université
PARIS DIDEROT
PARIS 7

L3 MIASHS - OPTIMISATION - 2018-2019

TP1 : MÉTHODES DE DICHOTOMIE ET DE NEWTON

L'objectif de ce TP est d'apprendre à utiliser le logiciel Scilab à travers deux exemples introductifs: la méthode de dichotomie et la méthode de Newton. Il s'agit de deux méthodes itératives d'approximation d'un zéro d'une fonction réelle à valeurs réelles. Dans toute la suite de l'introduction, on considère une fonction $f : \mathbb{R} \to \mathbb{R}$ et on cherche une approximation d'un point $c \in \mathbb{R}$ vérifiant $f(c) = 0$.

Méthode de dichotomie

Cette méthode consiste à obtenir un encadrement du point c recherché, entre deux nombres a et b tels que $f(a)$ et $f(b)$ sont de signes opposés, et l'écart $b - a$ est inférieur à un nombre ε petit. Si f est continue, le théorème des valeurs intermédiaires garantit alors que f s'annule au moins une fois entre a et b.

> **Initialisation.** On définit une tolérance $\varepsilon > 0$.
> On se donne $a, b \in \mathbb{R}$ tels que $a < b$, $a + \varepsilon < b$ et $f(a)f(b) < 0$.
> **Itérations.** Tant que $b - a > \varepsilon$, on pose $c := \frac{a+b}{2}$.
> Si $f(a)f(c) \leqslant 0$, on réalise l'affectation $b := c$. Sinon, on pose $a := c$.

Méthode de Newton

La méthode de Newton est une méthode d'approximation d'un point c vérifiant $f(c) = 0$ et $f'(c) \neq 0$. Si f est de classe C^1, on sait alors que $f'(x)$ sera non nul si x est proche de c.

> **Initialisation.** On fixe une tolérance $\varepsilon > 0$ et on se donne un point $x \in \mathbb{R}$.
> **Itérations.** Tant que $|f(x)| > \varepsilon$, on remplace x par $x - \frac{f(x)}{f'(x)}$.

On considère la fonction f définie par $f(x) = x^2 - 2$ pour $x \in \mathbb{R}$. On cherche à approcher le nombre $\sqrt{2}$, c'est-à-dire l'unique réel $c \geqslant 0$ tel que $f(c) = 0$.

(1) Définir dans Scilab une fonction `Dichotomie`, sous la forme suivante :

```
function [c,n]=Dichotomie(a,b,eps)
```

qui détermine l'approximation `c` de $\sqrt{2}$ obtenue par l'algorithme de dichotomie, appliqué aux données initiales `a,b` et avec la tolérance `eps`. La variable de sortie `n` correspond au nombre d'itérations qui ont été réalisées avant que le critère d'arrêt ne soit atteint.

Tester cette fonction avec $a = 1$, $b = 2$ et avec différentes valeurs de la tolérance $\varepsilon = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$... Noter le nombre d'itérations n et contrôler le résultat en calculant `c-sqrt(2)`. Que se passe-t-il si l'on remplace $a = 1$ par $a = -4$? Comment l'expliquer ?

(2) Modifier la fonction `Dichotomie` pour qu'elle fournisse également en sortie une matrice colonne `err`, donc les lignes contiennent l'erreur $|c - \sqrt{2}|$ à chaque itération. On pourra utiliser la commande `abs` et l'instruction `err=[err;val]`; pour ajouter une valeur `val` à la matrice colonne `err`.

(3) En utilisant la fonction modifiée précédemment, représenter graphiquement le vecteur `err` obtenu lorsque $a = 0$, $b = 2$, $\varepsilon = 10^{-6}$. Refaire le tracé en utilisant une échelle logarithmique pour les ordonnées. On pourra utiliser l'instruction

```
plot2d(err,logflag='nl',style=-1);
```

(4) Programmer une fonction `Newton` sous la forme

```
function [c,n]=Newton(x,eps,nmax)
```

qui prend en entrée un point de départ `x` et une précision `eps`, et renvoie l'approximation `c` calculée par la méthode de Newton, ainsi que le nombre d'itérations `n` réalisées. La variable d'entrée `nmax` fixe le nombre maximal d'itérations autorisé ; elle permet d'empêcher le programme d'entrer dans une boucle infinie si la méthode venait à diverger.

(a) Tester la fonction avec $\varepsilon = 10^{-4}$, $n_{max} = 100$ et différents points de départ, par exemple $x = 1, 1.3, 1.5, 1.7$... Peut-on partir de $x = 0$? D'après vos essais, quel est l'ensemble des x pour lesquels la méthode de Newton converge vers $\sqrt{2}$? Vers $-\sqrt{2}$?

(b) Pour la même précision ε, comparer le nombre d'itérations nécessaires pour obtenir la convergence des méthodes de dichotomie et de Newton. On choisira par exemple $\varepsilon = 10^{-3}, 10^{-4}, \ldots, 10^{-7}$, $a = 1$, $b = 2$, $x = 1$. Quelle méthode converge le plus rapidement ?

(c) En adaptant les questions (2) et (3), représenter l'erreur en fonction de l'itération, en échelle logarithmique pour les ordonnées, pour la méthode de Newton, en fixant $\varepsilon = 10^{-7}$ et pour différents points de départ x.

(5) Modifier la fonction `Newton` pour qu'elle calcule une approximation du zéro de la fonction $g(x) = x \exp(-x)$ définie sur $\mathbb{R}$. Déterminer numériquement une valeur $x_0 > 0$ telle que, si le point de départ x de la méthode appartient à l'intervalle $]0, x_0[$, la méthode de Newton converge, et si $x \geqslant x_0$, la méthode diverge.

Démarrage rapide avec Scilab

Une matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ se définit avec `A = [1 2 3;4 5 6];` Noter que cette commande n'affiche pas de résultat, car elle se termine par un point-virgule ";". Si on veut voir le résultat d'une fonction, ou d'une expression, il faut conclure la commande avec une virgule ",".
La définition d'un vecteur se fait de manière similaire, car Scilab considère que les vecteurs de taille n sont des matrices de taille $n \times 1$.

Quelques commandes de base pour la manipulation de matrices:

- Transposée d'une matrice : `A'` (Utile pour passer d'un vecteur ligne à colonne).
- Produit de deux matrices (ou matrice fois vecteur): `A*B`
- Puissance d'une matrice: `A^3`
- Inverse d'une matrice: `A^-1` ou `inv(A)`
- Solution de l'équation $Ax = b$: `A\b` (Attention: le vecteur doit être en colonne!)
- Déterminant d'une matrice: `det(A)`
- Spectre d'une matrice: `S=spec(A)` retourne un vecteur contenant toutes les valeurs propres de A rangées par ordre croissant.
- Norme euclidienne d'un vecteur: `norm(x)`

Quelques commandes pour générer des matrices:

- Matrice identité de taille $n \times n$: `eye(n,n)`
- Matrice aléatoire de taille $n \times m$: `rand(n,m)`
- Vecteur défini par progression arithmétique: `I=[3:0.1:4]` renverra le vecteur ligne `[3 3.1 3.2 ... 3.9 4]`

Syntaxe de la boucle `for`:

```
for i=1:100,
    INSTRUCTIONS;
end
```

Écrire et utiliser une fonction dans un fichier:

(1) Utiliser le navigateur de fichiers de Scilab pour se placer dans un dossier dans lequel on va travailler.
(2) Depuis la console Scilab, cliquer sur l'icône "Démarrer Scinotes".
(3) Écrire sa fonction dans Scinotes. La syntaxe est la suivante:
```
function [variables-de-sortie] = nom-de-la-fonction(variables-entrantes)
    INSTRUCTIONS;
endfunction
```
Si besoin est, on commentera une ligne en la faisant commencer par \\
(4) Enregistrer le fichier dans le dossier, en lui donnant le nom de la fonction.
(5) Cliquer sur l'icône "Exécuter".
(6) On peut alors faire appel à la fonction depuis la console ou d'autres fonctions.

Astuce: si Scilab mouline sur un calcul trop difficile, on peut tuer l'exécution depuis la console via Contrôle > Abandonner.