République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ibn-Khaldoun de Tiaret

Faculté des Sciences Appliquées
Département des Sciences et de la Technologie

BOUMEDIENE Cheikh
Maitre-Assistant

TP de Méthodes Numériques

Recueil des Fiches de Travaux Pratiques (TP)

Domaine: ST Niveau: L2 Semestre: S4 Unité: UEM2.2 Matière: TP-MN

Programmation: GNU-Octave / MATLAB

Equipe Pédagogique: BOUMEDIENE C.
BENSATALLAH A.
BENARIBA B.

Remerciements à: MOKHTARI A.

Références:

[J-D. Bonjour] MATLAB & GNU Octave. https://enacit.epfl.ch/cours/matlab-octave/base.shtml
[Alfio Q. et al.] Calcul Scientifique - Cours et exercices corrigées et illustrations en Matlab et Octave.
Téléchargement de l'interpréteur **Octave** : https://www.gnu.org/software/octave/download.html/

Edition: 04-04-2020
http://moodle.univ-tiaret.dz/

Préambule

Ce présent document est en premier lieu un document à **caractère pédagogique**. Le public ciblé, est principalement, les étudiants du niveau deuxième année de la licence (**L2**) en sciences et technologie (**ST**).

Dans ce niveau, des travaux pratiques (**TP**) de programmation informatique (en Octave, MATLAB ou autre logiciel de calcul scientifique facile d'accès) sont dispensés dans les deux matières Informatique 3 (Semestre 3) puis dans les TP de méthodes numériques (Semestre 4).

Le TP des méthodes numériques (**TP MN**) est un complément des travaux théoriques dispensés dans le cours et les TD de la matière Méthodes Numériques (ou Math5). Le contenu de la matière TP MN a été organisé en cinq (05) volets comme suit:

TP1 : Equations non linéaires (Méthodes: Newton-Raphson – Bisection - Point-fixe)
TP2 : Interpolation et Approximation (Méthodes: Newton - Chebychev)
TP3 : Intégration numérique (Méthodes: Rectangles – Trapèzes - Simpson)
TP4 : Equations Différentielles (Méthodes: Euler - Runge-Kutta)
TP5 : Systèmes d'équations linéaires (Gauss - Jaobi - Gauss-Seidel - Factorisation)

Ce document est un recueil de cinq (**05**) fiches de TP. Chaque fiche de TP est organisée comme suit :

Section 1. Algorithme
Section 2. Programme
Section 3. Expérimentation
Section 4. Travaux Individuels

Après **l'acquisition** des notions théoriques (Cours & TD), l'étudiant est appelé dans chaque TP à:

Travail demandé 1) Dérouler l'algorithme selon l'exemple donné.
Travail demandé 2) Compléter le code et exécuter le programme complété.
Travail demandé 3) Répondre aux questions d'expérimentation du programme.
Travail demandé 5) Préparer les travaux individuels.

L'évaluation du travail de l'étudiant, par l'équipe pédagogique, se portera sur **un compte-rendu individuel** pour **chaque** TP. Ce compte-rendu comprenant ses réponses théoriques et principalement pratiques aux travaux demandés. Les réponses pratiques nécessitent l'installation de l'environnement de programmation, l'édition puis l'exécution des programmes et la correction des erreurs de programmation possibles.

Université: IK-Tiaret - **Faculté:** SA **Département:** ST **- Niveau:** L2 - **Semestre:** S4 **Unité:** UEM 2.2 - **Matière:** TP MN

TP N°1: Equations non Linéaires

1. Algorithme

1.1. Exemple : $f(x) = -0.1x^2 - \cos(x) + 2 = 0$ $Intervalle: [-6,0]$ *ou* $[0,8]$ $Xo = 3$ $E = 0.1$

1.2. Données : Deux variables: X**o** *(X**old** – "pas Xzéro")* et X**n** *(X**new** – Courant).*

1.3. Traitements:

1. *Xo ← VI*
2. *Xn ← Xo*
3. ***tant que*** *(|f(Xn)| >= E)* ***faire***
4. *Xo ← Xn*
5. *Xn ← Xo – f(Xo) / f'(Xo)*
6. ***fintant que***
7. *Xs ← Xn*

2. Programme

2.1. Afin de mettre en œuvre la méthode Newton, compléter le code suivant:

```
clc, clear, format long
function [y]=f(x)    y = …… …… ……; end
function [dy]=df(x) dy = …… …… ……; end
function [ Xs, Ys, K ] = NEWTON(Xo, E)
        K = 0    ,    Xn = Xo    ,   Yn = f(Xn)
        while ( abs(Yn) >= E )
              K  = K + 1
              Xo = Xn
              Xn = Xo – f(Xo)/df(Xo),    Yn = f(Xn)
        end
        Xs = Xn   ,   Ys = f(Xs)
end
```

2.2. Editer puis exécuter le programme avec : `[Xs, Ys, K ] = NEWTON(3 , 0.1)`

3. Expérimentation

3.1. Remplir le tableau en utilisant ce programme:

Cas	*Xo= 3.0*	*Xo= 3.0*	*Xo= -0.19*	*Xo= 3.05*	*Xo= 0*
	E= 10^{-1}	*E= 10^{-9}*	*E= 10^{-5}*	*E= 10^{-9}*	*E= 10^{-5}*
K				∞	*0*
Converg.	*Oui*	*Oui*	*Oui*	*Non*	*Non*
Xs					
Ys		×$10^{-…}$			

3.2. Tracer le graphe de ***f(x)*** sur **[-6,8]** et signaler la solution pour (***Xo= 3.0*** *et* ***E= 10^{-1}***).

4. Travaux Individuels

4.1. Comment visualiser les tangentes ***Yt*** ?

4.2. Pour contrôler les cas de divergence :

- Limiter le nombre des itérations à ***Kmax*** égal à ***100*** *(refaire le cas-4).*
- Ajouter un deuxième un *TA (Test d'arrêt)* sur la dérivée: $|f'(x_k)| < 10^{-3}$ *(refaire le cas-5).*

4.3. Résoudre les équations suivantes:

$x^2\cos(\pi x) = 2x\ln(x) + 1$ $\quad Intervalle: [1,2]$

$x = \sqrt{\alpha}$ $\quad Intervalle: [1,2]$

4.4. Récrire le programme en remplaçant la boucle **`while`** par **`do–until`** puis par **`for`**.

4.5. Ecrire des programmes pour implémenter les méthodes : dichotomie et point-fixe.

Université: IK-Tiaret - **Faculté:** SA **Département:** ST **- Niveau:** L2 - **Semestre:** S4 **Unité:** UEM 2.2 - **Matière:** TP MN

TP N°2: Interpolation et Approximation

1. Algorithme

1.1. Exemple : **X**=[1 3 7 9], **Y**=[10 8 2 6] , **x**=[2 4 5 6 8], **y**=[? ? ? ? ?]

1.2. Données : ***X*** et ***Y*** *(en Majuscules)* deux vecteurs de mêmes tailles des ***(n+1)*** points déjà ***évalués***.
D est la matrice des différences divisées et ***d*** le vecteur des coefficients.
x *(en minuscule)* vecteur des points *à en approximer* le vecteur ***y*** *(en minuscule).*

1.3. Traitements:

1. ***n*** ← *(longueur du vecteur* ***X****) –1 ;*
2. ***pour*** ***i*** *de* ***1*** *à* ***n+1*** *pas* ***1*** ***faire***
3. ***D****(i, 1)* ← ***Y****(i) ;*
4. ***pour*** ***j*** *de* ***1*** *à* ***i–1*** *pas* ***1*** ***faire*** ***D****(i, j+1)* ← **(*D*(i, j) – *D*(i-1, j)) / (*X*(i) – *X*(i–j))** ;
5. ***finpour***
6. ***d*** ← *les éléments diagonaux de* ***D***
7. ***x*** ← *les abscisses des points à évaluer*
8. ***y*** ← ***d****(n+1)*
9. ***pour*** ***k*** *de* ***n*** *à* ***1*** *pas* ***–1*** ***faire*** ***y*** ← ***d****(k)* **+ (x–*X*(k)) * y** ;

1.4. *Dans la ligne 9, on* ***calcule*** *N(x) selon la forme optimale et récursive de Horner (1786-1837):*

$$H(x) = d_0 + (x-x_0)(d_1 + (x-x_1)(d_2 + (x-x_2)(d_3 + \dots + (x-x_{n-2})(d_{n-1} + (x-x_{n-1})\,d_n)))$$

2. Programme

2.1. Afin de mettre en œuvre l'interpolation de Newton, compléter le code suivant:

```
clc, clear, format bank
function [D, d] = COEFNEWTON(X, Y)
    n = length(X)-1 ;
    for i= …… … ……
        D(i,1)= …… ;
        for j= …… … ……
            …… = ( …… - …… )/( …… - …… ) ;
        end
    end
    d = …… ;
end
function y = POLYNEWTON(X, d, x)
    n = length(X)-1 ;    y = d(n+1) ;
    for k= …… … ……
        y = y.*(x - X(k)) + d(k);
    end
end
```

2.2. Exécuter le programme en prenant les X et Y déjà donnés.

3. Expérimentation

3.1. Remplir le tableau en utilisant le programme :

Cas	*Cas-1*	*Cas-2*
Nombre de points : ***n+1***	***3***	***5***
X	*X= [1 3 7]*	*X= [1 3 **5** 7 9]*
Y	*Y= [10 8 2]*	*Y= [10 8 **6** 2 6]*
Polynôme de Newton : ***N(x)***	……	……
Polynôme Standard : ***P(x)***	……	……
x	***x****= [2 4 5 6]*	***x****= [2 4 6 8]*
y	***y****= [ ]*	***y****= [ ]*

4. Travaux Individuels

4.1. Visualiser et comparer les deux cas du tableau précédent avec une **interpolation linéaire**.

4.2. Utiliser un parcours *colonne -**j** / ligne-**i*** pour calculer ***D*** (lignes : 2. 3. 4. de l'algorithme).

4.3. Trouver les a_i de ***P(x)*** en résolvant un système d'équations linéaires *(les inconnus sont les* a_i*)*

4.4. Ecrire un programme pour implémenter l'approximation de ***Tchebychev***.

Université: IK-Tiaret - **Faculté:** SA **Département:** ST **- Niveau:** L2 - **Semestre:** S4 **Unité:** UEM 2.2 - **Matière:** TP MN

TP N°3: Intégration Numérique

1. Algorithme

1.1. Exemple : $In = \int_0^1 f(x)\,dx = \int_0^1 (x^2 + \sin(3\pi x) + 1.46)\,dx = F(1) - F(0)$ $\quad In = 2.0055$

1.2. Données : Les bornes ***a*** et ***b*** de l'intervalle, le nombre des points ***Np*** et le pas ***h***.
Le vecteur des abscisses ***X*** et la surface totale ***Sr***.

1.3. Traitements :

1. ***a*** ← *Début,* ***b*** ← *Fin ,* ***Np*** ← *Nombre de points* % *Données en entrées*
2. ***h*** ← *(b−a)/(Np−1)* % *Calcul de* ***h***
3. ***X*** ← *Vecteur des abscisses* % *Génération de* ***X***
4. ***Sr*** ← *0* % *Initialisation de* ***Sr***
5. ***pour k*** *de* ***1*** *à* ***Np−1*** ***faire***
6. ***Sr*** ← ***Sr +*** *(* ***f****(X(k)) +* ***f****(X(k+1)) * **h**/2)* % *Calcul cumulé des sous-surfaces*
7. ***finpour***

2. Programme

2.1. Afin de mettre en œuvre la méthode des trapèzes, compléter le code suivant:

```
clc, clear, format short
function y = f(x)      y = ..... ... ..... ... ..... ... .....; end
function Y = F(X)      Y = ..... ... ..... ... ..... ... .....; end % F: Primitive
function [Sr,  In,  Er] = TRAPEZES(a,  b,  Np)
    Nt= Np - 1 ,   h= (b-a) / (Np-1)
    x= linspace(a, b, Np) ;  y= f(x) ;
    Sr= 0 ;
    for k= 1 : Np - 1
        Sr = Sr + (   y(k)   +    y(k+1)   )   *   h/2 ;
    end
    In= F(b) - F(a)   ;     Er= In - Sr ;
end
```

2.2. Editer puis exécuter le programme avec : `[Sr, In, Er] = TRAPEZES(0, 1, 6)`

3. Expérimentation

3.1. Remplir le tableau en utilisant ce programme avec ***Np****=21,* ***Np****=2001 et Np=200001:*

Np	***21***	***2001***	***200001***
Nt			
h			
In *(Intégral Analytique)*			
Sr *(Intégral Numérique)*			
Er = In − Sr			

4. Travaux Individuels

4.1. Dans le même programme *(on prend* ***Np=6****)* ajouter les instructions permettant de:

- Afficher à chaque itération: ***k*** *(numéro de l'itération),* ***Srk*** *(Surface k) et* ***Erk*** *(Erreur k).*
- Tracer le graphe de ***f(x),*** de la courbe d'interpolation. Puis dessiner les trapèzes.

4.2. Récrire le programme en remplaçant la boucle ***for*** par ***while*** *(ou* ***do-until****).*

4.3. Récrire le programme en n'utilisant pas la fonction ***F***.

4.4. Refaire le même travail avec $f(x) = e^{\sin(x)}$ et l'intervalle **[0,5]** *avec* ***In=7.1891****.*

4.5. Ecrire un programme pour implémenter la méthode des ***rectangles*** puis celle de ***Simpson***.

TP N°4: Equations Différentielles

1. Algorithme

1.1. Exemple : $y'(t) = t - y(t), \quad y_0 = 1, \quad 0 \le t \le 2 \quad$ ~~$y(t)$~~ $= (y_0 + 1)e^{-t} + t - 1$

1.2. Données: Les bornes ***a*** et ***b*** de l'intervalle, le nombre des points ***Np***, le pas ***h*** et la valeur ***yo***.
Les vecteurs : ***t*** et ***Y***.

1.3. Traitements:

1. ***a*** ← *Début,* ***b*** ← *Fin ,* ***Np*** ← *Nombre de Points,* ***Y(1)*** ← ***yo***
2. ***h*** ← *(b−a)/(Np−1)*
3. ***t*** ← *Vecteur des instants*
4. ***pour i*** *de* ***1*** *à* ***Np−1*** ***faire***
5. ***K1*** ← ***h*** * ***f***(*t(i) , Y(i)*)
6. ***Y****(i+1)* ← ***Y****(i)* + ***K1***
7. ***finpour***

2. Programme

2.1. Afin de mettre en œuvre la méthode d'Euler (RK1), compléter le code suivant:

```
clc, clear, format bank
function dY= f(t, Y)        dY = ..... ..... ..... ..... ..... ;    end
function  y= yan(t, yo)     y = (yo+1)*exp(-t) ... ... ...;    end
function [Y, y, Er] = RUNGEKUTTA(a,  b,  Np,  yo)
    h = (b-a)/(Np-1),  t = linspace(a, b, Np),  Y(1)= yo ;
    for i= 1 : Np-1
        K1 = h * f( t(i) , Y(i) ) ;
        Y(i+1) = Y(i) + K1 ;
    end
    y= yan(t, yo) ; Er= abs(y-Y) ; plot(t,y, t,Y) , grid on
end
```

2.2. Editer puis exécuter le programme avec : `[Y, y, Er]= RUNGEKUTTA(0, 2, 3, 1)`

3. Expérimentation

3.1. Remplir le tableau en utilisant ce programme avec ***Np=5***:

i	*1*	*2*	*3*	*4*	*5*
t	***0.00***	***0.50***	***1.00***	***1.50***	***2.00***
dY					
K1					
Y *(Numérique)*	*1.00*				
y (Analytique)	*1.00*				
Er = | ***Y*** − ***y*** |	*0.00*				

3.2. Discuter les résultats d'exécution *(Valeurs et Graphes)* avec ***Np=3***, ***Np=100*** puis ***Np=1000***.

4. Travaux Individuels

4.1. Refaire le même travail avec : $y'(t) + \alpha\, y(t) = 0, \quad y_0 = 1.5, \quad \alpha \in \{1, 2, 3, 4, 5\}$

4.2. Récrire le programme en n'utilisant pas la fonction ***yan***.

4.3. Réécrire le programme en suivant l'algorithme de *RK4* et comparer les résultats avec *RK1* :

pour ***i*** *de* ***1*** *à* ***Np−1*** ***faire***
K1 ← ***h*** * ***f***(*t(i)* , *Y(i)*)
K2 ← ***h*** * ***f***(*t(i)* + ***h/2*** , *Y(i)* + ***K1/2***)
K3 ← ***h*** * ***f***(*t(i)* + ***h/2*** , *Y(i)* + ***K2/2***)
K4 ← ***h*** * ***f***(*t(i)* + ***h*** , *Y(i)* + ***K3***)

Y*(i+1)* ← ***Y****(i)* + (***K1*** + ***2*K2*** + ***2*K3*** + ***K4***) /6

finpour

Université: IK-Tiaret - **Faculté:** SA **Département:** ST **- Niveau:** L2 - **Semestre:** S4 **Unité:** UEM 2.2 - **Matière:** TP MN

TP N°5: Systèmes d'équations linéaires

1. Algorithme

1.1. Exemple :

*Système Original : **As** X=**Bs***	*Système Réduit : **Ar** X=**B***	*Système Résolu : Ar **X**=B*
$\begin{bmatrix}4 & 8 & 12\\ 2 & 8 & 14\\ 5 & 14 & 13\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}-4\\+6\\+3\end{bmatrix}$	$\begin{bmatrix}1 & 2 & 3\\ 0 & 1 & 2\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}-1\\+2\\+0\end{bmatrix}$	$\begin{bmatrix}1 & 2 & 3\\ 0 & 1 & 2\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}-5\\+2\\+0\end{bmatrix}=\begin{bmatrix}-1\\+2\\+0\end{bmatrix}$

1.2. Données: ***As*** et ***Bs*** les matrices du système original. ***X*** est le vecteur de la solution.

A matrice augmentée et ***B*** second membre du système réduit.

1.3. Traitements:

1. ***n*** ← *nombre de lignes de* ***As*** ;
2. ***A*** ← *[As Bs]* ;
3. ***pour*** ***i*** *de* ***1*** *à* ***n*** ***faire***
4. ***si*** *(A**ii** est nul)* ***alors*** *échange des lignes si possible ;*
5. *Ligne(**i**) ← Ligne(**i**) / A**ii** ;*
6. ***pour*** ***j*** *de* ***i+1*** *à* ***n*** ***faire*** *Ligne(**j**) ← Ligne(**j**) – A**ji** × Ligne(**i**)* ;
7. ***finpour***
8. ***B*** ← *dernière colonne de A ;*
9. ***pour*** ***i*** *de* ***n*** *à* ***1*** *pas* ***-1*** ***faire*** $Xi \leftarrow Bi - \sum_{j=i+1}^{n} Aij \times X(j)$;

2. Programme

2.1. Afin de mettre en œuvre la méthode de ***Gauss***, compléter le code suivant:

```
clc, clear, format bank
function [X, A]= GAUSS(As, Bs)
    n= size(As, 1) ;  A= [As, Bs] ;

    for i= … … …
        if (A(i,i)== 0)   X=[] ; return;  end
        A(i,:) = A(i,:) / A(i,i) ;

        for j= … … …
            A(j,:) =  …… … …… … ……;
        end
    end
    B= A(……,……) ;
    for i= … … …
        S= 0;
        for j= … … …   S= S + A(i,j) * X(j)  ;  end
        X(i) = B(i) - S ;
    end
end
```

3. Travaux Individuels

3.1. Modifier le programme en remplaçant la boucle **`for`** par **`do-until`**.

3.2. Ecrire un programme pour implémenter la méthode itérative de ***Jacobi*** :

1. ***n*** ← *nombre de lignes de* ***A*** ; $X^o \leftarrow$ *Vecteur Initial* ; $X^n \leftarrow X^o$;
2. ***pour*** ***i*** *de* ***1*** *à* ***n*** ***faire*** $X_i^n = \left(B_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{ij} X_j^o \right) / A_{ii}$ *% **X**ⁿ new - **X**ᵒ old*
3. ***Si*** *(* $\max| X^n - X^o | < \varepsilon$ *)* ***Alors*** *Convergence : la solution* $X \leftarrow X^n$ *et Arrêter*
4. ***Sinon*** $X^o \leftarrow X^n$ *et Revenir à la ligne* ***2***

3.3. Ecrire un programme pour implémenter la méthode itérative de ***Gauss-Seidel*** :

1. ***n*** ← *nombre de lignes de* ***A*** ; $X^o \leftarrow$ *Vecteur Initial* ; $X^n \leftarrow X^o$;
2. ***pour*** ***i*** *de* ***1*** *à* ***n*** ***faire*** $X_i^n = \left(B_i - \sum_{j=1}^{i-1} A_{ij} X_j^n - \sum_{j=i+1}^{n} A_{ij} X_j^o \right) / A_{ii}$
3. ***Si*** *(* $\max| X^n - X^o | < \varepsilon$ *)* ***Alors*** *Convergence : la solution* $X \leftarrow X^n$ *et Arrêter*
4. ***Sinon*** $X^o \leftarrow X^n$ *et Revenir à la ligne* ***2***

3.4. Ecrire un programme pour implémenter la méthode ***factorisation LU***.