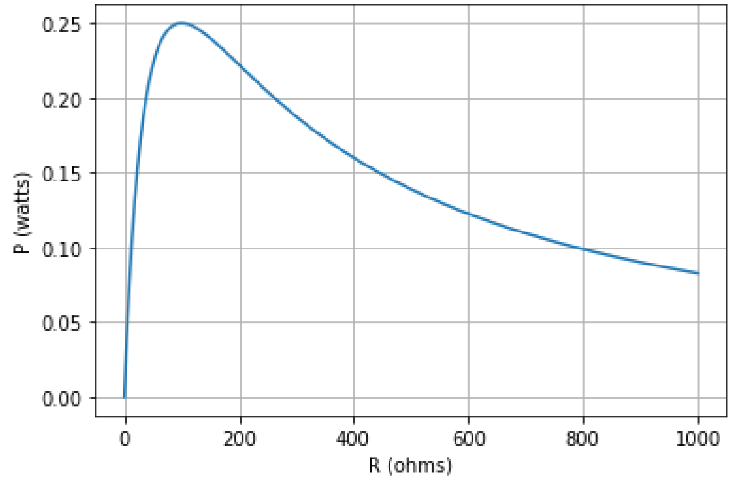
On fixe les valeurs suivantes: $E = 10$ V, $r = 100\ \Omega$.

b) Tracer l'évolution de la puissance $\mathscr{P}_R$ en fonction de la valeur de la résistance R, pour R variant de zéro à $10r$.

[232]:
```
# Définition des constantes
E = 10. # volts
r = 100 # ohms
# Etape 0 : import des modules
import matplotlib.pyplot as plt
import numpy as np
```



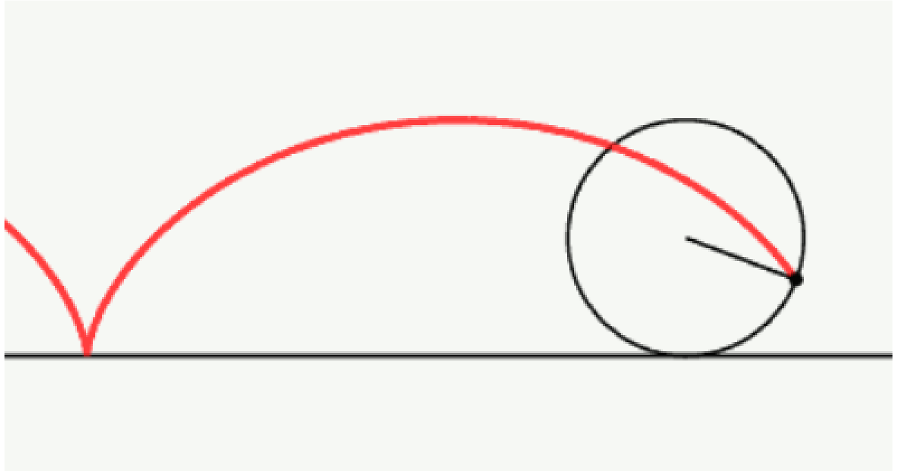
```
# Etape 1 : création de la liste des valeurs de la variable R
R = np.linspace(0,10*r,10**3) # 1000 valeurs régulièrement réparties dans␣
↪l'intervalle
# Etape 2 : calcul des valeur de puissance
P = R*(E/(R+r))**2 # P = R x I^2 = R (E/(R+r))^2
# Etape 3 : appel de la fonction plot de pyplot
plt.plot(R,P) # tracé du nuage de points
plt.xlabel('R (ohms)')
plt.ylabel('P (watts)')
plt.grid()
```



c) En déduire graphiquement la valeur de la résistance R telle que la puissance $\mathscr{P}_R$ dissipée par la résistance R soit maximale.

3.10 Exercice N1 n°9 équation de la cycloïde (niveau facile)

On appelle cycloïde (ou cycloïde droite) la courbe plane qui correspond à la trajectoire d'un point fixé à un cercle qui roule sans glisser sur une droite (cf figure et wikipedia https://fr.wikipedia.org/wiki/Cyclo%C3%AFde).



Le point mobile engendre une cycloïde droite.

L'équation paramétrique en coordonnées cartésienne de la cycloïde est la suivante:

$$\begin{cases} x(\theta) & = R(\theta - \sin(\theta)) \\ y(\theta) & = R(1 - \cos(\theta)) \end{cases}$$

a) Représenter graphiquement une telle courbe pour la variable θ évoluant de zéro à 4π. On prendra un rayon $R = 1$ m. On s'aidera du script suivant.

[ ]:
```
# Définition des constantes
R = 1. # m
# Etape 0 : import des modules

# Etape 1 : création de la liste des valeurs de la variable theta

# Etape 2 : alcul des coordonnées des points de la courbe

# Etape 3 : appel de la fonction plot de pyplot
plt.figure(figsize=(12,2)) # figure dans lequel la largeur a été allongée
# tracé du nuage de points
plt.axis('equal') # même échelle pour l'axe des x et celui des y.
plt.grid()
```

3.11 Exercice N1 n°10 spirale logarithmique (niveau moyen)

Une spirale logarithmique possède une équation paramétrique en coordonnées cartésienne de la forme:

$$\begin{cases} x(\theta) & = ab^{\theta}\cos(\theta) \\ y(\theta) & = ab^{\theta}\sin(\theta) \end{cases}$$

Les grandeurs a et b étant des constantes. On retrouve la spirale logarithmique dans le développement de certaines **coquilles de mollusque** (cf figure ci-dessous) ou dans l'agencement de certaines

fleurs.

a) Tracer une telle courbe pour les valeurs suivantes :

$$a = 1$$

$$b = \exp\left(\frac{\ln(\varphi)}{\pi/2}\right)$$

où $\varphi = \frac{1+\sqrt(5)}{2}$ est le *nombre d'or*.

Remarque : en python, le *logarithme népérien* est noté `log`. Il est donc appelé par la fonction `np.log()` du module Numpy.

La variable θ variera entre -5π et π.

Voici l'allure de la courbe à obtenir.