Informatique **CPGE**
2^{de} année Clemenceau

Tracés de courbes avec `Python`

Objectifs :

- Utilisation du module `matplotlib.pyplot`.
- Révisions sur les tracés de courbes.
- Complément sur l'habillage.
- Complément sur quelques courbes particulières.

Le tracé de courbes est une activité fréquente en informatique pour tous, et surtout dans ses applications pour les autres matières enseignées. Ce folio a l'ambition de répondre à la plupart des questions concernant le tracé des graphiques et les diverses améliorations qu'il est possible de leur apporter. Et si ce n'est pas le cas, utiliser un navigateur internet, la communauté Python aura sans doute déjà répondu à cette attente.

modules à importer Le package `matplotlib.pyplot` permet de tracer les courbes. Dans toute la suite, on considère qu'on a importé ce module ainsi :

```
import matplotlib.pyplot as plt
```

On importe également le module `numpy` pour avoir accès aux fonctions mathématiques sur les tableaux :

```
import numpy as np
```

I Tracé d'une courbe simple

C'est le niveau minimum attendu pour les concours, à l'écrit ou à l'oral.

Soit une fonction f définie ci-dessous :

```
def f(x):
    return( np.sin(x) / (1 + x ** 2) )
```

Pour tracer la courbe représentative de cette fonction, il faut créer une liste de points où l'on désire évaluer la fonction. Pour cela, il existe deux possibilités utilisant le module `numpy` :

1. L'instruction `linspace` permet de répartir la liste de manière linéaire entre les deux bornes avec le *nombre de points* souhaité. Dans l'exemple ci-dessous le nombre de points est égal à 200.

   ```
   x1 = np.linspace (0, 8 * np.pi, 200)
   ```

2. L'instruction `arange` permet de répartir la liste de manière linéaire entre les deux bornes avec un *pas* souhaité, ici de $8 \cdot \pi/200$.

   ```
   x2 = np.arange (0, 8 * np.pi, 8 * np.pi / 200)
   ```

Il ne reste plus qu'à tracer la courbe.

```
plt.plot(x1, f(x1))
plt.show()
```

Remarque 1. *Si la fonction f est définie par :* `f = lambda x : np.sin(x) / (1 + x ** 2)`*, le programme ne va pas se poursuivre car les fonctions* `lambda` *ne supportent pas les tableaux en argument.*

Remarque 2. *Toutes les courbes sont fournies à la fin du document.*

II Tracés de verticale et horizontale

Pour tracer une verticale pour un x donné, il faut créer une liste de points en x où x est constant et créer une liste en y avec ses bornes de variations. Il faut que les listes soient de *même taille*, soit par exemple 200 points. Dans l'exemple ci-dessous, x varie de [5,5] et y varie de [0,10].

```
xv = np.linspace (5, 5, 200)
yv = np.linspace (0, 10, 200)
```

Il ne reste plus qu'à tracer la courbe.

```
plt.plot(xv, yv)
plt.show()
```

Exercice 1.
Faire de même pour tracer une horizontale $y = 5$ pour les mêmes bornes de valeurs qui seront notées `(xh, yh)`.

III Tracé de courbes par superposition

Pour superposer des courbes, il faut mettre `plt.show()` à la fin de l'ensemble des tracés.

```
plt.plot(xv, yv)
plt.plot(xh, yh)
plt.show()
```

IV Tracé de plusieurs courbes sur la même figure

Pour cela, il faut utiliser la fonction `plt.subplot(n_lignes, n_colonnes, numero_figure)`. Le numéro de la figure est compté à partir de 1 (pour une fois) et augmente de gauche à droite et du haut vers le bas.

Voici un exemple de script :

```
plt.figure('nom_figure', figsize = (10, 5)) # Voir options de mise en forme

plt.subplot(2, 2, 1)
plt.plot(x1, f(x1))
plt.subplot(2, 2, 2)
plt.plot(xv, yv)
plt.subplot(2, 2, 3)
plt.plot(x2, f(x2))
plt.subplot(2, 2, 4)
plt.plot(xh, yh)

plt.show()
```

V Des options de mise en forme des graphiques

On reprend l'exemple du premier tracé. Mais à présent, on gère la taille de l'image, on ajoute un titre, une légende pour les deux axes, une grille et on définit la limite du rendu en x.

```
# On gère le nom et la taille de la figure (x, y) en inch (1 inch vaut 2.54 cm)
plt.figure('Tracé amelioré', figsize=(8, 6))
# Ajout d'un titre
plt.title(r"Courbe représentative de $f : x \mapsto \frac{\sin(x)}{1 + x^2}$", \
          verticalalignment = 'bottom')
# Légende pour l'axe des abscisses
plt.xlabel(r"$x$")
# étiquettes pour les abscisses
x_valeurs = [np.pi * k for k in range(9)]
x_etiquettes = [r'$0$', r'$\pi$', r'$2\cdot\pi$', r'$3\cdot\pi$', \
```

```
                r'$4\cdot\pi$', r'$5\cdot\pi$', r'$6\cdot\pi$', \
                r'$7\cdot\pi$', r'$8\cdot\pi$']
plt.xticks(x_valeurs, x_etiquettes)
# Légende pour l'axe des ordonnées
plt.ylabel(r"$y = f(x)$", rotation = 'vertical')
# il est possible aussi bien sûr d'indiquer des étiquettes pour les ordonnées
# Ajout d'une grille
plt.grid(True)
# Limite le rendu à la plage [0, 8 pi] en abscisse
plt.xlim([0, 8 * np.pi])
# On change la couleur, l'épaisseur, le style et l'étiquette
plt.plot(x1, f(x1), color = "red", linewidth = 2.5, linestyle ="-", label = r"$f(x)$")
# On localise la légende du graphique
plt.legend(loc = 'upper right')
plt.savefig('figure_6.png')
plt.show()
```

La figure est alors beaucoup plus lisible. Le titre, les légendes des axes, le quadrillage augmente fortement la plus-value.

Remarque 3. *Pour modifier la taille de la police d'affichage et la police elle-même, il faut utiliser la fonction à option* `rcParams[]` *comme indiqué ci-dessous.*

```
# Choix des paramètres d'affichage du texte dans les graphiques.
plt.rcParams['font.size'] = 8
plt.rcParams['font.sans-serif'] = 'Arial'
```

Les options pour le dessin des courbes sont les suivantes :
- `color = 'color'`, color = couleur du trait. Les couleurs standards sont : white, black, red, green, blue, cyan, magenta, yellow.
- L'épaisseur du trait est défini avec `linewidth = z`, z = épaisseur du trait en pt.
- Le style du trait est défini avec `linestyle = 'ls'`.
  Voici les différents styles :

: → ligne en pointillés	- → ligne pleine (défaut)	* → pas de courbe
– → ligne en tirets	-. → alternance tiret-point	

- Le nom de la figure est défini avec `label = 'nomdelafigure'`.
- La localisation de l'étiquette est définie avec `plt.legend(loc = 'localisation')`.
  Voici les différentes localisations :

`upperleft` ou 2	`uppercenter` ou 9	`upperright` ou 1
`centerleft` ou 6	`center` ou 10	`centerright` ou 5 ou 7
`lowerleft` ou 3	`lowercenter` ou 8	`lowerright` ou 7

On peut aussi marquer certains points de la courbe associée. Il suffit de changer la valeur de `linestyle` et de choisir une liste de points plus petite. On dispose aussi des options `markersize`, `markerfacecolor`, `markeredgecolor` et `markeredgewidth` pour modifier la taille, la couleur, la couleur du bord et la largeur du bord des marqueurs.
Dans l'exemple ci-dessous, la courbe possède initialement 300 points. On choisit de mettre en évidence uniquement 15 points uniformément répartis.

```
# Choix de l'ensemble des points où on évalue la fonction
x2 = np.linspace (0, 8 * np.pi, 15)
plt.plot (x2, f(x2), linestyle = '*', marker = 'o', markersize = 5,
markerfacecolor = 'cyan', markeredgecolor = 'blue', markeredgewidth = 1)
...
plt.show()
```

Remarque 4. *La taille de la police a été changée entre les deux dernières courbes.*

Les options pour le dessin des marqueurs sont les suivantes :
- Le type du marqueur est défini avec `marker = 'ma'`, `ma` étant le type de marqueur.

Voici les différents types de marqueurs :

o → boulette	+ → plus	. → point	s → carré
x → croix	* → étoile	∧ → triangle	

- La taille x du marqueur avec `markersize = x`.
- La couleur `color` du marqueur avec `markerfacecolor = 'color'`.
- La couleur `color` du bord du marqueur avec `markeredgecolor = 'color'`.
- La largeur y du bord du marqueur avec `markeredgecolor = y`.

On utilise alors le principe de superposition précédemment défini.

Remarque 5. *Il est possible d'enregistrer les courbes obtenues en ajoutant* `plt.savefig('nom_fichier.png')` *avant la ligne* `plt.show()`*. La courbe est alors enregistrée dans le répertoire de travail au format* `png`*.*

VI Tracés en échelle logarithmique

Exemple de tracé en échelle logarithmique décimal sur l'axe des abscisses.

```
x3 = np.linspace(1, 100)
def g(x) :
    return(x)

plt.figure('Echelle logarithmique', figsize = (9, 5))
# En échelle log décimal pour les abscisses
plt.subplot(1, 2, 1)
plt.grid(which = 'both')
plt.semilogx(x3,g(x3))
# En échelle linéaire pour les abscisses
plt.subplot(1, 2, 2)
plt.grid()
plt.plot(x3, g(x3))

plt.savefig('figure_8.png')
plt.show()
```

VII Tracés de courbes de niveau

Ces courbes de niveau sont particulièrement utiles au moment de tracer une courbe définie par une fonction implicite du type $f(x, y) = 0$, c'est à dire lorsque les deux variables ne sont pas séparables analytiquement. Une courbe de niveau représente l'intersection d'une surface d'équation $f(x, y)$ avec un plan de niveau 0.

Elles reposent sur la fonction `meshgrid` du module `numpy` et sur `contour` de `matplotlib.pyplot`.

Voici tout d'abord une présentation de la fonction `meshgrid` qui effectue un maillage d'une surface définie par ses coordonnées cartésiennes au croisement d'une liste d'abscisses et d'une liste d'ordonnées.

```
xlim = np.pi
ylim = np.pi
h = np.pi /200
# Discretisation des abscisses et ordonnees
X = np.arange(-xlim, xlim, h)
Y = np.arange(-ylim, ylim, h)

XX, YY = np.meshgrid(X, Y)

n = len(X)
Z = np.zeros((n, n))

for i in range(n):
    for j in range(n):
        Z[i, j] = np.sin(X[i]) * np.cos(Y[j])

plt.figure('Courbes de niveaux', figsize = (11, 5))
```

```
plt.subplot(1, 3, 1)
fig = plt.contour(X, Y, Z)

plt.subplot(1, 3, 2)
fig = plt.contour(X, Y, Z)
plt.clabel(fig, inline = True, fontsize = 10)

plt.subplot(1, 3, 3)
fig = plt.contour(X, Y, Z)
plt.clabel(fig, inline = False, fontsize = 10, color = 'green')

plt.savefig('figure_9.png')
plt.show()
```

Ici encore, plusieurs options d'agrément sont disponibles. Mais internet et la communauté `Python` est là pour vous guider si besoin !

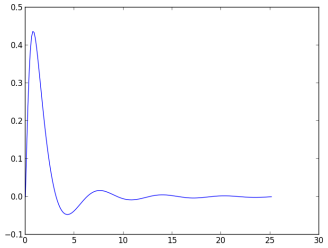

FIGURE 1 – Tracé d'une courbe simple

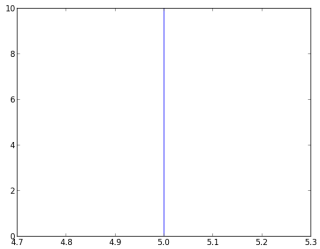

FIGURE 2 – Tracé d'une verticale

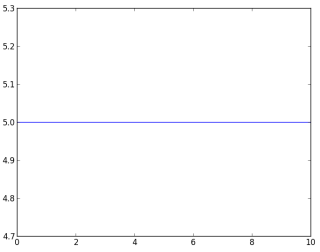

FIGURE 3 – Tracé d'une horizontale

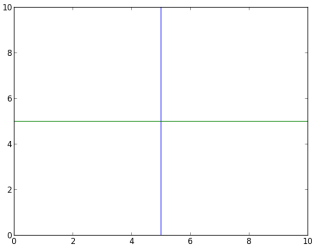

FIGURE 4 – Tracé de deux courbes superposées

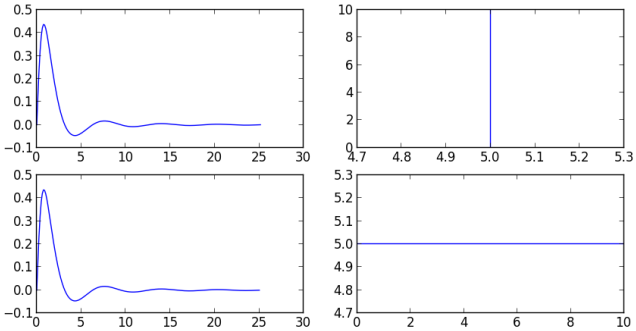

FIGURE 5 – Tracé de courbes multiples

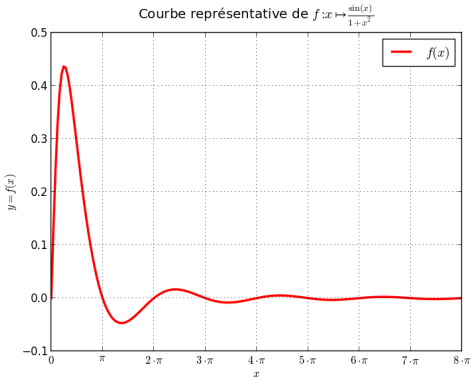

FIGURE 6 – Tracé de courbe agrémenté

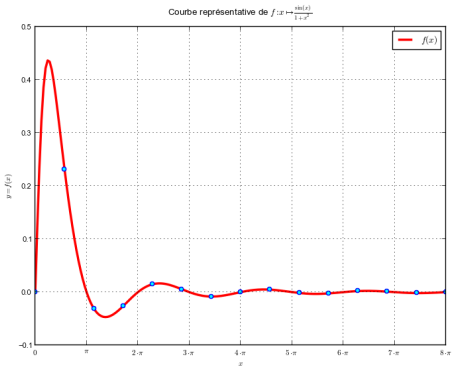

FIGURE 7 – Tracé de courbe avec marqueurs

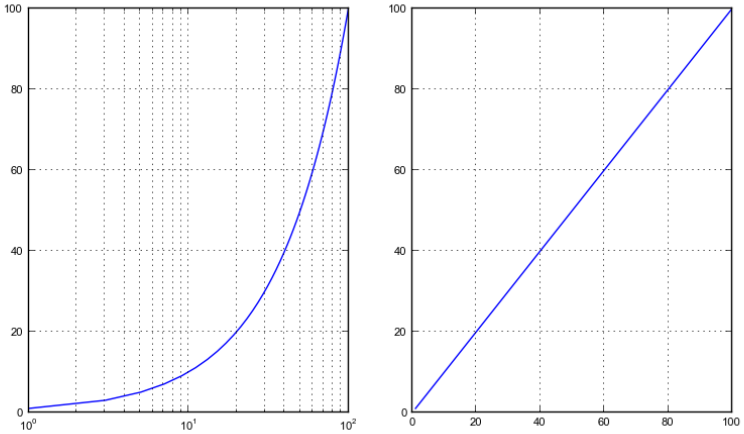

FIGURE 8 – Tracé de courbe en échelle logarithmique

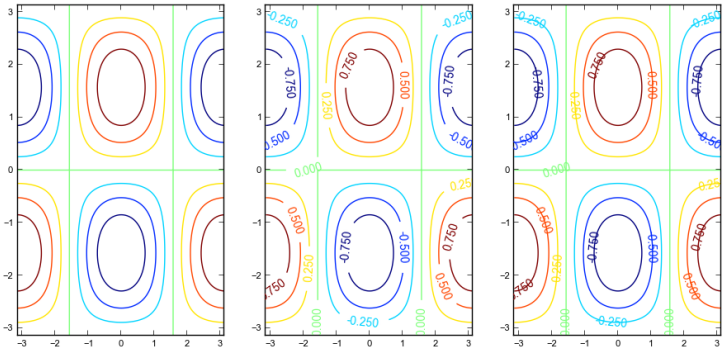

FIGURE 9 – Tracé de courbes de niveaux