Python-TP2

TP 2 – Informatique CM3 –

1 Python @ Polytech'Lille

Le texte de cette session de travaux pratiques est également disponible ici:
https://github.com/ecalzavarini/python-at-polytech-lille/blob/master/Python-TP2.ipynb

1.0.1 Modalités pour accomplir le TP et compte rendu

Nous vous rappelons que les modalités pour accomplir ce TP sont les mêmes de la première séance :

Vous aurez à écrire plusieurs scripts (script1.py , script2.py , ...). Les scripts doivent être accompagnés par un document descriptif unique (README.txt). Dans ce fichier, vous devrez décrire le mode de fonctionnement des scripts et, si besoin, mettre vos commentaires. Merci d'y écrire aussi vos noms et prenoms complets. Tous les fichiers doivent être mis dans un dossier appelé TP2-nom1-nom2 et ensuite être compressés dans un fichier archive TP2-nom1-nom2.tgz .

Enfin vous allez envoyer ce fichier par email à l'enseignant: soit Enrico (enrico.calzavarini@polytech-lille.fr) soit Stefano (stefano.berti@polytech-lille.fr).

Vous avez une semaine de temps pour compléter le TP, c'est-à-dire que la date limite pour envoyer vos travaux c'est dans 7 jours à partir d'aujourd'hui.

1.0.2 Gestion des accents en Python

Dans la première séance de TP, vous avez souvent rencontré un message d'erreur issu de l'utilisation des caractères accentués dans les scripts en Python. Voici une solution à ce problème. En première ligne d'un script, il faut insérer la ligne :

```
In [1]: # -*- coding: utf-8 -*-
```

ou bien

```
In [2]: # -*- coding: latin-1 -*-
```

Il s'agit d'un pseudo-commentaire indiquant à Python le système de codage utilisé, ici l'utf-8 (ou le latin-1) qui comprend les caractères spéciaux comme les caractères accentués, les apostrophes, les cédilles , etc .

1.0.3 Utiliser les fonctions en Python

Vous savez déjà ce qu'est une fonction mathématique d'une variable réelle. Considérons par exemple la fonction f(x) suivante :

f : x —> 2 x + 1

pour la définir en Python :

```
In [3]: #definition d'une fonction
        def f(x):
            return 2 * x + 1

        #utilisation de la fonction

        print(f(4))
```

```
9
```

Avez-vous remarqué qu'à la deuxième ligne, on n'a pas commencé à écrire au début de la ligne? De la même façon que pour les structures de contrôle 'if' ou 'for', nous devons appliquer la règle d'indentation. Cette indentation est indispensable pour que l'interpréteur Python comprenne la fin d'un bloc de définition d'une fonction.

Le principe de définition de fonctions est intéressant pour deux raisons :

1) cela nous permet de ne pas répéter un calcul long à taper,

2) Python possède un type spécial dédié au fonctions, que l'on peut donc manipuler, mettre dans des listes pour les étudier les unes à la suite des autres.

Par exemple :

```
In [4]: print( f(1) , f(2) , f(3) , f(4) , f(5) , f(6))
```

```
(3, 5, 7, 9, 11, 13)
```

```
In [5]: print( type(f))
```

```
<type 'function'>
```

```
In [6]: # definition d'une seconde fonction
        def hi(name):
            print("hello " + name + " from Python!!!")

        #exemple d'utilisation
        hi("Mark")
```

```
hello Mark from Python!!!
```

```
In [7]: #definition  d'une trosieme fonction
        from random import choice
        def lettre():
            return choice('abcdefghijklmnopqrstuvwxyz')

        #exemple d'utilisation
        lettre()
```

```
Out[7]: 'm'
```

```
In [8]: #definition d'une liste de fonctions
        mes_fonctions = [f,hi,lettre]

        #utilisation
        mes_fonctions[1]("Dominique")

        mes_fonctions[2]()
```

```
hello Dominique from Python!!!
```

```
Out[8]: 'g'
```

1.1 Script 1 : calcul des forces sur un ballon de football

Les forces sur un ballon de football en vol suite à un coup de pied d'un joueur sont deux: la force de la pesanteur (le poids F_P) et la force de traînée exercée par le frottement de l'air sur le ballon (F_T). Leurs expressions sont les suivantes :

$$F_P = M\ g$$

$$F_T = 0.5\ C_D\ \rho\ u^2\ S$$

où M est la masse du ballon, g l'accélération de gravité, C_D le coefficient de traînée, ρ la masse volumique de l'air, u la vitesse du ballon et enfin S la section du ballon $S = \pi\ R^2$ (avec R le rayon).

Nous demandons d'écrire un script qui tout d'abord demande à l'utilisateur d'entrer les valeurs du rayon du ballon (en mètres), de la masse du ballon (en kg) et de la vitesse du ballon ainsi que l'unité de mesure pour cette dernière (soit "m/s" ou "km/h"). Ensuite le script devra calculer les forces F_P et F_T, afficher les deux valeurs ainsi que leur rapport F_T/F_P.

On demande de faire tout cela à l'aide de quatre fonctions :

1) une fonction qui convertit la vitesse de "km/h" en "m/s" si besoin

2) une fonction qui calcule la surface S du ballon à partir du rayon R

3) une fonction qui calcule F_T

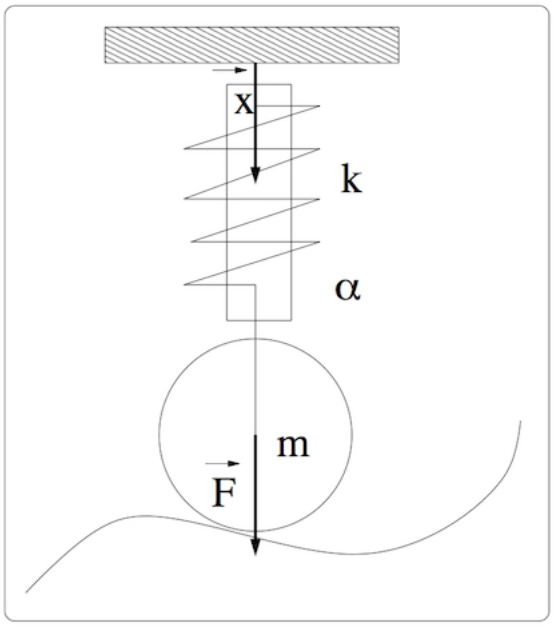
4) une fonction qui calcule F_P

Les données du problème sont l'accélération de gravité $g = 9.81\,m/s^2$, la masse volumique de l'air $\rho = 1.2\,kg/m^{-3}$, le coefficient de trainée $C_D = 0.2$. Tourner le script plusieurs fois et dire ce qui change dans le rapport F_T/F_P pour des valeurs de vitesse croissantes (p.ex. $u = 10\,km/h$, $u = 120\,km/h$) et à masse et rayon fixes.

1.2 Script 2 : étude de la fonction de transfert du système ressort - amortisseur (suspension) d'un automobile

L'objet de ce script est d'illustrer à l'aide de Python les rôles respectifs joués par le ressort et l'amortisseur d'un système automobile. Pour ce faire, nous étudions un ressort couplé à un amortisseur en parallèle.

Comme dans la figure ci-dessous:



L'équation vérifiée par le système est la suivante :

$$\ddot{x} + \frac{\omega_0}{Q}\dot{x} + \omega_0^2\ x = \frac{F}{m}\cos(\omega t)$$

Ici m est la masse de la roue, F et ω sont respectivement l'intensité de la force appliquée (qui modélise l'effet des ondulations du sol sur la roue) et sa pulsation , ω_0 la pulsation caractéristique du ressort ($\omega_0 = \sqrt{k/m}$) et enfin Q est un coefficient appelé coefficient de qualité. Ici on négligera le poids de la roue.

Un système très amorti a un Q faible. À l'inverse, un Q élevé correspond à un système peu amorti. Pour fixer les idées, le Q d'une voiture avec des amortisseurs en bon état est légèrement supérieur à 1.

De la solution de l'équation du système ressort-amortisseur on trouve la fonction de transfert qui décrit la réponse du système en fonction de la pulsation ω.

En particulier la fonction de transfert (le module de cette fonction pour la précision) s'écrit :

$$T(\omega) = \frac{F}{m\ \omega_0^2}\frac{1}{\sqrt{\left(1-\omega^2/\omega_0^2\right)^2 + Q^{-2}\ \omega^2/\omega_0^2}}$$

ou en utilisant la pulsation adimensionnée u = $\omega/\omega 0$:

$$T(u) = \frac{F}{m\ \omega_0^2}\frac{1}{\sqrt{\left(1-u^2\right)^2 + u^2/\ Q^2}}$$

Nous demandons d'écrire un script qui trace un graphique du module de la fonction de transfert $T(u)$ en fonction de la pulsation adimensionnée u (dans l'intervalle $[0, 2]$) pour toutes les valeurs de Q dans l'intervalle $[0, 10]$ avec rapport entre une valeur de Q et la valeur précédente de $\delta Q = 2.0$ (facteur de redoublement).

Prendre la valeur suivante pour l'amplitude de la fonction de transfert

$$\frac{F}{m\omega_0^2} = 2$$

Ça pourrait être utile :

La boucle while Le but de la boucle "while" est de répéter certaines instructions tant qu'une condition est respectée. On n'est pas donc obligé de savoir au départ le nombre de répétitions à faire.

```
In [11]: nb_repetitions = 3
         i = 1

         while i <= nb_repetitions :
             print "Et "+str(i)+"!"
             i = i+1
         print "Zéro!"
```

```
Et 1!
Et 2!
Et 3!
Zéro!
```

L'instruction "break" sert, non pas à interrompre le programme, mais à sortir de la boucle.

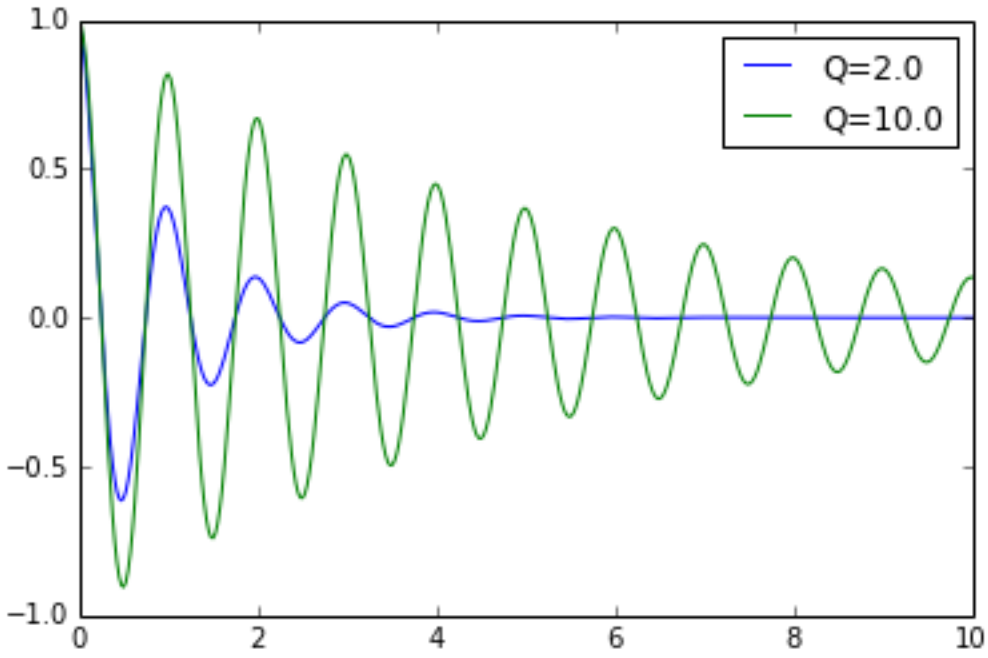
```
In [12]: nb_repetitions = 3
         i = 1

         while i <= nb_repetitions :
             print "Et "+str(i)+"!"
             i = i+1
             if i==3:
                 break
         print "Zéro!"
```

```
Et 1!
Et 2!
Zéro!
```

Création d'une légende sur un graphique

```
In [13]: % matplotlib inline

         from math import *
         import numpy as np
         import matplotlib.pyplot as plt

         # Création d'une array numpy : Temps = Abscisses
         t = np.linspace(0,10,400)

         # Fonction U dépendant d'un paramètre Q
         def U(t,Q):
             return np.exp(-2/Q*t)*np.cos(2*pi*t)

         # Plot1 avec Q = 2 et édition du label correspondant
```

```
Q=2.0
plt.plot(t,U(t,Q),label="Q=" + str(Q))

# Plot2 avec Q = 10 et édition du label correspondant
Q=10.0
plt.plot(t,U(t,Q),label="Q=" + str(Q))

# Appel de la légende
plt.legend(loc='upper right')

plt.show()
```



1.3 Script 3 : Calcul de la valeur critique du coefficient de qualité d'une suspension

Écrire un script qui calcule la valeur critique de Q, c'est-à-dire la valeur pour laquelle on observe une transition entre le régime de vibrations sur-amorties ($T(u)$ non croissante) et celui de résonance ($T(u)$ non monotone, avec un maximum pour $0 < u < 1$).

Pour cela faire, on peut par exemple calculer la valeur maximale T_{max} de $T(u)$ pour des valeurs croissantes de Q, et arrêter le calcul lorsque T_{max} est supérieur à $F/(m\omega_0^2)$.

Dans ce cas on vous suggère de prendre $Q = 0.5$ comme valeur de départ et de l'incrémenter pas à pas d'une valeur $\delta Q = 10^{-6}$.

Utiliser encore : $F/(m\omega_0^2) = 2$

1.3.1 Calcul de la valeur maximale avec numpy

Pour ce script, il peut être utile d'utiliser la fonction numpy pour calculer le maximum d'une liste :

```
In [14]:  import numpy as np
```

```
val=np.linspace(0,7,500)

val_max = np.amax(val)

print(val_max)
```

```
7.0
```

1.4 Script 4 : Préparation des données d'entrée pour une machine-outil à contrôle numérique

Les machines-outil d'usinage à contrôle numérique peuvent être contrôlées à travers un fichier de données d'entrée. Ce fichier contient toutes les spécifications géométriques de l'objet à usiner et il a souvent la forme d'une matrice en deux dimensions, dans laquelle chaque élément identifie une coordonnée spatiale (x, y) alors que la valeur de l'élément contient l'information sur l'usinage (par exemple dans une fraiseuse "0" signifie couper et "1" signifie garder tel qu'il est).

Nous considérons ici une machine-outil fraiseuse avec résolution du dixième de millimètre, et une surface de travail de $(10 \times 10)cm^2$. Nous voulons découper de la pièce brute une bielle à partir d'un bloc de métal de dimensions $(10 \times 10)cm^2$ (l'épaisseur du bloc n'a pas d' importance ici).

La géométrie de la bielle est montrée dans la figure ci-dessous :

Détails de la géométrie de la bielle:

- le grand anneau a rayon interne $r_1 = 10.0\ mm$, rayon externe $r_2 = 16.0\ mm$, position du centre $x_1 = 50.0\ mm$, $y_1 = 20.0\ mm$.
- le petit anneau a rayon interne $r_3 = 3.0\ mm$, rayon externe $r_4 = 11.2\ mm$, position du centre $x_2 = 50.0\ mm$, $y_2 = 75.0\ mm$.
- la barre de liaison a longueur $l = 30.0\ mm$, hauteur $h = 10.0\ mm$ et centre géométrique en $x_3 = 50.0\ mm$, $y_3 = 50.0\ mm$.

On demande d'écrire un script qui crée la matrice pour usiner la bielle en figure.

Cette matrice devra avoir 1000 x 1000 éléments, chaque élément correspondant à $(0.1 \times 0.1)mm^2$, et la valeur des éléments doit être soit 0 si le matériau doit être enlevé (ou bien complètement fraisé dans ce cas) soit 1 s'il doit être gardé comme il est.

a) Montrer à travers un graphique la matrice que vous avez obtenu

b) Tracer aussi un graphique de la section longitudinale centrale de la bielle.

Cela pourrait vous être utile :

Créer et manipuler un tableau multidimensionnel avec numpy

```
In [16]: import numpy as np

         #définition d'une matrice avec numpy
         tableau = np.array ([[0 ,1 ,2] ,[1,2,3], [4,6,12], [44 ,55 ,56]])

         print(tableau)
```

```
[[ 0  1  2]
 [ 1  2  3]
 [ 4  6 12]
 [44 55 56]]
```

```
In [17]: # copier un element du tableau, tableau[num. ligne][num. colonne]
         n=tableau[3,0]

         print(n)
```

```
44
```

Noter que l'indice de gauche indique toujours le nombre de la ligne tandis que l'indice de droite indique le numéro de la colonne.

Ce système d'indexage (dite convention "row-major order" ou "ligne par ligne") n'est pas toujours intuitif : dans le cas d'une matrice $A[i, j]$ qui contient des informations spatiales l'indice à gauche (i) identifie la coordonnée verticale y et l'indice de droite (j) identifie la coordonnée horizontale x.

```
In [18]: c=tableau[:,0] # copier une colonne du tableau

         print(c)
```

```
[ 0  1  4 44]
```

```
In [19]: d=tableau[:2,:2] # copier une colonne du tableau

         print(d)
```

```
[[0 1]
 [1 2]]
```

```
In [20]: tableau = tableau + 2  #ajouter 2 aux éléments du tableau
         print(tableau)
```

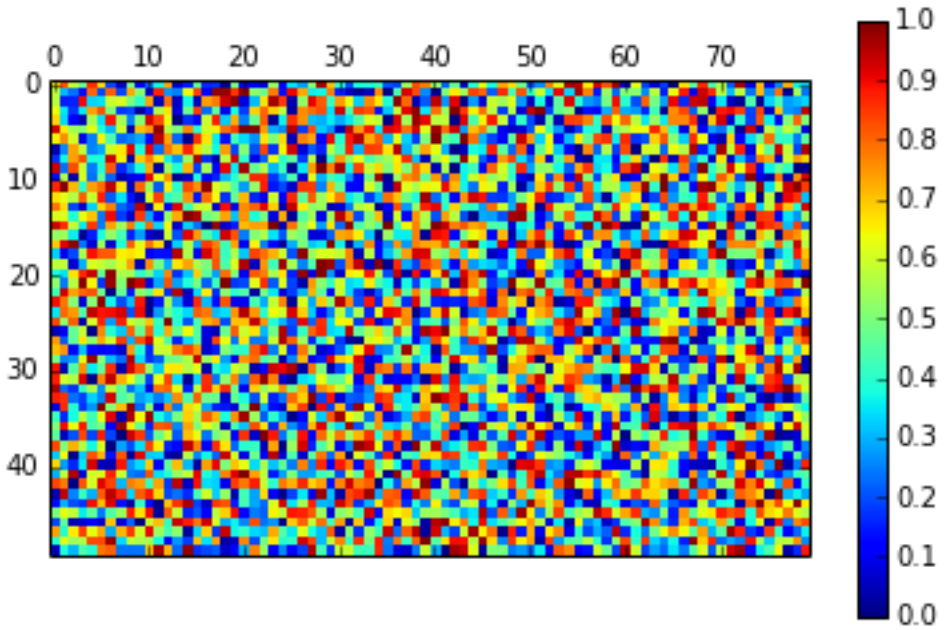
```
[[ 2  3  4]
 [ 3  4  5]
 [ 6  8 14]
 [46 57 58]]
```

```
In [21]: tableau2 = np.zeros((4,3))  #définition d'une matrice composée de zéros

         print(tableau2)
```

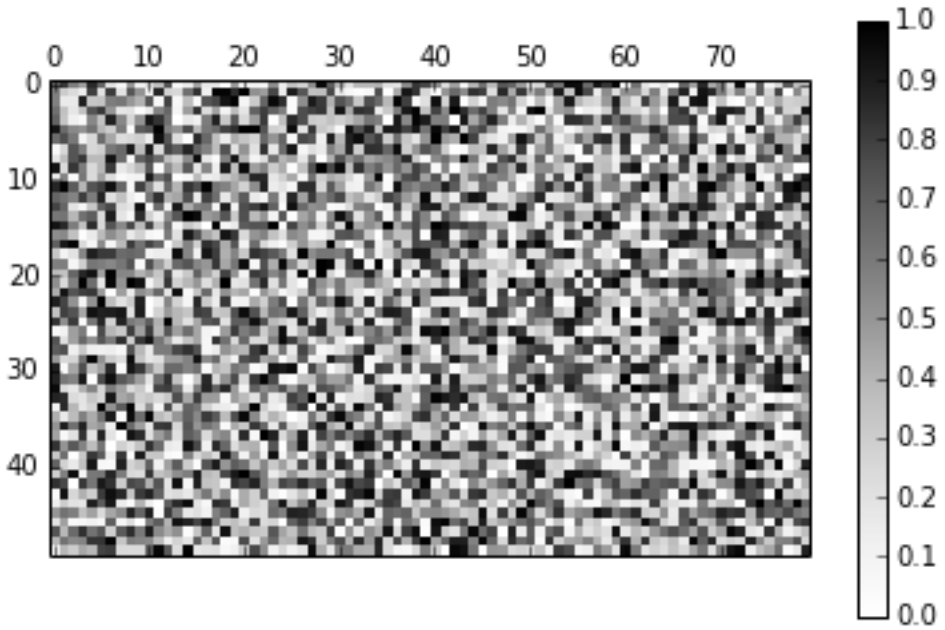
```
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
```

Représenter graphiquement une matrice

In [22]:
```
import numpy as np
import matplotlib.pyplot as plt

#création d'une matrice aléatoire de 80x50 elements
my_mat = np.random.random((50, 80))

# représentation graphique
plt.matshow(my_mat)

# définition des marqueurs en x
plt.xticks(range(0,80,10))
# définition des marqueurs en y
plt.yticks(range(0,50,10))

plt.colorbar()

plt.show()
```



In [23]:
```
# représentation graphique en blanc et noir
plt.matshow(my_mat,cmap='Greys')
plt.colorbar()
plt.show()
```


0
10
20
30
40
50
60
70
0
10
20
30
40
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0