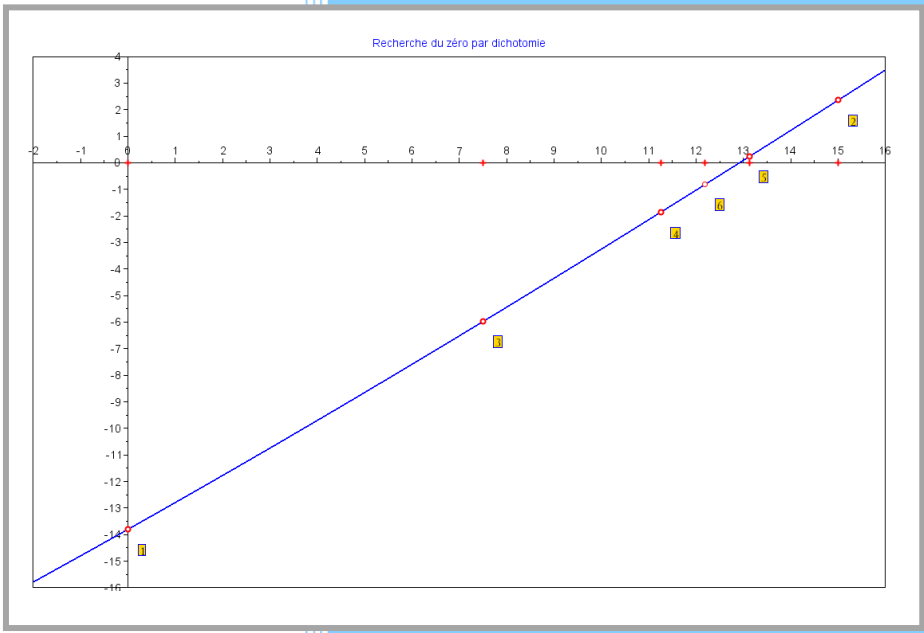
LYCEE DEODAT DE SEVERAC
INFORMATIQUE EN CPGE PTSI

TP7 : METHODES NUMERIQUES POUR LA RESOLUTION DES PROBLEME STATIONNAIRE - DICHOTOMIE



Patrice MICHEL
patrice.michel@tesa.prd.fr

Alexis REDONDO
a.redondo@free.fr

python™

OBJECTIFS DU TP

Les objectifs de ce TP sont :

- *Evaluer*

CONSIGNES :

1. ACTIVITE 1 : DETERMINATION DE L'ANGLE D'ORIENTATION D'UNE TUYERE DE REACTEUR

L'exemple introduit dans le cours, conduit à chercher la solution de l'équation suivante :

$$37,5.\cos\alpha - 57,5.\sin\alpha = 23,7$$

La détermination de l'angle reviendra donc à résoudre une équation trigonométrique de la forme $A.\cos\alpha + B.\sin\alpha = C$, soit à chercher les zéros de la fonction :

$$f(\alpha) = C - (A.\cos\alpha + B.\sin\alpha).$$

On donne l'expression de la solution analytique qui permettra d'évaluer l'erreur :

$$\alpha_0 = \arccos\frac{C}{D} - \arccos\frac{A}{D}$$

où $\begin{vmatrix} A = 37,5 \\ B = 57,5 \\ C = 23,7 \\ D = \sqrt{A^2 + B^2} \end{vmatrix}$,

APPLICATION

Pour le problème considéré :

Q1. *Avec Scilab, après avoir défini la fonction f(α), réaliser son tracé pour α ϵ [-45,45].*

Q2. *Réaliser le code Scilab permettant la recherche de la solution. Pour cela :*

- *on se basera sur l'algorithme itératif présenté en cours,*
- *on structurera le code de la manière suivante :*
  - *on construira une fonction **dichotomie_iteratif** qui sera codée dans un module (fichier) indépendant à l'extension « .sci » et qui pourra être appelée depuis le programme principal,*
  - *dans le programme principal **recherche_zero.sce** on définira les différents paramètres, la fonction f(α), et on appellera la procédure **dichotomie_iteratif***
  - *au préalable, dans le programme principal il est nécessaire d'exécuter le module dichotomie_iteratif.sci, grâce à la commande :*

```
exec ('dichotomie_iteratif.sci',-1)
```

  - *les fonctions de tracé pourront éventuellement être définies dans un module indépendant*

Q3. *Exécuter le code et caractériser la **précision** de la méthode : on poussera la tolérance jusqu'à la précision machine.*

Q4. *Analyser la convergence de la méthode : pour cela, calculer à chaque itération l'erreur à la solution analytique, et tracer son évolution à l'issue de la résolution.*

On se propose maintenant de chercher le zéro en utilisant directement la fonction `fsolve` implémentée dans Scilab.

On utilise la fonction `fsolve` sous la forme :

```
fsolve(x0,f)
```

où f est la fonction dont on veut déterminer une racine et x0 est une valeur de départ pour l'algorithme de recherche.

La fonction `fsolve` retourne 3 valeurs :

- la valeur approchée de la racine,
- la valeur de la fonction en ce point (qui doit être « assez petite » si tout s'est bien passé)
- une troisième valeur qui indique comment l'algorithme s'est déroulé (vaut 1 si tout s'est bien passé).

On donne ci-dessous un exemple illustrant l'utilisation de cette fonction pour une équation scalaire :

```
function y=fct(x)
   y=2*x^3-30*x^2-3*x+200
endfunction

x=[-3:0.1:15];plot2d(x,fct(x));

sol,eps,info]=fsolve(0,fct)
disp([sol,eps,info])
```

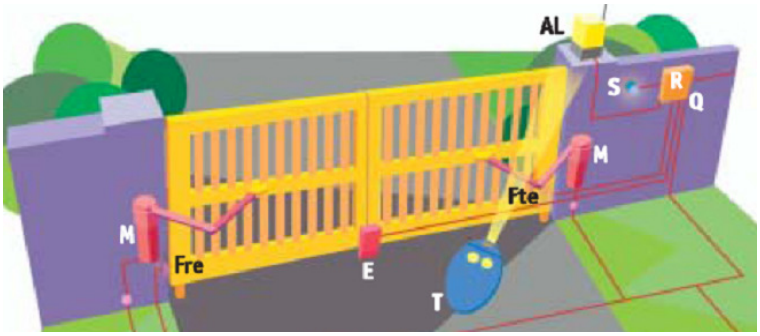
APPLICATION

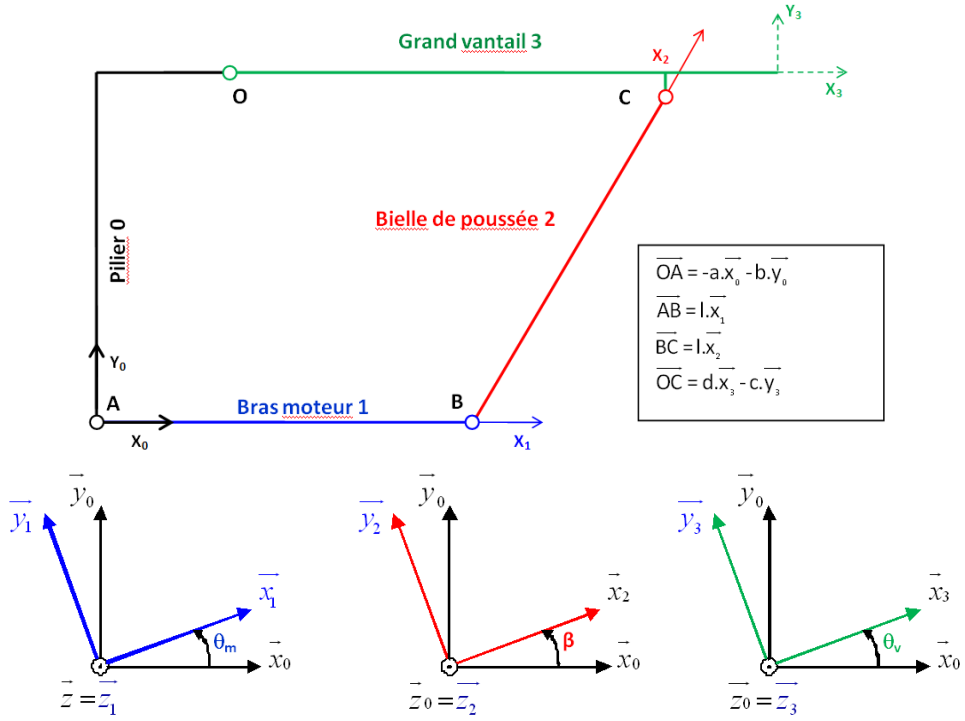
Q5. *Chercher la solution en utilisant la fonction* `fsolve` *de Scilab*

Q6. *Utiliser la fonction* `timer` *pour comparer les temps de calcul des deux méthodes*

2. ACTIVITE 2 : RESOLUTION DE LA RELATION E/S DU PORTAIL

En TP de SII, vous avez travaillé sur la relation entrée-sortie géométrique du portail. On se propose ici de résoudre numériquement l'équation obtenue lors de l'écriture de la fermeture géométrique.





Avec a=100, b=260, c=20, d=324.22, l=280

La fermeture géométrique conduit à la relation entrée/sortie suivante, avec θ_m angle moteur et θ_v angle du vantail :

$$\left[bc - ad + dl.\cos(\theta_m) - cl.\sin(\theta_m)\right].\cos(\theta_v) + \left[-bd - ac + cl.\cos(\theta_m) + dl.\sin(\theta_m)\right].\sin(\theta_v) = \frac{a^2 + b^2 + c^2 + d^2}{2} - al.\cos(\theta_m) - bl.\sin(\theta_m)$$

APPLICATION

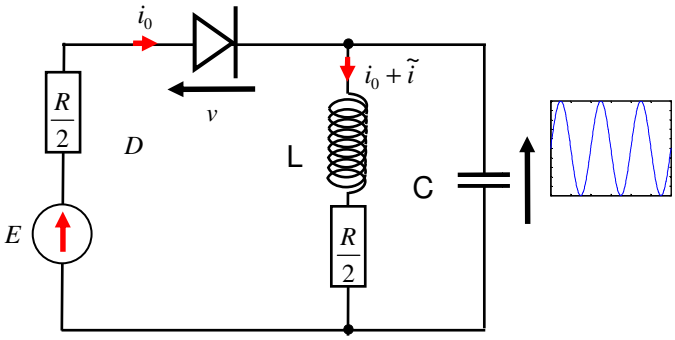
Q. *Sur la base de ce qui a été développé précédemment, concevoir un code Scilab permettant la résolution de cette équation par dichotomie, pour θ_m variant entre 0 et 135°*

3. ACTIVITE 2 : OSCILLATEUR HARMONIQUE REALISE AVEC UNE DIODE TUNNEL

Une diode tunnel est un dipôle semi-conducteur. L'effet tunnel qui donne son nom à ce composant, est un effet quantique qui a pour conséquence « macroscopique » une augmentation de la tension à ses bornes qui s'accompagne d'une diminution du courant la traversant. Cette zone de sa caractéristique courant-tension (figure 1) correspond à une résistance dynamique négative.

Cette propriété est avantageusement mise en œuvre dans les oscillateurs harmoniques afin de compenser les éléments dissipatifs.

Le circuit ci-dessous est un oscillateur harmonique pour lequel les pertes induites par les composants dissipatifs (les résistances) sont compensées par une diode tunnel correctement polarisée par les éléments R et la source E .



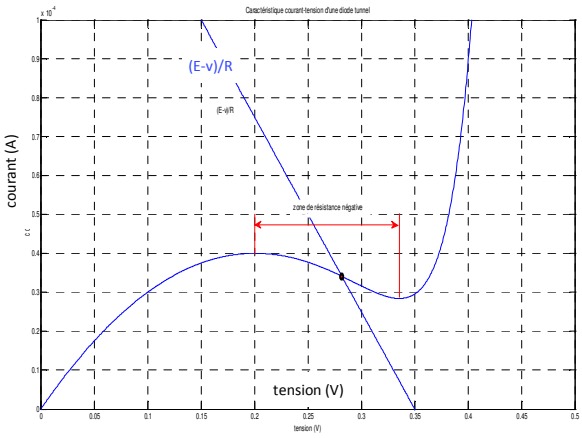
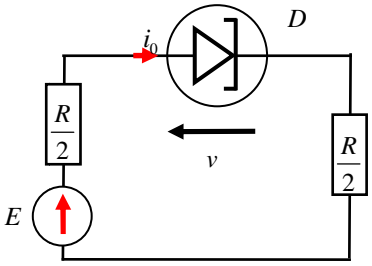


Figure 1 : Caractéristique courant tension

La polarisation d'un circuit correspond à la composante continue i_0 sur laquelle sera superposée la composante variable $\tilde{i}$. La composante continue est calculée lors de l'étude statique du circuit. Elle est décorrélée de la composante variable. Ainsi, en régime statique, le schéma est équivalent au schéma ci-contre.



Objectif de cet exercice : On se propose de calculer le point de fonctionnement de la diode.

La caractéristique statique courant-tension de la diode D a pour expression :

$i = \alpha\left(e^{v/\beta} - 1\right) - \mu v(v - \gamma)$ avec i $[A]$ et v en $[V]$.

Les constantes prennent les valeurs suivantes pour la diode choisie :

$\alpha = 10^{-12}A,\ \mu = 10^{-3}A \cdot V^{-2},\ \gamma = 0{,}4\ V, \beta = 40\ V, E = 0{,}35V\ et\ R = 2000\ \Omega.$

L'application de la loi des mailles permet d'obtenir une seconde expression du courant dans la diode : $i = \frac{E-v}{R}$

En conséquence, par identité nous avons : $i = \alpha\left(e^{v/\beta} - 1\right) - \mu v(v - \gamma) = \frac{E-v}{R}$

Finalement : $$\alpha\left(e^{v/\beta} - 1\right) - \mu v(v - \gamma) - \frac{E}{R} + \frac{v}{R} = 0$$

APPLICATION

Q. *En vous aidant de la figure 1, proposez un protocole qui permettra d'estimer le point de fonctionnement de la diode tunnel en utilisant l'algorithme de Newton-Raphson.*

TP7 : ELEMENTS DE CORRECTION

1. ALGORITHMES DE RECHERCHE PAR DICHOTOMIE

1.2. RECHERCHE DICHOTOMIQUE DANS UN TABLEAU TRIE

- **ALGORITHME**

ALGORITHME

Fonction recherche_dichotomique
{ }

```
# Déclaration des variables
ENTREES: T : Tableau trié d'entiers
ENTREES: x (valeur recherchée) : ENTIER

DEBUT
    # Initialisation des variables
    g,d ← 0, longueur(T)-1

    # BOUCLE PRINCIPALE TANT QUE
    TANT QUE g <= d
        m ← ( g+d ) // 2        # quotient dans la division entière

        SI T[m] ==x
          RETOURNE m
        SI T[m]  < x
          g ← m + 1
        SINON
          d ← m – 1
        FIN SI
    FIN TANT QUE
    RETOURNE None
FIN
```

- **COMPLEXITE DE L'ALGORITHME**

Montrons maintenant que la complexité de cet algorithme est au pire $O(\log n)$ où n est la longueur du tableau. En particulier, on effectue au pire un nombre logarithmique de comparaisons. La démonstration consiste à établir qu'après k itérations de la boucle, on a l'inégalité

$$\mathsf{d} - \mathsf{g} < \frac{n}{2^k}.$$

La démonstration se fait par récurrence sur k. Initialement, on a $\mathsf{g} = 0$ et $\mathsf{d} = n - 1$ et $k = 0$, donc l'inégalité est établie. Supposons maintenant l'inégalité vraie au rang k et $\mathsf{g} \leqslant \mathsf{d}$. À la fin de la $k+1$-ième itération, on a soit $g = m+1$, soit $d = m-1$. Dans le premier cas, on a donc

$$\mathsf{d} - \left(\left\lfloor \frac{\mathsf{g}+\mathsf{d}}{2} \right\rfloor + 1\right) \leqslant \mathsf{d} - \frac{\mathsf{g}+\mathsf{d}}{2} = \frac{\mathsf{d}-\mathsf{g}}{2} < \frac{n}{2^k \times 2} = \frac{n}{2^{k+1}}.$$

Le second cas est laissé au lecteur. On conclut ainsi : pour $k \geqslant \log_2(n)$, on a $\texttt{d} - \texttt{g} < 1$, c'est-à-dire $\texttt{d} - \texttt{g} \leqslant 0$. On fait alors au plus une dernière itération.

La complexité de la recherche dichotomique est donc $O(\log n)$, alors que celle de la recherche par balayage est $O(n)$. Il ne faut cependant pas oublier qu'elles ne s'appliquent pas dans les mêmes conditions : une recherche dichotomique est exclue si les données ne sont pas triées.

VARIANT DE BOUCLE : TERMINAISON DE L'ALGORITHME

RAPPEL DEMARCHE : montrer qu'un algorithme se termine à l'aide d'un variant de boucle

Une quantité vérifiant bien les deux propriétés : être un entier positif tout au long de l'algorithme et décroître strictement après chaque itération constitue un ***variant de boucle***.

L'identification d'un variant de boucle permet de s'assurer de la sortie effective de la boucle.

d-g ici

INVARIANT DE BOUCLE : CORRECTION DE L'ALGORITHME

RAPPEL DEMARCHE : montrer qu'un algorithme est correct à l'aide d'un invariant de boucle

On utilise une invariant de boucle, c'est-à-dire une propriété

- qui est vérifiée avant d'entrer dans la boucle,
- qui si elle est vérifiée avant une itération est vérifiée après celle-ci,
- qui lorsqu'elle est vérifiée en sortie de boucle permet d'en déduire que le programme est correct.

On maintient l'invariant suivant : les valeurs strictement à gauche de g sont inférieures à x et les valeurs strictement à droite de d supérieures à x.

SCRIPT PYTHON

```
def recherche_dichotomique(x, T):
"""renvoie la position d'une occurrence de x dans T,
supposée trié, si elle existe, et None sinon"""

# initialisation des variables g et d
g,d = 0,len(T)-1

# boucle principale
while g <= d:
    m = (g + d) // 2      # division entière
    if a[m] == x:
        return m
    elif a[m] < x:
        g = m+1
    else:
        d = m-1
return None
```

1.3. RECHERCHE DICHOTOMIQUE DU ZERO D'UNE FONCTION

Dans find_root.py, on fournit une fonction find_root_binsearch qui s'occupe de tester la précondition (f(a) ≤ 0 et f(b) ≥ 0), et appelle find_root_binsearch_internal. Cette deuxième fonction est **récursive**, et ne refera pas les tests de précondition.

- La fonction find_root_binsearch a un argument precision qui ne changera pas pendant le calcul.
- La fonction find_root_binsearch_internal ne prend pas d'argument precision ; elle utilise directement l'argument de find_root_binsearch. Vu que c'est une **fonction imbriquée**, les variables et paramètres de la fonction englobante restent visibles.
- L'argument f de la fonction find_root_binsearch est lui-même une fonction. Ça ne pose pas de problème en Python : on peut manipuler f soit comme une variable (g = f), soit comme une fonction (f(x)).

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import math

def find_root_binsearch(f, min, max, precision):
    if f(min) > 0:
        raise ValueError("f(min=%f) is not negative" % min)
    if f(max) < 0:
        raise ValueError("f(max=%f) is not positive" % max)

    def find_root_binsearch_internal(f, min, max):
       if max - min < precision:
            return min
        pivot = (min + max)/2
        value = f(pivot)
        if value < 0:
            return find_root_binsearch_internal(f, pivot, max)
        elif value > 0:
            return find_root_binsearch_internal(f, min, pivot)
        else:
            return pivot

    return find_root_binsearch_internal(f, min, max)

print("sin(%.10f) ~ 0" % find_root_binsearch(math.sin, 4.0, 7.0, 1e-10))
print("sqrt(2) ~", find_root_binsearch(lambda x: x * x - 2, 0.0, 10.0, 1e-5))
```

L'instruction raise permet de déclencher des exceptions.

raise <nom> # déclenche une exception manuellement.
raise <nom>, <données> # idem mais avec des données.

L'instruction lambda utilisée dans l'évaluation de sqrt(2) permet de définir une fonction ($f(x)=x^2-2$ ici)**.**

2. ALGORITHME DE RECHERCHE D'UN MOT DANS UNE CHAINE DE CARACTERES

- On effectue la recherche avec une boucle for, qui va considérer toutes les positions possibles pour le mot m, c'est-à-dire tous les indices i entre 0 et len(t) - len(m) , au sens large.

```
def recherche_mot(m, t):
for i in range(1 + len(t) - len(m)):
```

- On teste si le mot m apparaît à la position i avec une **seconde boucle**, qui compare les caractères de m et de t un à un.
  On utilise une variable j pour cela et on s'arrête soit lorsque j atteint len(m) , soit lorsque les caractères diffèrent :

```
j = 0
while j < len(m) and m[j] == t[i + j]:
j += 1
```

- Une fois sorti de la boucle while, on a reconnu le mot m à la position i si et seulement si j == len(m) , auquel cas on renvoie i :

```
if j == len(m):
return i
```

- Sinon, on passe à la valeur suivante de i . Si on parvient à la fin de la boucle for principale, c'est qu'il n'y a pas d'occurrence de m dans t , ce que l'on signale en renvoyant None :

```
return None
```

```
def recherche_mot(m, t):
"""renvoie la position de la première occurrence du mot m
dans le texte t, et renvoie None sinon"""

for i in range(1 + len(t) - len(m)):
    j = 0
    while j < len(m) and m[j] == t[i + j]:
        j += 1
    if j == len(m):
        return i
return None
```