L3 E - S5 2018-19 --- TD ordinateurs --- séance 3

1 Un peu d'analyse et de représentations graphiques

L'instruction `pylab inline` permettra l'insertion des graphiques dans la feuille notebook (au lieu de les avoir dans des fenêtres "surgissantes").

```
In [ ]: pylab inline
```

Et on fait appel à la bibliothèque `matplotlib` pour utiliser les fonctionnalités graphiques de python :

```
In [ ]: from matplotlib.pylab import *
```

Exemple de représentation graphique :

```
In [ ]: def f(t):
            return t**2

        t = linspace(-4, 5, 30)    # on choisit le domaine de définition de la fonction
                                   # et le pas : 30 points de -4 à 5
        y=f(t)

        plot(t, y)
        show()
```

Est-ce clair ? On a tracé la fonction $y = f(t) = t^2$ sur l'intervalle $[-5, 5]$.

```
In [ ]: def f(t):
            return sqrt(t**2+2)


        t = linspace(-4, 5, 30)

        y=f(t)

        plot(t, y)
        savefig("mon_test.pdf")
        show()
```

On peut vouloir conserver ce joli dessin dans un fichier pdf ; il sufit de le demander avec `savefig` ; pour cela, il faut lui dire dans quel répertoire sauvegarder le fichier et c'est l'objet des instructions suivantes :

```
In [ ]: import os
        os.chdir('......')
```

Evidemment, chacun doit ajouter le bon chemin vers le répertoire choisi. La commande suivante permet de savoir dans quel répertoire la page notebbok actuelle est ouverte :

```
In [ ]: % pwd
```

A présent, on pourra vérifier que la commande `savefig(mondessin.pdf)` a gardé une copie du fichier `monfichier.pdf` dans le répertoire qui a été indiqué :

```
In [ ]: def f(t):
            return t**2

        t = linspace(-4, 5, 30)

        y=f(t)

        plot(t, y)
        savefig("monfichier.pdf")
        show()
```

On peut également "décorer" le dessin de commentaires, labels, légende, titre...

```
In [ ]: def f(t):
            return sqrt(t**2+2)

        t = linspace(-4, 5, 31)    # 31 points between 0 and 3

        y=f(t)

        plot(t,y)

        xlabel("t")
        ylabel("y")
        legend(["t^2*exp(-t^2)"])
        axis([-2, 3, -0.05, 6])   # [tmin, tmax, ymin, ymax]
        title("Le titre de mon dessin")
        savefig("x-puissance-2.pdf")

        show()
```

Il est très souvent utile de pouvoir visualiser plusieurs tracés sur une même figure :

```
In [ ]: def g1(t):
            return t**3-2*t**2-t+2
        def g2(t):
            return 3*t**2-4*t-1
        t = linspace(-1.5, 2.6, 60)
        y1 = g1(t)
        y2 = g2(t)
        plot(t, y1, "g-")
        plot(t, y2, "yo")
        xlabel("abscisse t")
        ylabel("ordonnee y")
```

```
axis([-2, 3, -3, 6])
legend(["t**3-2*t**2-t-2", "3*t**2-4*t-1"])
title("f et sa derivee")
savefig("f-et-sa-derivee.pdf")

show()
```

La commande `figure()` avec `subplot` permet de séparer plusieurs tracés sur différents dessins :

```
In [ ]: figure()          # pour faire apparaître des graphiques séparés
        subplot(2, 1, 1)  ### la 1ère figure

        def g1(t):
            return t**3-2*t**2-t+2
        def g2(t):
            return 3*t**2-4*t-1
        t = linspace(-1.5, 2.6, 60)
        y1 = g1(t)
        y2 = g2(t)
        plot(t, y1, "g-")
        hold
        plot(t, y2, "yo")
        xlabel("abscisse t")
        ylabel("ordonnee y")
        legend(["t**3-2*t**2-t-2", "3*t**2-4*t-1"])
        title("f et sa derivee")
        savefig("f-et-sa-derivee_1er-exemple.pdf")

        subplot(2, 1, 2)  ### la 2ème figure

        def g1(t):
            return 2*sin(3*t/2)
        def g2(t):
            return 3*cos(3*t/2)
        t = linspace(-6, 6, 100)
        y1 = g1(t)
        y2 = g2(t)
        plot(t, y1, "g-")
        hold
        plot(t, y2, "yo")
        xlabel("abscisse t")
        ylabel("ordonnee y")
        legend(["2*sin(3*t/2)", "3*cos(3*t/2)"])
        title("f et sa derivee")
        savefig("f-et-sa-derivee_2d-exemple.pdf")
        show()
```

2 Exercices

Tous les graphiques réalisés sont à sauvegarder dans le répertoire personnel

3 Exercice 1

Tracer sur un même dessin et pour $t > 0$ les graphes des fonctions $x \mapsto x^a$ pour $a \in \{-1, 0, 1/2, 1, 2\}$: on distinguera les représentations graphiques en prenant des couleurs différentes et on fera un dessin avec une fenêtre avec $x \in]0, 3]$ et un autre dessin avec une fenêtre avec $x \in]0, 10]$.

4 Exercice 2

On se donne une application de classe $\mathcal{C}^1$ sur un intervalle $[a, b]$. Supposant qu'on ne connaît pas sa dérivée, on va la calculer en considérant, pour h petit, la fonction suivante, définie pour $x \in [a + h, b - h]$:

$$x \mapsto \frac{f(x+h) - f(x-h)}{2h}$$

1. Ecrire une fonction `ApproximationDerivee(f,h,t)` qui retourne $\frac{f(x+h) - f(x-h)}{2h}$.
2. Tester la qualité de cette approximation pour la fonctions $f(t) = t^3 - t$, puis pour $g(t) = \sin(t)$. Chaque fois, on tracera sur un même graphique les graphes de f, f' et de l'approximation de f'. De plus, on fera apparaître les 3 graphiques dans une même fenêtre.

5 Exercice 3

On se donne une fonction strictement convexe. On rappelle que cela signifie que sa courbe repr'esentative est toujours en-dessous de ses cordes :

1-- Ecrire une fonction `Pente(f,a,b)` qui retourne le taux d'accroissement de la fonction f sur le segment $[a, b]$, c'est-à-dire $\frac{f(b) - f(a)}{b - a}$.

2-- Si f est strictement convexe sur $[a, d]$, on sait qu'elle admet un unique minimum. Ecrire une fonction `ComparePente(f,a,b,c,d)` qui retourne, parmi les 4 intervalles $[a, b]$, $[a, c]$, $[b, d]$,$[c, d]$, l'intervalle dans lequel se trouve son minimum (voir dessins ci-dessous où on appelle x_0 le minimum).

3-- Ecrire une fonction `ApproximationMinimum(f,a,b,)`ϵ) qui, pour une fonction strictement convexe f sur un intervalle $[a, b]$, retourne une approximation de son minimum à ϵ près.

Tester cette fonction avec $f(t) = t^2$ puis avec $g(t) = (t-2)^2$ sur l'intervalle $[-4, 6]$ avec $\epsilon = 0.1$.