Université des Sciences et de la Technologie d'Oran MB
Faculté de Génie Électrique – Département d'Electronique
Licence LMD L2 (Semestre 4)
Filières : Electronique / Génie Biomédical / Télécommunication

Module : Méthodes Numériques

TP 1 : Résolution des équations non linéaires

Durée du TP : 2 séances de 1h30

But du TP :

Durant ce TP, nous allons mettre en œuvre les algorithmes des méthodes de résolution des équations non linéaires étudiées pendant le cours : la ***bipartition***, ***Regula-Falsi***, les ***approximations successives*** et ***Newton-Raphson***.

Rappel sur les différentes méthodes :

***La bipartition* :**

1- Existence et unicité de la solution :
Si la fonction $f(x)$ est définie et continue et strictement monotone sur l'intervalle (a, b) et que $f(a) \times f(b) < 0$, alors $f(x)$=0 n'a qu'une solution x^* dans cet intervalle.

2- Approximation de la solution:
On calcule c par l'expression :

$$c = \frac{a+b}{2}$$

On compare ensuite $f(c)$ avec $f(a)$ et $f(b)$ pour déterminer l'intervalle de la solution et on recommence le calcul de c itérativement jusqu'à ce que :
$|x_n - x_{n-1}| < \varepsilon$.

Le nombre n d'itérations nécessaire pour avoir une approximation de la solution à ε près est : $n \geq \frac{Log\left(\frac{b-a}{\varepsilon}\right)}{Log(2)}$

Regula-Falsi

On calcule c par l'expression :

$$c = \frac{a\,.f(b) - \ b\,.f(a)}{f(b) - \ f(a)}$$

On compare ensuite $f(c)$ avec $f(a)$ et $f(b)$ pour déterminer l'intervalle de la solution et on recommence le calcul de c itérativement jusqu'à ce que :
$|x_n - x_{n-1}| < \varepsilon$.

Les approximations successives

Il faut réécrire l'équation $f(x) = 0$ sous la forme $x = g(x)$,

La condition de convergence suffisante mais pas nécessaire est :

$$|g(x)| < 1 \; pourtout \; x \in [a \quad b]$$

Pour approximer la solution de l'équation

On part de la valeur initiale $x_0 = \cdots$,
On calcule itérativement les valeurs de x_n par :

$$x_n = g(x_{n-1})$$

Les critères d'arrêts peuvent être

$|x_n - x_{n-1}| < \varepsilon$

$\frac{|x_n - x_{n-1}|}{|x_n|} < \varepsilon$

$|f(x_n)| < \varepsilon$

Newton-Raphson

L'algorithme de *Newton-Raphson* est :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Les critères d'arrêts peuvent être

$|x_n - x_{n-1}| < \varepsilon$

$\frac{|x_n - x_{n-1}|}{|x_n|} < \varepsilon$

$|f(x_n)| < \varepsilon$

Méthodologie :

Pour mettre en œuvre les algorithmes, nous allons écrire des programmes (scripts) sous Matlab. Un script (ou .m) est un fichier ASCII qui contient une succession d'instructions et d'opérations, et exécutable depuis la fenêtre de commande ou de la fenêtre d'édition.

Pour écrire un script, il suffit de sélectionner l'icône « New Script » dans le menu principal du Matlab. A l'ouverture d'une fenêtre d'édition (Editor), on saisit le programme puis on le sauvegarde dès qu'on termine. Pour l'exécuter, on appui sur le triangle vert (Run) depuis la fenêtre d'édition ou bien en tapant le « nom du programme » depuis la fenêtre de commande (Command Window).

On rappelle, la syntaxe des structures de contrôle et de répétition:

- L'instruction de test ***if*** s'écrit comme suit :

```
if condition1    action1
    elseif  condition2   action2
    ...
    else    action3
end
```

- Les boucles :
  - La boucle ***for*** :

```
forcompteur = valeur_initiale : incrément : valeur_finale
        action
end
```

  - la boucle ***while*** :

```
while condition
        action
end
```

Pour avoir une aide instantanée sur une commande Matlab, l'étudiant peut taper la commande « help » suivie du nom de la commande, dans la fenêtre de commande de Matlab (Command Window).

Manipulations :

Partie A : Rappel sur le tracé de graphiques 2D à l'aide de la fonction 'plot'

1. Former une liste de valeurs (un vecteur) x allant de 0 à 2×pi avec un pas de pi/20 ;
2. Calculer y = cos(x) et z = sin (x) ;
3. Tracer la courbe de y = cos(x) ;
4. Tracer la courbe de z = sin (x) ;
5. Tracer les deux courbes dans un même graphe.
   La fonction graphique plot( ) peut contenir plusieurs couples de points :
   plot (a,b,a,c,a,d) trace les courbes de b, c, et d en fonction de a ;
6. Exécuter les commandes suivantes :
   - xlabel ('Abcisses') ;
   - ylabel('Ordonnées') ;
   - title('Les fonctions Cos et Sin') ;
   - grid
   - legend('cosinus','sinus') ;

Partie B : Résolution des équations non linéaires

Soit à résoudre l'équation : $f(x) = x + 2\ln(x) = 0$ où $x \in]0, +\infty[$

a) Tracer le graphe $y = f(x)$ sur un intervalle tel qu'il vous permet de localiser la solution de l'équation.
b) Localiser la solution dans le plus petit intervalle $]a, b[$ possible.

1. Ecrire un script, que vous appellerez « *bipart.m* » qui implémente **la méthode de bipartition** suivant les étapes :
   - Initialiser les limites du domaine de recherche a et b ;
   - Initialiser un compteur d'itération k à 0 ;
   - Ecrire l'algorithme de bipartition en incrémentant le compteur k à chaque passage de boucle ;
   - Arrêter la boucle quand la largeur du domaine devient $\leq 10^{-5}$;
   - Afficher la solution calculée ainsi que le nombre d'itérations.
2. Ecrire un autre script, que vous appellerez « *RegFal.m* » qui implémente **la méthode de Regula-Falsi**. Il faut faire attention ici au critère d'arrêt. En effet, la précision n'est pas comparée à la largeur du domaine final, comme en (1), mais c'est la différence entre deux valeurs consécutives de la solution.
3. Ecrire un autre script, que vous appellerez « *Newton.m* » qui implémente **la méthode de Newton-Raphson**. L'arrêt des itérations se fera de la même manière que dans la manipulation (2).
4. Ecrire un autre script, que vous appellerez « *AppSuc.m* » qui implémente **la méthode des approximations successives**. Pour cela, on doit réécrire l'équation sous la forme $x = g(x)$, en assurant la convergence de l'algorithme. L'arrêt des itérations se fera de la même manière que dans la manipulation (2).
5. Comparer les différentes méthodes implémentées.