Informatique - TP 4 **Fonctions et graphiques** 7 - 11 octobre 2024

Code de partage avec Capytale : d3e7-2046463

Corrigé

Exercices

Exercice 1 - échauffement - reprise en main de la représentation graphique

1. Tester le programmme suivant et faire varier les paramètres pour comprendre leur rôle.

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

x = np.linspace(-1,4,100)
y=x**3-5*x**2+2*x+8
plt.plot(x,y,'b')
plt.show()
```

En faisant varier le dernier paramètre de `linspace` (par exemple en remplaçant 1 000 par 5), on comprend que cela fait varier le nombre de points (ici le nombre d'abscisses définies). -1 et 4 représentent les bornes de l'intervalle sur lequel on représente (à travers la définition des abscisses ici encore).

2. Remplacer `np.linspace(-1,4,100)` par `np.arange(-1,4,0.1)`, quelle est la différence ?

Aucune en apparence, la différence est qu'avec `linspace`, on définit un nombre de points entre les deux bornes (ici 100), alors qu'avec `arange` on définit l'espacement entre deux points (ici $0,1$), ce que l'on appelle **le pas**.

3. Représenter la tangente au point d'abscisse 0.

Il suffit de représenter une droite en plus sur le graphique. Commençons par déterminer son équation. La tangente a pour équation : $y = f'(0)(x-0) + f(0)$
or $f(0) = 8$ et $\forall x \in \mathbb{R}, f'(x) = 3x^2 - 10x + 2$, donc $f'(0) = 2$
donc la tangente a pour équation : $y = 2x + 8$
Ensuite deux points suffisent pour représenter la tangente : on peut le faire avec $(-1, 6)$ et $(2, 12)$ et il suffit d'ajouter au programme ci-dessus.

```
x1=[-1,2]
y1=[6,12]
plt.plot(x1,y1,'r')
```

Exercice 2 - exponentielle et logarithme

Sur un même graphique, et sur un intervalle de votre choix, représenter les fonctions exponentielle, logarithme et identité.
Que retrouve-t-on ?

On représente sur l'intervalle $[-5, 5]$. Il faut définir trois listes d'ordonnées pour les trois fonctions (la fonction identité est définie par $f(x) = x$ pour tout réel. On essaie avec le programme suivant :

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

x = np.linspace(-5,5,100)
```

```
y1=np.exp(x) #première liste d'ordonnées avec la fonction exponentielle
y2=np.log(x) #deuxième liste d'ordonnées avec la fonction logarithme

plt.plot(x,y1,'b') # représente la fonction exponentielle
plt.plot(x,y2,'r') # représente la fonction logarithme
plt.plot(x,x,'g') # représente la fonction identité
plt.show()
```

Le problème est que les valeurs de la fonction exponentielle « écrasent » graphiquement les valeurs des autres fonctions qui deviennent peu visibles. Accessoirement, le logarithme, n'est pas défini pour $x \leqslant 0$. On va donc jouer sur les abscisses pour rendre l'échelle identique en x et en y. On fait donc en sorte que l'exponentielle s'arrête à l'ordonnée 5, ce qui correspond à l'abscisse vérifiant $e^x = 5$ et donc $x = \ln(5)$
De même pour le logarithme, on commence pour x tel que $\ln(x) = -5$, i.e. $x = e^{-5}$
Il faut ensuite modifier en conséquence les ordonnées et les `plot`.

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

x = np.linspace(-5,5,100)
x1 = np.linspace(-5,np.log(5),100)
x2 = np.linspace(np.exp(-5),5,100)
y1=np.exp(x1) #première liste d'ordonnées avec la fonction exponentielle
y2=np.log(x2) #deuxième liste d'ordonnées avec la fonction logarithme

plt.plot(x1,y1,'b') # représente la fonction exponentielle
plt.plot(x2,y2,'r') # représente la fonction logarithme
plt.plot(x,x,'g') # représente la fonction identité
plt.show()
```

Exercice 3 - représentation graphique et limite

En représentant graphiquement la fonction $x \mapsto \sqrt{x+1} - \sqrt{x}$, faire une conjecture sur sa limite.
On adapte à la fonction demandée qui est définie sur $\mathbb{R}_+$, on peut donc effectuer une représentation sur $[0, 100]$ comme ci-dessous et on pressent que cette fonction tend vers 0 quand x tend vers l'infini (ce qui est le cas).

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

x = np.linspace(0,100,1000)
y=np.sqrt(x+1)-np.sqrt(x)
plt.plot(x,y)
plt.show()
```

Exercice 4 - une limite de notre méthode avec la fonction valeur absolue

Définir la fonction valeur absolue, puis la représenter sur un intervalle de votre choix, mais symétrique par rapport à 0
Un message d'erreur s'affiche, pourquoi ? et comment contourner le problème ?
En définissant la fonction valeur absolue, conformément à sa définition mathématique puis en reproduisant la méthode habituelle ci-dessous, on obtient un message d'erreur.

```
def vabs(x):
    if x>0 :
```

```
        return x
    else :
        return -x

x = np.linspace(-10,10,100)
y=vabs(x)
plt.plot(x,y)
plt.show()
```

En fait, c'est la commande `y=vabs(x)` qui pose problème. Python n'arrive pas à calculer plusieurs images avec une fonction qui contient plus d'une formule (à cause du `if` dans la définition de notre fonction `vabs`). Il faut donc « bricoler » et on peut le faire de la manière suivante en définissant deux fonctions :

```
def vabs1(x):
  return x

def vabs2(x):
  return -x

x_1 = np.linspace(0,10,100)
y_1=vabs1(x_1)
plt.plot(x_1,y_1)
x_2 = np.linspace(-10,0,100)
y_2=vabs2(x_2)
plt.plot(x_2,y_2)
plt.show()
```

Exercice 5 - les fonctions usuelles

Représenter les fonctions valeur absolue, partie entière, $x \mapsto x^{\frac{1}{3}}$, cosinus, sinus...

On peut utiliser les fonctions prédéfinies dans `numpy`, en l'occurence : `np.abs`, `np.floor`, `np.cos` et `np.sin` en adaptant les intervalles de représentation aux domaines de définition respectifs.

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

# valeur absolue est définie sur R
x = np.linspace(-100,100,1000)
y=np.abs(x)
plt.plot(x,y)
plt.show()

# partie entière est définie sur R (mais l'escalier n'est pas visible si on prend
    un intervalle trop grand)
x = np.linspace(-10,10,1000)
y=np.floor(x)
plt.plot(x,y)
plt.show()

# les fonctions puissances quelconques sont définies sur R+*
x = np.linspace(0.01,100,1000)
y=x**(1/3)
plt.plot(x,y)
plt.show()

# cosinus est définie sur R
x = np.linspace(-50,50,1000)
y=np.cos(x)
```

```python
plt.plot(x,y)
plt.show()

# sinus est définie sur R
x = np.linspace(-50,50,1000)
y=np.sin(x)
plt.plot(x,y)
plt.show()
```