Université des Sciences et de la Technologie d'Oran –Mohamed Boudiaf–

Faculté de Génie Electrique

Département d'Electronique

Polycopié de Travaux Pratiques :

Méthodes Numériques

Licence 2ème Année

Filières : Electronique / Génie Biomédical / Télécommunication

Année universitaire 2016-2017

Dr. S. Karoui

Avant-propos

Ce polycopié se trouve être un support pédagogique de travaux pratiques, destiné aux étudiants de 2ème Année des Licences LMD assurées au département d'électronique. Ce polycopié regroupe un certain nombre de méthodes étudiées dans les différents chapitres du cours de méthodes numériques. L'objectif de ces TP est d'implémenter sous MATLAB les différentes méthodes pour les comparer et mieux les comprendre.

Références

1- C. Brezinski, Introduction à la pratique du calcul numérique, Dunod, Paris 1988.
2- G. Allaire et S.M. Kaber, Algèbre linéaire numérique, Ellipses, 2002.
3- G. Allaire et S.M. Kaber, Introduction à Scilab. Exercices pratiques corrigés d'algèbre linéaire, Ellipses, 2002.
4- G. Christol, A. Cot et C.-M. Marle, Calcul différentiel, Ellipses, 1996.
5- M. Crouzeix et A.-L. Mignot, Analyse numérique des équations différentielles, Masson, 1983.
6- S. Delabrière et M. Postel, Méthodes d'approximation. Équations différentielles. Applications Scilab, Ellipses, 2004.
7- J.-P. Demailly, Analyse numérique et équations différentielles. Presses Universitaires de Grenoble, 1996.
8- E. Hairer, S. P. Norsett et G. Wanner, Solving Ordinary Differential Equations, Springer, 1993.
9- P. G. Ciarlet, Introduction à l'analyse numérique matricielle et à l'optimisation, Masson, Paris, 1982.
10- M. Lakrib, Cours d'analyse numérique, OPU Alger 2008.
11- B. Demidovitch, I. Maron, Elément de calcul numérique, Ed. MIR, Moscou, 1979.

Table de Matières

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.

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 2 : Interpolation polynômiale

Durée du TP : 1 ou 2 séances de 1h30

But du TP :

Le but de ce TP est l'implémentation des algorithmes d'interpolation étudiés au cours sous Matlab, il sera ensuite question d'étudier un phénomène qui se produit lorsque l'on augmente le nombre de points de collocation.

Manipulations :

A. La méthode de Lagrange

Soit une fonction $f(x) = \ e(x)$ définie sur l'intervalle [a ,b] avec :

a = 3.50
b = 3.70

Ecrire un programme qui :

- Détermine le pas d'interpolation pour un nombre 'n' donné de sous intervalles (n = 4) ;
- Remplit un tableau avec les coordonnées des points d'appui ;
- Interpole la fonction $f(x)$ pour $\boldsymbol{x = 3.62}$ en utilisant la méthode de Lagrange.

  Pour rappel, l'algorithme de Lagrange est comme suit :

$$P_n(x) = \sum_{i=0}^{n} y_i . L_i(x) \qquad o\grave{u} \qquad L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{(x - x_i)}{(x_i - x_j)}$$

- Refaire l'exécution pour n = 10.

B. La méthode de Newton

On considère la même fonction $f(x) = \ e(x)$ définie sur l'intervalle [a, b].
On subdivise cet intervalle en 'n' sous intervalles, ce qui nous donne 'n+1' points d'appui.

Pour n = 4, écrire un programme qui interpole cette fonction avec la méthode de Newton.

Le programme exécutera les étapes suivantes :

- Construire la matrice des différences finies que nous appelons 'D' comme suit :

$$D(i,j) = 0 \qquad pour \quad i = 1..(n+1) \quad et \quad j = 1..(n+1)$$

$$D(i,1) = f(x_i) \qquad pour \quad i = 1..(n+1)$$

$$D(i,j) = \frac{D(i,j-1) - D(i-1,j-1)}{x_i - x_{i-j+1}} pour \quad j = 2..(n+1) \quad et \quad i = j..(n+1)$$

- Extraire ensuite la diagonale de D dans un vecteur M :

$$M(i) = D(i,i) \qquad pour\ i = 1..(n+1)$$

- Interpoler la fonction $f(x)$ pour $\boldsymbol{x = 3.62}$ en utilisant la méthode de Newton :

$$I = M(1) + M(2).(x - x_1) + M(3).(x - x_1).(x - x_2) + \cdots + M(n+1).(x - x_1)\ldots(x - x_n)$$

- Refaire ces étapes pour n = 10.

- Calculer la valeur exacte de la fonction et comparer avec les interpolations précédentes, quel est votre commentaire.

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 3 : Intégration numérique de fonctions

Durée du TP : 1 ou 2 séances de 1h30

But du TP :

Le but de ce TP est le calcul d'une valeur approximative de l'intégrale définie d'une fonction $f(x)$ sur un intervalle$[x_0, x_n]$.

Pour se faire, il existe plusieurs méthodes d'approximation de l'intégrale $\int_{x_0}^{x_n} f(x)dx$, basées sur l'interpolation polynômiale de $f(x)$:

Rappel des méthodes :

La méthode du trapèze : qui n'est autre que l'intégrale du polynôme d'interpolation de $f(x)$ sur 2 points d'appui, par extension à l'ensemble des points, on obtient la formule généralisée suivante :

$$\int_{x_0}^{x_n} f(x)dx \cong \frac{h}{2} \times (y_0 + y_n + 2 \times \sum_{i=1}^{n-1} y_i)$$

La méthode de Simpson : qui n'est autre que l'intégrale du polynôme d'interpolation de $f(x)$ sur 3 points d'appui équidistants, par extension à l'ensemble des points, on obtient la formule généralisée:

$$\int_{x_0}^{x_{2n}} f(x)dx \cong \frac{h}{3} \times (y_0 + y_{2n} + 4 \times \sum_{i=0}^{n-1} y_{2i+1} + 2 \times \sum_{i=0}^{n-1} y_{2i})$$

Travail demandé :

On se propose de calculer l'intégrale définie $I = \int_0^1 \sin(x) \, . \, e^x dx$, en utilisant les deux méthodes suscitées.

1) Ecrire un programme qui calculerait cette intégrale en utilisant la méthode du trapèze et en divisant l'intervalle d'intégration en 10 intervalles élémentaires.

2) Refaire l'exécution pour n=100.

3) Matlab possède une fonction équivalente, c'est la fonction **TRAPZ**, TRAPZ (Y) donne la valeur calculée par la méthode du trapèze sur un vecteur Y avec un pas d'intégration unitaire. Adapter cette fonction pour le calcul de notre intégrale I.

4) Ecrire un deuxième programme qui calculerait cette intégrale en utilisant la méthode de Simpson avec les mêmes conditions que dans (1).

5) Refaire l'exécution pour n=100.

6) Il existe une fonction dans Matlab qui ressemble à la méthode de Simpson, la fonction QUAD qui implémente l'algorithme de Simpson adaptatif.
La syntaxe de cette fonction est : $I = QUAD(nom\_fonction, a\ , b\ )$ où nom_fonction est le nom de la fonction à intégrer, et a et b sont les bornes d'intégration. La fonction doit impérativement accepter des variables de type vecteur. Utiliser cette fonction et comparer avec le résultat obtenu en (3).

7) Comparer les résultats obtenus sachant que la valeur exacte est 0,909330673631479.

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 4 : Résolution des équations différentielles ordinaires

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 différentielles ordinaires étudiées pendant le cours : ***la méthode d'Euler et la méthode de Range Kutta***. Il sera ensuite question de comparer les deux méthodes avec la solution exacte.

Rappel sur les différentes méthodes :

Méthode d'Euler :

$$h = x_i - x_{i-1}$$
$$y_1 = y_0 + hf(x_0, y_0)$$
$$y_2 = y_1 + hf(x_1, y_1)$$
$$y_{i+1} = y_i + hf(x_i, y_i)$$

Méthode de Range Kutta d'ordre 4 :

$$k_1 = f(x_0, y_0)$$
$$k_2 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} * k_1\right) =$$
$$k_3 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} * k_2\right) =$$
$$k_4 = f(x_0 + h, y_0 + hk_3) =$$
$$y_1 = y_0 + \frac{h}{6} * (k_1 + 2k_2 + 2k_3 + k_4) =$$

$$x_1 = x_1 + h =$$

$$k_1 = f(x_1, y_1) =$$

$$k_2 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{h}{2} * k_1\right) =$$

$$k_3 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{h}{2} * k_2\right) =$$

$$k_4 = f(x_1 + h, y_1 + hk_3) =$$

$$y_2 = y_1 + \frac{h}{6} * (k_1 + 2k_2 + 2k_3 + k_4) =$$

Travail demandé :

Soit l'équation différentielle

$$y = \frac{y}{1 + x^2}$$

avec $y(0) = 1 \;\; et \; x \in [0 \quad 0{,}4]$

Résoudre cette équation avec un pas d'intégration $h = 0{,}2$ en utilisant :

- La méthode d'Euler
- La méthode de Range Kutta d'ordre 4 (RK4)

Calculer la solution exacte de l'équation et comparer avec les approximations précédentes, quel est votre commentaire.

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 5 : Résolution des systèmes d'équations linéaires

Durée du TP : 2 séances de 1h30

But du TP :

Le but de ce TP est l'implémentation de la méthode de triangularisation de Gauss pour la résolution d'un système d'équations linéaires.

Rappel de la méthode:

- On commence par le remplissage de la matrice augmentée A du système qui n'est autre que l'augmentation de la matrice du système d'une colonne contenant le vecteur des données: $A = (A : b)$
  La dimension de la matrice A sera de 'n' lignes et 'n+1' colonnes.

- Ensuite, on programme l'algorithme de Gauss qui va trianguliariser la matrice A. Il procède de la manière suivante :

  Pour $k\ allant\ de\ 1\ à\ \ n-1$

$$ligne\ \ i = \ ligne\ \ i\ -\ \frac{a_{ik}}{a_{kk}}\ \times ligne\ k;\quad avec\ \ \ i\ \ \ allant\ de\ k+1\ \ \ à\ \ \ n$$

- Et enfin, on extrait la solution du système suivant l'algorithme:
-

$$x_i = \frac{a_{i,n+1} - \sum_{i+1}^{n} a_{ij} \times x_j}{a_{ii}}\ ;\qquad i\ allant\ \ de\ \ n\ \ à\ \ 1$$

Travail demandé :

a. Ecrire un script Matlab qui utilise la méthode de Gauss pour trouver la solution du système $A.x = b$ suivant :

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 3 \\ 1 \end{bmatrix}$$

b. Vérifier ensuite votre résultat en le calculant directement avec : $x = inv(A) \times b$

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

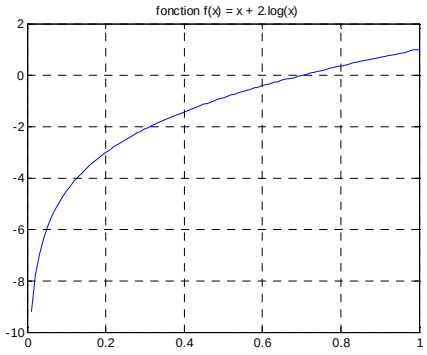
Module : Méthodes Numériques

Annexe 1 : Proposition de solution du TP n°1

```
%    Sachant que le théorème des valeurs intermédiaires est vérifié pour les deux
%   valeurs  '0' et '1' (c.à.d: f(0) * f(1) < 0), nous pouvons tracer la fonction proposée
%   dans cet intervalle comme suit :

clearall, close all, clc
x=[0:0.01:1];
f=inline('x+2*log(x)');
plot(x,f(x));
gridon
title ('fonction f(x) = x + 2.log(x)');
```

Exécution :




```
%  A partir du tracé de la fonction, nous pouvons cadrer la solution :   x* ∈  [0.6 , 0.8]
% La solution de la fonction par la méthode de dichotomie (bipartition):


formatlong
a=0.6;b=0.8;k=0;eps=1.0e-5;
while(abs(b-a)>eps)
   c=(a+b)/2;
   k=k+1;
```

```
if(f(a)*f(c)<0)
b=c;
else
a=c;
end
end
Sol_dichot =c
Nombre_iter=k
```

Exécution :

```
Sol_dichot =    0.703472900390625
Nombre_iter =    15
```

N.B. :
Vous pouvez expliquer à l'étudiant que Matlab dispose d'une fonction prédéfinie de recherche de solution, appelée ***: fzero( )***.

```
%   recherche de solution par la fonction prédéfinie de Matlab: fzero
x0=fzero(f,[0.1 5])
```

Exécution :

```
x0 =     0.703467422498392
% La solution de la fonction par la méthode de Newton-Raphson:

clearall, close all, clc
f=inline('x+2*log(x)');
df=inline('1+2/x');
x0=0.6;
k=0;
eps=1.0e-5;
while(abs(f(x0))>eps)
   x1=x0-(f(x0)/df(x0))
    k=k+1;
    x0=x1;
end
Sol_Newt=x0
Nombre_iter=k
```

Exécution :

```
Sol_Newt =   0.703467422286415
Nombre_iter =     3
```

Proposer le lien suivant aux étudiants désireux faire une initiation rapide à Matlab :

http://perso.mines-albi.fr/~louisnar/ENSEIGNEMENT/MATLAB/polymatlab.pdf

```
% La solution de la fonction par la méthode de Regula-Falsi

clearall, close all, clc
f=inline('x+2*log(x)');
a=0.5;b=1.0;c=a;k=0;eps=1.0e-5;

while(abs(f(c))>eps)
   c=(a*f(b)-b*f(a))/(f(b)-f(a));
if (f(c)*f(a) <0)   b=c;
else       a=c;
end
   k=k+1;
end

Sol_regfal=c
Nombre_iter=k
```

Exécution :

```
Sol_regfal =          0.703468124700322
Nombre_iter =         6
```

```
% La solution de la fonction par la méthode des approximations successives

clearall, close all, clc
g=inline('exp(-x/2)');
x0=0.1;
t=g(x0);
eps=1.0e-5;k=0;

while(abs(t-x0)>eps)
   x1=g(x0);
t=x0;
   x0=x1;
   k=k+1;
end

Sol_mss=x0
Nombre_iter=k
```

Exécution :

Sol_mss = 0.703465010766408
Nombre_iter = 12

Complétez et commentez le tableau suivant :

Méthode	**Solution**	**Itérations**	**Complexité**
Bipartition			
Regula-Falsi			
Newton-Raphson			
Approximations suc.			

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

Annexe 2 : Proposition de solution du TP n°2

Corrigé de la partie A

```
% La méthode d'interpolation de Lagrange
clc, clear all, close all
format long
% l'intervalle
a=3.50;
b=3.70;
% le nombre de sous intervalles n
n=4;
pas=(b-a)/n;
% Génération du vecteur des abscisses des points d'appui  Xi:
x=[a:pas:b];
dim_x=length(x);
% Génération des ordonnées des points d'appui Yi:
for i=1:dim_x
    y(i)= exp(x(i));
end
% Calcul de la taille des deux vecteurs –
n=length(x);
% la valeur à interpoler
t=3.62;
% Boucle de calcul du polynôme de Lagrange
p=0.0;
for i=1:n
% Boucle de calcul des coefficients de Lagrange Li(t)
        L=1;
for j=1:n
           if(i~=j) L=L*(t-x(j))/(x(i)-x(j));
              end
end
      p=p+y(i)*L;
end

% Affichage de la valeur interpolée
P
```

Corrigé de la partie B

```
clear all, close all, clc
format long
% l'intervalle
a=3.50;
b=3.70;
% le nombre de sous intervalles n
n=4;
pas=(b-a)/n;
% Génération du vecteur des abscisses des points d'appui  Xi:
x=[a:pas:b];
dim_x=length(x);
% Génération des ordonnées des points d'appui Yi:
for i=1:dim_x
    y(i)= exp(x(i));
end
% Construction de la matrice des différences finies D:
D=zeros(dim_x,dim_x);
D(:,1)=y';
for j=2:dim_x
   for i=j:dim_x
   D(i,j)=(D(i,j-1)-D(i-1,j-1))/(x(i)-x(i-j+1));
   end
end
m=diag(D)
%Génération de la valeur interpolée :
t=3.62;
s=m(1);
for i=2:dim_x
   p=1;
   for k=1:i-1
    p=p*(t-x(k));
   end
   p
   s=s+p*m(i);
end
s
```

Complétez et commentez le tableau suivant :
Solution exacte = 37.337567822053657

Méthode	**Solution**	**Précision**
Lagrange n=4	37,3375676986358	
Lagrange n=10	37,3375678220536	
Newton n=4	37,3375676986358	
Newton n=10	37,3375678220536	

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

Annexe 3 : Proposition de solution du TP n°3

Partie A

```
clc, clear all, close all
format long

% Méthode du trapèze programmée

f=inline('sin(x).*exp(x)');
a=0;
b=1;
n=10;
h=(b-a)/n;
x=[a:h:b];
y=f(x);

sigma=sum(y(2:n));

It=(h/2)*(y(1)+y(n+1)+2*sigma)

% Utilisation de la fonction prédéfinie Matlab: TRAPZ

Itm=h*trapz(y)

% La valeur exacte

IEx=(sin(1)-cos(1))*exp(1)/2 - (sin(0)-cos(0))*exp(0)/2
```

Partie B

```
% Méthode de Simpson programmée

f=inline('sin(x).*exp(x)');
a=0;
b=1;
n=10;
h=(b-a)/n;
x=[a:h:b];
y=f(x);

sigma1=0;
k=n/2;

for i=1:k
   sigma1=sigma1+y(2*i-1);
end

sigma2=0;

for i=1:k
   sigma2=sigma2+y(2*i);
end

Is=(h/3)*(y(1)+y(n+1)+2*sigma1+4*sigma2)

% Utilisation de la fonction prédéfinie Matlab: QUAD

Ism=quad(f,0,1)
```

Complétez et commentez le tableau suivant :

Solution exacte = 0,909330673631479

Méthode	**Solution**	**Précision**
Trapèze n=10	0,911627886139755	
Trapèze n=100	0,909353640758889	
Simpson n=10	0,909328660197856	
Simpson n=100	0,909330673429412	

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

Annexe 4 : Proposition de solution du TP n°4

- Méthode d'Euler :

$y_1 = y_0 + 0{,}2f(x_0, y_0) = 1{,}200000$
$y_2 = y_1 + 0{,}2f(x_1, y_1) = 1{,}430769$

- Méthode de Range Kutta d'ordre 4

- $x_0 = 0, \quad y_0 = 1$
-

$k_1 = f(x_0, y_0) = 1$
$k_2 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} * k_1\right) = 1.089108910891089$
$k_3 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} * k_2\right) = 1.097931575335752$
$k_4 = f(x_0 + h, y_0 + hk_3) = 1.172679149103029$
$y_1 = y_0 + \frac{h}{6} * (k_1 + 2k_2 + 2k_3 + k_2) = 1.218225337385224$

- $x_1 = 0{,}2\,, \quad y_1 = 1.218225337385224$

$k_1 = f(x_1, y_1) = 1.171370516716561$
$k_2 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{h}{2} * k_1\right) = 1.225103109226495$
$k_3 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{h}{2} * k_2\right) = 1.230032704869608$
$k_4 = f(x_1 + h, y_1 + hk_3) = 1.262268860654435$
$y_2 = y_1 + \frac{h}{6} * (k_1 + 2k_2 + 2k_3 + k_2) = 1.463022370903997$

La solution exacte :

$$\frac{dy}{dx} = \frac{y}{1+x^2}$$

$$\frac{dy}{dx}\frac{dy}{y} = \frac{dx}{1+x^2}$$

$$\int \frac{dy}{y} = \int \frac{dx}{1+x^2}$$

$$\log(y) = \arctan(x) + c$$

Donc $y = C \times e^{\arctan(x)}$

Pour déduire la solution générale de cette équation, on pose $C = C(t)$ et on remplace dans l'équation différentielle :

$$y = C\,e^{\arctan(x)} + C\frac{e^{\arctan(x)}}{1+x^2} = C\frac{e^{\arctan(x)}}{1+x^2}$$

$$C\,e^{\arctan(x)} = 0$$

$$C = 0 \Rightarrow C = cste$$

Alors la condition initiale déterminera la valeur de C

$$y(0) = C e^{\arctan(0)} = C = 1$$

Donc la solution de l'équation est $y(x) = e^{\arctan(x)} \Rightarrow y(0{,}4) = 1.463025244391611$

Et l'on voit bien que RK4 est de loin la méthode qui donne l'approximation la plus proche de la valeur exacte.

A. Méthode d'Euler :

```
clearall, close all, clc
formatlong

% définition en ligne de la fonction f(x,y)
f=inline('y/(1+x^2)');

% [a b] :  L'intervalle d'approximation
a=0 ;b=0.4 ;

% h :     Le pas
h=0.2 ;

% N :    Nombre de valeur à approximer
N=floor((b-a)/h) ;

% x :   vecteur des abscisses d'approximation
x=[a :h :b] ;

% y :     vecteur des ordonnées approximées
y=zeros(1,N) ;y(1)=1 ;

for i=1 :N
y(i+1)=y(i)+h*f(x(i),y(i)) ;
end

x
y
```

Résultat d'Exécution :

```
x =     0                       0.200000000000000       1.400000000000000
y =     1.000000000000000       1.200000000000000       1.430769230769231
```

B. Méthode de Range Kutta d'ordre 4 :

```
clearall, close all, clc
formatlong

% définition en ligne de la fonction f(x,y)
f=inline('y/(1+x^2)');
% [a b]:  L'intervalle d'approximation
a=0;b=0.4;
% h:     Le pas
h=0.2;
 % N :    Nombre de valeur à approximer
N=floor((b-a)/h);
% x:     vecteur des abscisses d'approximation
x=[a:h:b];
% y:     vecteur des ordonnées approximées
y=zeros(1,N);y(1)=1;

for i=1:N
   k1=f(x(i),y(i));
   k2=f(x(i)+h/2,y(i)+k1*h/2);
   k3=f(x(i)+h/2,y(i)+k2*h/2);
   k4=f(x(i)+h,y(i)+k3*h);
y(i+1)=y(i)+(h/6)*(k1+2*k2+2*k3+k4);
end

x
y
```

Résultat d'Exécution :

```
x =  0                    0.200000000000000    0.400000000000000
y =  1.000000000000000    1.218225337385224    1.463022370903997
```

Complétez et commentez le tableau suivant :

Méthode	**Solution**	**Précision**	**Complexité**
Euler			
RK4			

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

Annexe 5 : Proposition de solution du TP n°5

```
clc, clear all, close all
%Nombre de variables et d'équations
n=4;
%Initialisation de la matrice du système 'a(4x4)'
a=[10 7 8 7;7 5 6 5;8 6 10 9;7 5 9 10];
%Initialisation du vecteur des données 'b(4x1)'
b=[4 3 3 1]';
%Formation de la matrice augmentée A(4x5) = (a|b)
A=[a b];

%Algorithme de triangularisation de Gauss
for k=1:n-1
   for i=k+1:n
      A(i,:)=A(i,:)-(A(i,k)/A(k,k))*A(k,:);
   end
end
A
 %Extraction de la solution du système d'équations
for i=n:-1:1
   s=0;
   for j=i+1:n
      s=s+A(i,j)*x(j);
   end
   x(i)=(A(i,n+1)-s)/A(i,i);
end
xgauss=x'
 %Solution du système par calcul direct en utilisant les fonctions Matlab
xm=inv(a)*b
```

Résultat obtenu :

```
A =
  10.0000    7.0000    8.0000    7.0000    4.0000
        0    0.1000    0.4000    0.1000    0.2000
        0         0    2.0000    3.0000   -1.0000
        0         0         0    0.5000   -0.5000

xgauss =
    1.0000
   -1.0000
    1.0000
   -1.0000
xm =
    1.0000
   -1.0000
    1.0000
   -1.0000
```