UNIVERSITE MOHAMMED PREMIER
ECOLE NATIONALE DES SCIENCES APPLIQUEES
OUJDA-MAROC

Filière : GIndus3 — Année universitaire : 2020-2021
Module : Méthodes Numériques — Enseignant : M. Derouich

Travaux Pratiques d'Analyse Numérique Avec *MATLAB*

1 Interpolation d'une fonction continue par un polynôme

1.1 Position du problème

Le but de cet exercice est de compléter un programme Matlab permettant d'interpoler une fonction continue par un polynôme .
Soit $f : [a, b] \to \mathbb{R}$ une fonction continue donnée et soit $t_0 < t_1 < t_2 <, \cdots, < t_n$, (n+1) points de [a,b]. On cherche un polynôme p de degré n tel que

$$P(t_j) = f(t_j), 0 <= j <= n.$$

D'après le cours le polynôme p est donné par :

$$p(t) = \sum_{j=0}^{n} f(t_j)\varphi_j(t),$$

Où $\varphi_0, \varphi_1, \cdots, \varphi_n$ est la base de Lagrange de p associée à $t_0, t_1, \cdots, tn$. Les fonctions de base de Lagrange sont définies par

$$\varphi_j(t) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{(t - t_i)}{(t_j - t_i)} \tag{1.1}$$

1.2 Travail demandé

A partir la donnée de a, b, f, n le programme Lagrange permet de calculer le polynôme p qui interpole f aux points $t_j, 0 \leq j \leq n$, équidistribués uniformément sur l'intervalle $[a, b]$.
Ce programme est incomplet, en particulier vous devez programmer la formule (1.1).
Une fois le programme écrit et compilé. Son exécution aura pour conséquence de calculer la valeur de p à des points donnes par l'utilisateur et comparer le résultat avec la valeur de f en ce point.

Exemple :

```
%Etant donne une fonction continue f
%Le programme interpole la fonction f par un polynôme
%p aux points xData(j).
%xData(j) : est le j-eme point d'interpolation.
%yData(j) : la valeur de la fonction f au point xData(j).
xData = [-1;-0.5;0;0.5;1]; yData = [-1.5;0;0.25;0;0]; n
=length(xData);
k=1;
for x = -1.2: 0.001: 1.2
     pol=0;
        for j= 1: n
            %----------- calcul de phi_j( la le j eme fonction de base) au point x ------
          phi_j=1;
             for i = 1 : n
                if(i~=j)
                     phi_j=phi_j*(x-xData(i))/(xData(j)-xData(i));
                end;
             end;
             %-- évalue pol au point x --
             pol=pol+phi_j*yData(j);
         end;
        X(k)=x;
        P(k)=pol; %P(x)
        k=k+1;
        fprintf('X(%d)=%2.3f        P(%2.3f)=%2.10f  \n',k,x,x,pol)
end
    %----------Construction du graphe du polynôme d'interpolation-----------
plot(xData,yData,'*',X,P,'-')
 h =legend('points d''interpolation','l''interpolant de f',1);
 hold on
 yyy=[-3:0.01:2];
 xx=[-1.5:0.001:1.5];
 plot(xx,0,'r--',0,yyy,'r--')
 hold on
```

2 Intégration numérique

2.1 Formule du trapèze

2.1.1 Position du problème

Soit $f : [a, b] \to \mathbb{R}$ une fonction continue donnée. Le but de cet exercice est de compléter et d'utiliser un programme permettant d'approximer

$$\int_a^b f(x)dx \tag{2.1}$$

en utilisant la formule du trapèze à 2 points d'intégration. Pour ce faire l'intervalle $[a, b]$ est partitionné en N intervalles $[x_i, x_{i+1}], i = 0, 1, 2, \cdots, N-1$ de même longueur $h = \frac{b-a}{N}$.

2.1.2 Travail demandé

A partir de la donnée de f, a, b, N le programme trapeze permet d'approcher (2.1). Ce programme est incomplet, en particulier vous devez :

1. monter que :
$$L_h(f) = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{N-1} f(x_i)\right] \tag{2.2}$$
2. programmer la formule (2.2).

Une fois le programme écrit et compilé. Son exécution aura pour conséquence de calculer (2.1).

```
%Etant donné un intervalle [a,b] et une fonction continue f, le
%programme approche l'intégral de f entre a et b
%Paramètre :
%a,b :l'intervalle [a,b]
%n : l'intervalle [a,b] est partionné en n intervalles
%f : la fonction à intégrer
%
f=inline('1./(1.+x)');
 a=input('donner a :');
  b=input('donner b :');
  n =100;
%--------- Calcul de la valeur approché de l'intégral de f ---
```

```
h=(b-a)/n;
ST=h*(f(a)+f(b))/2;
for i=1:n-1
    xi=a+i*h;
    ST= ..... ; % A COMPLETER
end
 fprintf('%1.20f      \n',ST)
```

2.2 formule de Simpson

2.2.1 Position du problème

Soit $f : [a, b] \to \mathbb{R}$ une fonction continue donnée.

Le but de cet exercice est de compléter et d'utiliser un programme permettant d'approximer

$$\int_a^b f(x)dx \tag{2.3}$$

en utilisant la formule de Simpson à 2 points d'intégration.
Pour ce faire l'intervalle $[a, b]$ est partitionné en N intervalles $[x_i, x_{i+1}], i = 0, 1, 2, \cdots, N-1$ de même longueur $h = (b-a)/N$.

2.2.2 Travail demandé

A partir de la donnée de f, a, b, N le programme Simpson permet d'approcher (2.3). Ce programme est incomplet, en particulier vous devez : 1

1. monter que :

$$L_h(f) = \frac{h}{3}\left[f(a) + f(b) + 2\sum_{i=1}^{n/2-1} f(x_{2i}) + 4\sum_{i=1}^{n/2} f(x_{2i-1})\right] \tag{2.4}$$

2. programmer la formule (2.4).

Une fois le programme écrit et compilé. Son exécution aura pour conséquence de calculer la valeur de (2.3), et comparer le résultat avec le résultat du programme trapeze.

```
%Etant donné un intervalle [a,b] et une fonction continue f, le
%programme approche l'intégral de f entre a et b
%Paramètre :
%a,b :l'intervalle [a,b]
%n : l'intervalle [a,b] est partionné en n intervalles (n est pair)
```

```
%f : la fonction à intégrer
%
f=inline('1./(1.+x)');
a=input('donner a :');
b=input('donner b :');
% ------------------ n est pair --------------------
n=100;
p=n/2;
%--------- Calcul de la valeur approché de l'intégral de f ---
h=(b-a)/(2*p);
S1=0;
S2=0;
%--------------- Formule de simpson A COMPLETER -----------
for i=1:p
xi1=a+ ..... ; % A COMPLETER
S1=S1+f(xi1);
end
for i=1:p-1
xi2=a+ ..... ; % A COMPLETER
 S2=S2+f(xi2);
 end
SS=h/3*(f(a)+f(b)+4*.....+2*.....);  % A COMPLETER
fprintf('SS=%1.20f\n',SS)
```

3 Méthode de Newton

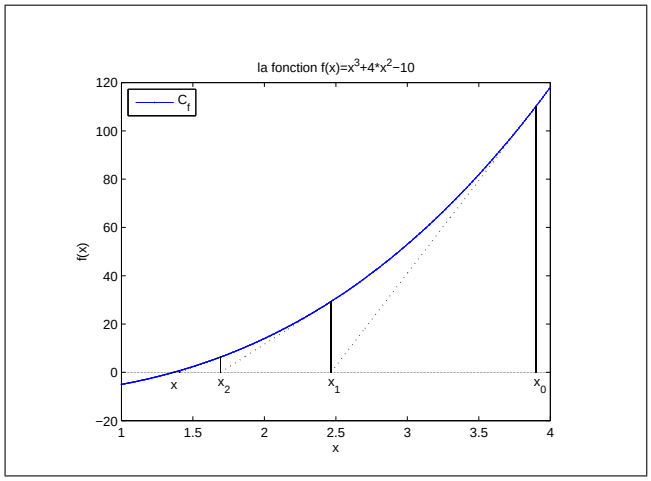
3.1 Position du problème :

Le but de cet exercice est de compléter et d'utiliser un programme permettant de résoudre l'équation $f(x) = 0$ grâce à la méthode de Newton.

La méthode

L'approche graphique est la suivante : On choisit une valeur d'abscisse raisonnablement proche du vrai zéro. On remplace alors la courbe par sa tangente et on calcule le zéro de l'approximation affine associée à la tangente (ce qui se réalise facilement avec l'algèbre élémentaire). Ce zéro de la tangente sera généralement plus proche du zéro de la fonction, et la méthode peut être réitérée.

En pratique, voici les opérations à poser pour $f : [a, b] \to \mathbb{R}$, fonction définie et dérivable



et sur l'intervalle $[a, b]$, et à valeurs réelles. Choisissons une valeur arbitraire x_0 (le plus près du zéro est le mieux). On définit alors, par récurrence pour chaque nombre naturel n :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.1}$$

où f' désigne la dérivée de la fonction f.
On peut prouver que, si f' est continue et si le zéro inconnu a est isolé, alors il existe un voisinage de a tel que pour toutes les valeurs de départ x_0 dans ce voisinage, la suite (x_n) va converger vers a. De plus, si $f'(a) \neq 0$, alors la convergence est quadratique, ce qui signifie intuitivement que le nombre de chiffres corrects est approximativement doublé à chaque étape.

3.2 Travail demandé :

A partir du programme Newton et la donnée de f, x_0 , ce programme résoud l'équation $f(x) = 0$. Ce programme est incomplet, en particulier vous devez programmer la formule 3.1.

Exemple 3.1. :
Considérons le problème de trouver le nombre positif x vérifiant

$$cos(x) = x^3 \tag{3.2}$$

1. Montrer $cos(x) = x^3 \Longleftrightarrow f(x) = 0$ avec $f(x) = cos(x) - x^3$
2. Montrer que l'équation $f(x) = 0$ admet une solution unique $\theta \in [0, 1]$.
3. Compléter le script suivant, que vous appellerez "**Newton.m**", qui calcule la racine de l'équation : $f(x) = 0$.

Nous essayons une valeur de départ de $x_0 = 0,5$. et tolérance $= 10^{-10}$

```
%Etant donné une fonction f et df sa dérivée
%NMAX: le nombre maximal d'itérations
%x0: la valeur initiale
%eps: la précision
f=inline('cos(x)-x^3');
df=inline('-sin(x)-3*x^2');
NMAX=200;
x0=0.05;
k=1;
eps=10^(-10);
err=eps+1;
fprintf('x%d =%21.2f \n',k-1,x0)
while((err>eps)&(k<=NMAX))
       x = x0 -(f(x0)/df(x0));
       fprintf('x%d =x%d-f(x%d)/df(x%d)= %2.20f    f(x%d)=%2.20f  \n',k,k-1,k-1,k-1,x,k,f(x))
       err=abs(x-x0);
       x0=x;
       k=k+1;
 end
fprintf('La racine de f est x=%2.12f\n',x)
```

Remarque 3.1. *:*

1. *Pour créer des fonctions sous Matlab on a deux manières :*
   - *En utilisent la commande* **inline** *: Ainsi* **f=inline('x-cos(x)-1')** *affecte à la variable f la fonction $f(x) = x - cos(x) - 1$*
   - *Ecrire dans un fichier* **f.m** *les deux lignes*

     ```
     function z=f(x)
     z=x-cos(x)-1;
     ```

2. *La fonction f et sa dérivé df sont définies par l'utilisateur comme suit :*

   -La fonction f

   ```
   function s=f(x)
        s=cos(x)-x.^3;
   ```

   -La fonction df

   ```
   function[s]=df(x)
       s=-sin(x)-3*x.^2;
   ```