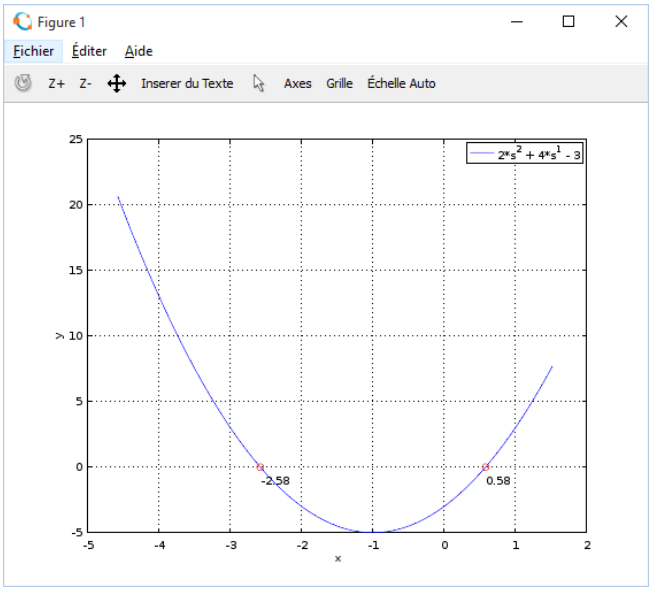
TP 3 : affichage de données – partie 2

Exercice 1 : Calcul et affichage de racines d'une fonction du second degré



Créer une fonction `affiche_racine(f)` qui permet d'afficher un polynôme du second degré f et ses racines si elles existent.

La fonction `f` sera codée sous forme d'un tableau de ses coefficients.

Par exemple, la fonction $2x^2+4x-3$ sera codée

```
[2 4 -3]
```

L'appel à la fonction depuis un script sera donc :

```
affiche_racine([a b c])
```

Les racines si elles existent seront affichées par un rond rouge avec leur abscisse (utiliser la fonction **`text`**)

Elément de Cours : Afficher des courbes en 3D

Affichage 3D

Enfin, Octave via GNUplot permet d'afficher des graphes en 3 dimensions. Là encore, plusieurs commandes sont disponibles.

- **`plot3`** : l'équivalent en 3D de la commande **`plot`** prend `(x,y,z)` en arguments. Avec plot3, on peut ajouter **`zlabel`** qui fonctionne comme **`xlabel`** et **`ylabel`**

Il est possible de créer une grille de points dans lequel une fonction `z=f(x,y)` sera affichée en utilisant les commandes **`surf`** (pour tracer une surface paramétrée d'équations) ou **`mesh`**.

Il est aussi possible d'interagir directement avec les données sur le graphe à l'aide de touches ou d'actions avec la souris.

Exercice 2 : courbe en 3D

Afficher dans l'intervalle `[0,30]` la fonction `f(x,y,z)` définie par :

```
x = t
y = sin(t)
z = cos(t)
```

Exercice 3 : surface en 3D

Définir la fonction `z = formeetrange(x,y)` qui calcule la fonction suivante

$$z = \cos(3.\sqrt[3]{x^2+y^2})$$

Définir une fonction `affichage(bornes)` où bornes contient un vecteur des bornes `min/max` pour l'axe `x` et `y` qui affiche le résultat de l'équation précédente sous forme de surface (fonction **`mesh`**)

Afficher la fonction sur l'intervalle `[0 1, 0 1], [0 2, 0 2] et [0 5, 0 5]`

Manipuler la forme générée avec les boutons de la souris.

Nota : Pensez à générer avant l'affichage **l'ensemble des points de maillage** en utilisant la fonction **`meshgrid`**

```
x = [0:0.1:1]
y = [0:0.1:1]
[X,Y] = meshgrid(x,y);
…
```

Exercice 4 : élections

Soit le résultat d'une élection mettant aux prises 4 candidats.

- Le candidat 1 (Louis) obtient : 231 voix
- Le candidat 2 (Jeanne) obtient : 424 voix
- Le candidat 3 (Maurice) obtient : 489 voix
- Le candidat 4 (Albertine) obtient : 12 voix

Le nombre d'inscrits à l'élection est de 1412.

Affichez les résultats des élections sous formes de barres (`bar` ou `barh`) et sous forme de camembert (`pie` ou `pie3`)

Exercice 5 : valeur en série

Nous voulons afficher les valeurs discrètes de la suite `de Fibonacci` dans l'intervalle `[1, 16]`

1. écrire la fonction `Fibonacci(n)` qui permet de renvoyer un vecteur des n valeurs successives de la suite de `Fibonnacci`
2. afficher les 16 premières valeurs de manière discrète (fonction **`stem`**)
3. créer le fichier **`fibo16.jpg`** correspondant au graphique du résultat

Nota : Suite de `Fibonacci`

$F_1 = F_2 = 1$

$F_{n+2} = F_{n+1} + F_n$

Elément de Cours : Sauvegarder les figures

Sauver et imprimer des figures

Pour imprimer des figures, c'est simple, il suffit de taper la commande **`print`** (sous unix).

Pour sauvegarder le graphique, la même commande print est utilisée. La commande générale est :

```
print('nom.extension','-dextension')
```

De nombreux formats sont disponibles : `gif`, `jpg`, `png`, `eps`, `svg`…. La figure est sauvée dans le répertoire courant.

```
print('graphe.jpg','-djpeg')
```

Exercice 6 : utiliser des données issues de capteurs

Nous souhaitons visualiser des données enregistrées par un accéléromètre 3 axes. Téléchargez le fichier `'accelero.csv'` à l'adresse suivante :
`https://github.com/truillet/upssitech/blob/master/GCGEO/1A/TP/accelero.csv`

Le fichier contient les données (*temps, X, Y, Z, thetaX,thetaY,thetaZ*)

1. Charger le fichier dans Octave en utilisant la commande **`load`** (*load("-ascii","accelero.csv")*, et définissez les vecteurs temps, `X,Y,Z` à partir du fichier chargé
2. Afficher les fenêtres suivantes : `X, Y` en fonction de t (en bleu), `X,Z` en fonction de `t` (en vert)
3. Sauver votre résultat dans un fichier image « `png` »
4. Ecrire un script octave (ou une fonction) qui permet d'effectuer ces opérations automatiquement

(**Optionnel**) : enregistrer vos propres données de capteurs et réeffectuez ces opérations.