

## DEVOIR SURVEILLÉ D'INFORMATIQUE 3

### CI 2 : ALGORITHMIQUE ET PROGRAMMATION

#### TRACÉ DE L'ABAQUE DU TEMPS DE RÉPONSE RÉDUIT

Nom : .....

### Question de cours

On donne le code de la fonction mystère.



```
def mystere(nb,tab):
    """
    Entrées :
        * nb, int — nombre entier
        * tab, list — liste de nombres entiers triés
    Sorties :
        *
    """
    g, d = 0, len(tab)-1
    while g <= d:
        m = (g + d) // 2
        if tab[m] == nb:
            return m
        if tab[m] < nb:
            g = m+1
        else:
            d = m-1
    return None
```

**Question 1** Quel est le nom de cet algorithme ? Que renvoie-t-il ?

**Question 2** Indiquer les valeurs successives prises par  $m$  dans le cas où  $nb=4$  et  $tab=[1,2,3,4,5,6,7,8,9]$ .

**Question 3** Quelle est la complexité de l'algorithme? Expliquer brièvement pourquoi? Quelle serait la complexité d'un algorithme naïf ayant le même objectif?

## Tracé de l'abaque de temps de réponse à 5 %

Objectifs

L'objectif de ces travaux est de construire le programme permettant de tracer l'abaque du temps de réponse réduit utilisé en asservissement pour connaître le temps de réponse à 5% des systèmes d'ordre 2.

### Mise en situation

L'équation différentielle d'un système du second ordre peut se mettre sous la forme :

$$s(t) + \frac{2\xi}{\omega_0} \cdot \frac{ds(t)}{dt} + \frac{1}{\omega_0^2} \cdot \frac{d^2s(t)}{dt^2} = K \cdot e(t)$$

en notant :

- $K$  : le gain statique ;
- $\xi$  : le coefficient d'amortissement ;
- $e(t)$  et  $s(t)$  : l'entrée et la sortie du système.

On suppose que toutes les conditions initiales sont nulles. Pour une entrée unitaire de type échelon unitaire  $e(t) = u(t)$ ,  $K = 1$  et  $t \geq 0$  on montre que :

- si  $\xi < 1$ , le régime est pseudo périodique et :

$$s(t) = 1 - \frac{e^{-\xi\omega_0 t}}{\sqrt{1-\xi^2}} \sin\left(\omega_0 t \sqrt{1-\xi^2} + \arcsin \sqrt{1-\xi^2}\right)$$

- si  $\xi = 1$ , le régime est critique et :

$$s(t) = 1 - (1 + \omega_0 t) e^{-\omega_0 t}$$

- si  $\xi > 1$ , le régime est apériodique et :


$$s(t) = 1 + \frac{e^{-\omega_0 t(\xi + \sqrt{\xi^2 - 1})}}{2(\xi\sqrt{\xi^2 - 1} + \xi^2 - 1)} - \frac{e^{-\omega_0 t(\xi - \sqrt{\xi^2 - 1})}}{2(\xi\sqrt{\xi^2 - 1} - \xi^2 + 1)}$$

**Dans l'ensemble de ce sujet, on considèrera que  $s$  est une fonction du temps réduit  $t \cdot \omega_0$ . On notera indifféremment le coefficient d'amortissement  $z$  ou  $\xi$ .**


## Tracé de la réponse indicielle

On dispose des fonctions Python `f_pseudo` et `f_aperiodique` permettant d'évaluer la fonction pour  $s$  pour un couple  $(t\omega_0, \xi)$ .

**Question 4** Donner, en Python, le contenu de la fonction `f_critique` permettant de définir la fonction  $(t\omega_0) \rightarrow s(t\omega_0)$  dans le cas où  $\xi = 1$ . On respectera impérativement la syntaxe Python. Les spécifications de la fonction seront les suivantes :

 `def f_critique(t, om0):` 1  
 `"""` 2  
*Fonction permettant de calculer  $s(t)$  dans le cas où  $z > 1$ .* 3  
*Entrées :* 4  
*\* t,flt : le temps en secondes* 5  
*\* om0,flt : la pulsation en rad.s<sup>-1</sup>* 6  
*Sortie :* 7  
*\* res,flt :  $s(t)$ . Ici, sans unité.* 8  
 `"""` 9

**Question 5** Donner, en Python, le contenu de la fonction `f_s` permettant de définir la fonction  $(t\omega_0, \xi) \rightarrow s(t\omega_0\xi)$  dans le cas où  $\xi \in \mathbb{R}_+^*$ . On donne ci-dessous les spécifications de la fonction.

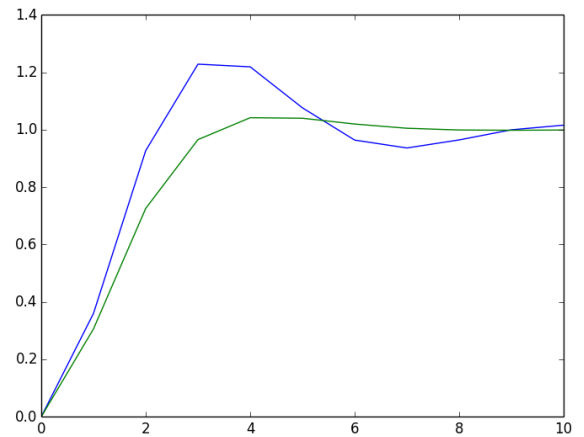
 `def f_s(tom0,z):` 1  
 `"""` 2  
*Fonction permettant de calculer la réponse indicielle d'un système du second ordre.* 3  
*Entrées :* 4  
*\* tom0,flt : temps de réponse réduit* 5  
*\* z,flt : coefficient d'amortissement* 6  
*Sortie :* 7  
*\* s(tom0,z),flt* 8  
 `"""` 9

La fonction **trace\_s** donnée ci-dessous permet de tracer  $s(t\omega_0, \xi)$  pour  $t\omega_0 \in [0, 10]$  par pas de 1 et pour une valeur de  $\xi$  déterminée. Les deux appels successifs de la fonction **trace\_s** permettent de réaliser le tracer les 2 courbes ci-dessous.

```

# Définition de la fonction trace
def trace_s(z):
    x = []
    y = []
    for i in range(11):
        t = i
        x.append(t)
        y.append(f_s(t,z))
    plot(x,y)
# Appels de la fonction trace
trace_s(0.4)
trace_s(0.7)

```



**Question 6** Expliquer l'objectif des lignes 2 à 9.

On observe que la courbe tracée n'est pas lissée. Pour avoir un meilleur rendu, il est nécessaire d'évaluer la fonction en davantage de points.

**Question 7** Modifier les lignes 5 et 6 pour que la courbe tracée soit réalisée en 1000 points sur un intervalle de  $t\omega_0$  variant de 0 à 10.

## Tracé de l'abaque

On note  $t_r$  le temps de réponse à 5%. L'abaque du temps de réponse permet de tracer le produit  $t_r\omega_0$  en fonction du coefficient d'amortissement  $\xi$ .

**Question 8** Dans les conditions de la fonction  $s$  définie dans la partie précédente, quelle est la valeur finale prise par  $s(t)$  ?

**Question 9** Écrire en Python la fonction **is\_in\_strip** ayant les spécifications suivantes :

python

```
def is_in_strip(x):
    """
    Fonction permettant de savoir si une valeur est dans la bande des + ou - 5% de la valeur finale .
    Entrée :
        x, flt : réel
    Sortie :
        True si la valeur est dans la bande à + ou - 5%
        False si la valeur n'est pas dans la bande à + ou - 5%
    """
```

On donne la fonction suivante permettant de connaître le temps de réponse réduit à partir duquel la réponse indicielle d'un système est dans la bande à plus ou moins 5% pour un coefficient d'amortissement particulier.

python

```
def calcul_tom0(z,tom0=500):
    """
    Recherche du temps de réponse à 5%
    Entrées :
        * z, flt : coefficient d'amortissement
        * tom0 (flt, optionnel) : si non précisé, on calcule le temps de réponse en partant de tom0 = 500
    Sortie :
        * tom0 (flt) : temps de réponse à 5%
    """
    pas_tom0=0.05
    x = f_s(tom0,z)
    if z<0.7:
        # Dans cas, s'assurer que le tom0 initial est suffisamment grand
        while is_in_strip(x) :
            tom0 = tom0 - pas_tom0
            x = f_s(tom0,z)
            tom0=tom0+pas_tom0
    else :
        # Dans cas, s'assurer que le tom0 initial est suffisamment petit
        while not is_in_strip(x) :
            tom0 = tom0 + pas_tom0
            x = f_s(tom0,z)
            tom0=tom0-pas_tom0
    return tom0
```

**Question 10** Expliquer le mode de recherche du temps de réponse à 5% dans le cas où  $z < 0,7$  puis dans le cas où  $z \geq 0,7$ . Pourquoi distingue-t-on ces 2 cas ? On pourra s'aider de l'abaque donné ci-dessous.

Remarque

On pourra remarquer que dans un cas, la recherche se fait « en avançant » et dans le second cas « en reculant ».

L'algorithme suivant permet de créer les listes xx et yy permettant de tracer l'abaque du temps de réponse réduit en fonction du coefficient d'amortissement.

python

```
xx,yy = [],[]
n = 1000
z = 0.01
tom0 = 500
pasz = 0.01
while z<=100:
    print(z)
    if z<0.7:
        tom0=calcul_tom0(z,tom0)
    else :
        tom0=calcul_tom0(z,0)

    if z<0.1:
        pasz = 0.001
    elif z<1:
        pasz = 0.01
    elif z<10:
        pasz = 0.1
    else :
        paz = 1

    xx.append(z)
    yy.append(tom0)
    z=z+pasz
```

### Question 11

1. Donner l'intervalle de variation de  $z$  pour le tracé demandé.
2. Donner le pas de  $z$  sur chacun des intervalles.
3. Pourquoi ne pas conserver le même pas sur chacun de ces intervalles ?
4. En vous aidant du tracé de l'abaque, expliquer pourquoi  $\text{tom0}$  est calculé différemment suivant la valeur de  $z$  ? Expliquer le choix des arguments de la fonction `calcul_tom0` dans chacun des cas.

**Question 12** On souhaite stocker les données des listes `xx` et `yy` dans un fichier texte encodé en ASCII. Chacun des nombres doit être stocké sur 17 caractères. Indiquer la taille du fichier ainsi créé.



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

plt.plot(xx,yy, label="Temps de réponse réduit $r \backslash \omega_0$")
plt.xlabel("Coefficient d'amortissement $\xi$")
plt.ylabel("Temps de réponse réduit $r \backslash \omega_0$")
plt.title("Temps de réponse réduit d'un système du 2nd ordre")
plt.legend()
plt.loglog()
plt.grid(which="major",axis="x",linewidth=1.5, linestyle='--')
plt.grid(which="major",axis="y",linewidth=1.5, linestyle='--')
plt.grid(which="minor",axis="x",linewidth=0.75, linestyle='--', color='0.75')
plt.grid(which="minor",axis="y",linewidth=0.75, linestyle='--', color='0.75')
```

