الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université Akli Mohand Oulhadj - Bouira -
ⵜⴰⵙⴷⴰⵡⵉⵜ ⴰⴽⵍⵉ ⵎⵓⵃⴻⵏⴷ ⵓⵍⵃⴰⵊ - ⵜⵓⴱⵉⵔⴻⵜ -

وزارة التعليم العالي والبحث العلمي
جامعة أكلي محند أولحاج
-البويرة-

Faculté des Sciences et des Sciences Appliquées
كلية العلوم والعلوم التطبيقية

Département de physique
قسم الفيزياء

Polycopie Pedagogique :

Travaux Pratiques - Analyse Numerique

Master physique Theorique, Master Physique des Materiaux et l3-Physique

Analyse Numerique

Dr. LEILA HAMIOUD

Ce polycopie est un support pedagogique de travaux pratiques, de la matiere Analyse Numerique destine aux etudiants Master physique theorique, Master physique des materiaux et Licence physique assurees au departement de physique. Il est redige dans le cadre de l'implementation sous l'ordinateur de certaines methodes de resolution numerique etudiees dans les differents chapitres du cours. Ce polycopie est un outil pratique aidant les etudiants a maitriser leurs connaissances en utilisant un logiciel scientifique. Donc il est adresse a tous les etudiants suivant un cursus universitaire de type scientifique et qui ont besoin de stimuler leurs connaissances operationnelles.

Nous avons presente quelques methodes numeriques couramment utilisees pour un calcul scientifique : l'approximation des fonctions par interpolation polynomiale, resolution des systemes lineaires, des equations non lineaires et des equations differentielles. a l'aide du matlab, nous avons analyse les proprietes theoriques des methodes numeriques, ainsi nous avons illustre quelques avantages et inconvenients de ces methodes a l'aide d'exemples. Ceci nous a permis d'expliquer, de visualiser et de comparer nos resultats.

Années 2018-2021

Avant-propos

Ce polycopié est un support pédagogique de travaux pratiques, de la matière Analyse Numérique destiné aux étudiants Master physique théorique, Master physique des matériaux et Licence physique assurées au département de physique. Il est rédigé dans le cadre de l'implémentation sous l'ordinateur de certaines méthodes de résolution numérique étudiées dans les différents chapitres du cours. Ce polycopié est un outil pratique aidant les étudiants à maitriser leurs connaissances en utilisant un logiciel scientifique. Donc il est adressé à tous les étudiants suivant un cursus universitaire de type scientifique et qui ont besoin de stimuler leurs connaissances opérationnelles.

Nous avons présenté quelques méthodes numériques couramment utilisées pour un calcul scientifique : l'approximation des fonctions par interpolation polynomiale, résolution des systèmes linéaires, des équations non linéaires et des équations différentielles. A l'aide du Matlab, nous avons analysé les propriétés théoriques des méthodes numériques, ainsi nous avons illustré quelques avantages et inconvénients de ces méthodes à l'aide d'exemples. Ceci nous a permis d'expliquer, de visualiser et de comparer nos résultats.

Tables des Matières

TP Nº 1 - Représentation graphique des résultats sous Matlab

1. But du TP

Manipuler et maîtriser la représentation graphique des résultats des calculs scientifiques.

2. Rappel sur quelques commandes les plus utilisées

Matlab peut représenter des graphiques d'une façon simple et puissante. Les fonctions de représentation plus employées sont **plot**, **plot3**, **mesh**, et **surf**. Voici une introduction rapide aux commandes de représentation graphique.

2.1 Graphiques 2D

La commande $plot$ génère des graphiques $x - y$ linéaires ; si x et y sont des vecteurs (ou des matrices) de même dimension alors $\boldsymbol{plot}(x, y)$ ouvre une fenêtre graphique et dessine les valeurs de y en ordonnées et celles de x en abscisses, par exemple :

```
x = -4:0.1:4;
y = sin(x);
plot(x,y)
```

>> Essayez cet exemple. L'exécution de ce code nous a donné la figure (1)

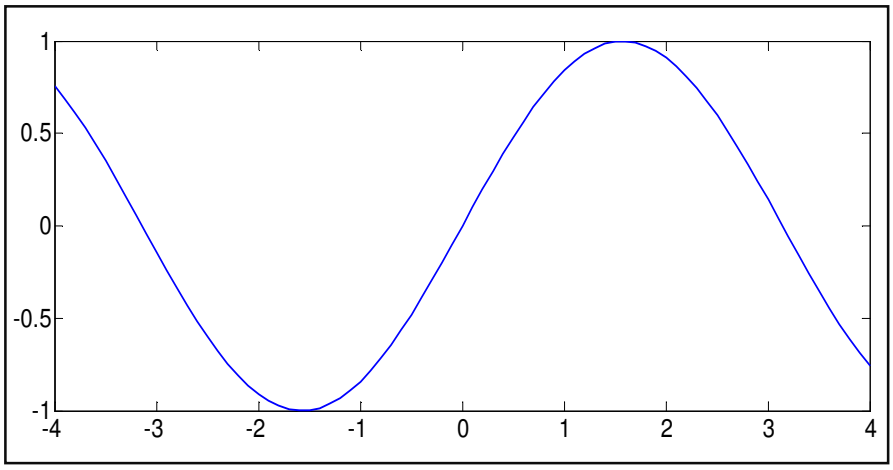


Fig.1 –La courbe de la fonction $sin(x)$

On peut avoir plusieurs figures au même temps. Par exemple, si l'on a déjà une figure dans une fenêtre, on peut ouvrir une deuxième fenêtre pour une nouvelle figure avec la commande ***figure*.** Ainsi les figures sont numérotées en ordre croissant. Pour revenir sur une figure quelconque il suffit de faire ***figure*** (*numéro de figure*).

A l'aide des commandes Matlab on peut rendre les figures plus présentables comme par exemple : ajouter un titre ou un texte dans n'importe quelle position, changer les couleurs des courbes ou bien la forme du point de coordonnées, modifier le type de lignes reliant les points….

Les commandes Matlab suivantes sont les plus utilisées pour la gestion des figures :

title titre de la figure,
xlabel titre de l'axe des abscisses
ylabel titre de l'axe des ordonnées
text insère du texte à certaines coordonnées.

La commande ***grid on*** dessine une grille dans la figure actuelle.

Les axes sont calculés automatiquement. La commande ***axis*** permet d'éviter cette configuration automatique.
Il y a deux façons de représenter deux courbes sur la même figure. Par l'exemple :

```
x=0:.01:2*pi;
y1=sin(x);
y2=sin(2*x);
y3=sin(4*x);
plot(x,y1,x,y2,x,y3)
```

>> Essayez cet exemple pour pouvoir apprécier les différences.

La deuxième façon consiste à utiliser la commande ***hold***. La commande ***hold on***, retient le contenu de la fenêtre graphique de façon à pouvoir superposer une nouvelle courbe sur la même fenêtre.
On peut changer les types des lignes et des points et ainsi leurs couleurs. En ajoutant un nouvel argument à la fonction plot par exemple :

```
plot(x,y1,'b--',x,y2,'r:',x,y3,'g+').
```

Pour plus des commandes pour des caractères spéciaux, types de marqueurs, types de lignes et ainsi leurs couleurs sont disponibles dans les références [3] et [4].

```
clear all ;close all ; clc
x=0:.01:2*pi;
y1=sin(x);
y2=sin(2*x);
y3=sin(4*x);
plot(x,y1,'b--',x,y2,'r:',x,y3,'g+')
```

```
grid on
xlabel('X')
ylabel('y1,y2 et y3')
hold on
legend('y1=sin(x)','y2=sin2x', 'y3=sin(4x)')
hold on
title(' les fonction sin(x)sin(2x)et sin(4x)')
```

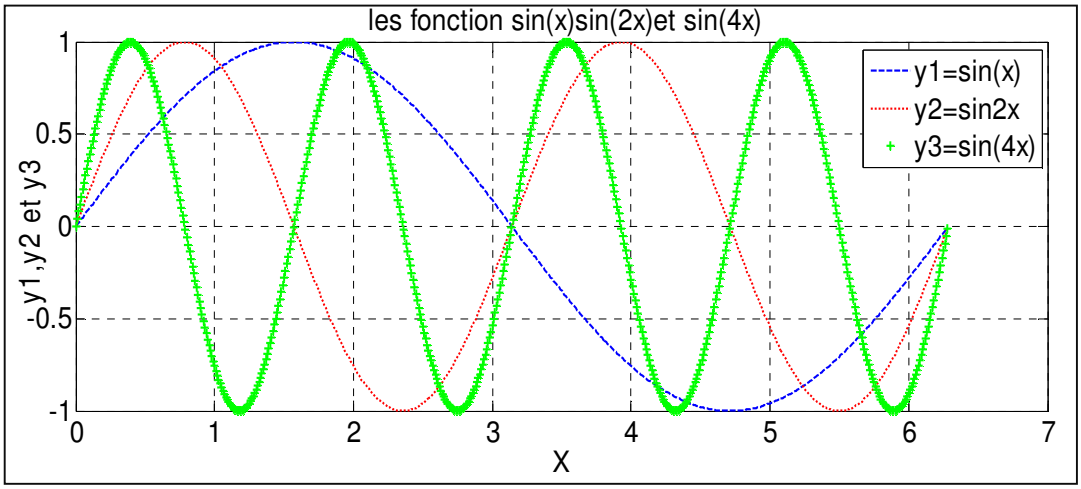


Fig. 2

La deuxième façon pour tracer les courbes $y1$ $y2$ et $y3$ dans la même figure (comme le montre la figure (2)) est effectuée via le code suivant :

```
clear all ;close all ; clc
x=0:.01:2*pi;
y1=sin(x);
plot(x,y1,'b--')
grid on
hold on
y2=sin(2*x);
plot(x,y2,'r:') ;
hold on
y3=sin(4*x);
plot(x,y3,'g+') ;
xlabel('X')
ylabel('y1,y2 et y3')
hold on
legend('y1=sin(x)','y2=sin2x', 'y3=sin(4x)')
hold on
title(' les fonction sin(x)sin(2x)et sin(4x)')
```

La commande ***subplot*** peut être utilisée pour diviser l'écran en plusieurs morceaux. Voir *help subplot*.

Plusieurs autres commandes, telles que ***semilogx(), semilogy(), loglog(), escaliers (), tige(), barre ()/barh()*** et ***hist(),*** peuvent être utilisées pour dessiner divers graphiques (**Fig. 3** et **Fig.4**).

>> Essayez cet exemple :

```
th = [0: .02:1]*pi;
subplot(221), polar(th,exp(-th))
subplot(222), semilogx(exp(th))
subplot(223), semilogy(exp(th))
subplot(224), loglog(exp(th))
pause, clf
subplot(221), stairs([1 3 2 0])
subplot(222), stem([1 3 2 0])
subplot(223), bar([2 3; 4 5])
subplot(224), barh([2 3; 4 5])
pause, clf
y = [0.3 0.9 1.6 2.7 3 2.4];
subplot(221), hist(y,3)
subplot(222), hist(y,0.5 + [0 1 2])
```

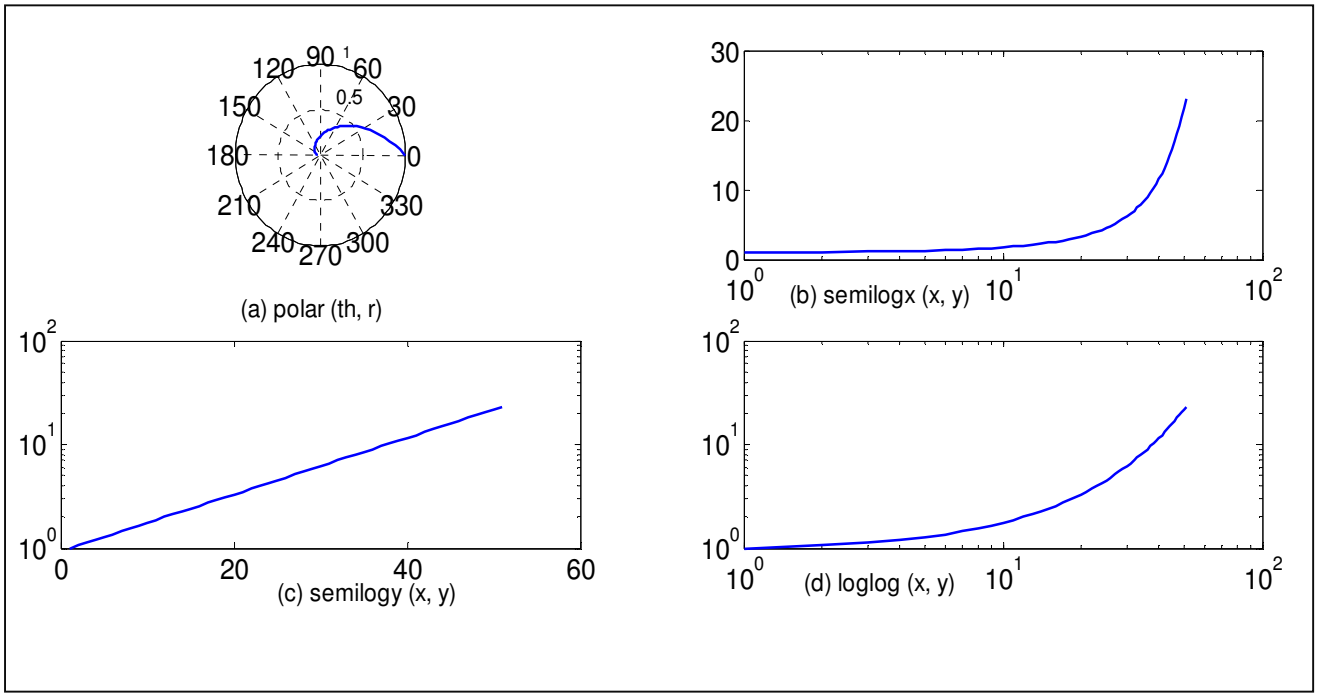


Fig. 3 - Graphiques tracés par différentes commandes graphiques ***2D***.

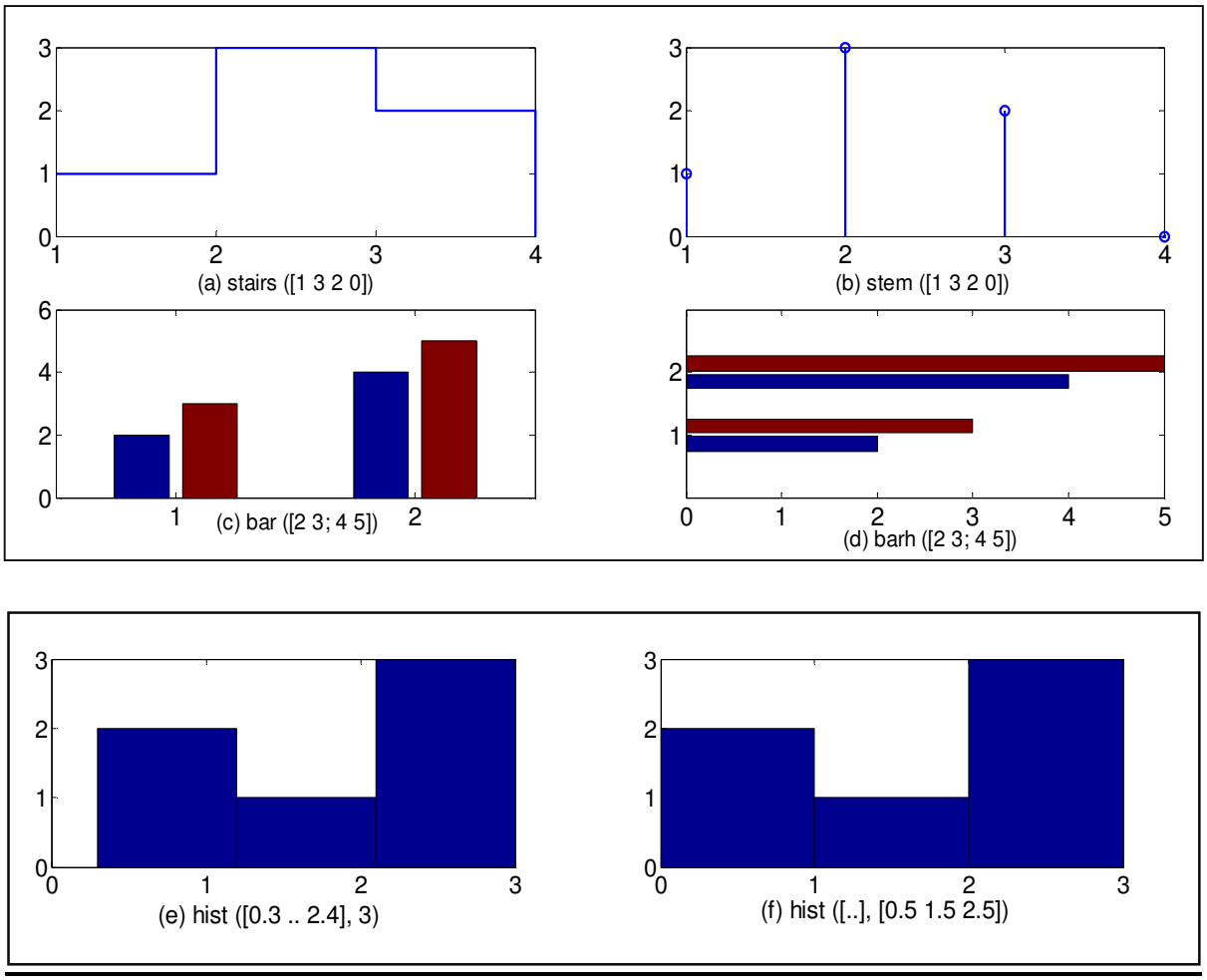


Fig.4 - Graphiques tracés par différentes commandes graphiques ***2D***

2.2 Graphiques 3D

La commande ***plot*** en deux dimensions a une version ***3D*** appelée ***plot3*** qui produit des courbes en 3 dimensions. Si $x, y,$ et z sont trois vecteurs de la même taille, alors la commande ***plot3(x,y,z)*** produit une figure en ***3D*** en perspective avec une courbe qui passe par les points $x, y,$ et z.

Matlab a plusieurs autres commandes graphiques ***3D*** telles que, ***maille(),*** et le ***contour().*** ***parcelle3()*** trace une fonction de valeur ***2D*** d'une variable évaluée par scalaire; ***maillage()/contour()*** trace une fonction scalaire d'une variable ***2D*** dans un style ***maille/contour***, respectivement (**Fig. 5 et Fig. 6**).

>> Essayez cet exemple :

```
t = 0:pi/50:6*pi;
expt = exp(-0.1*t);
xt = expt.*cos(t); yt = expt.*sin(t);
%%dividing the screen into 2 x 2 sections
```

```
subplot(221), plot3(xt, yt, t), grid on %%helix
subplot(222), plot3(xt, yt, t), grid on, view([0 0 1])
subplot(223), plot3(t, xt, yt), grid on, view([1 -3 1])
subplot(224), plot3(t, yt, xt), grid on, view([0 -3 0])
pause, clf
x = -2:.1:2; y = -2:.1:2;
[X,Y] = meshgrid(x,y); Z = X.^2 + Y.^2;
subplot(221), mesh(X,Y,Z), grid on %%[azimuth,elevation] = [-
37.5,30]
subplot(222), mesh(X,Y,Z), view([0,20]), grid on
pause, view([30,30])
subplot(223), contour(X,Y,Z)
subplot(224), contour(X,Y,Z,[.5,2,4.5])
```

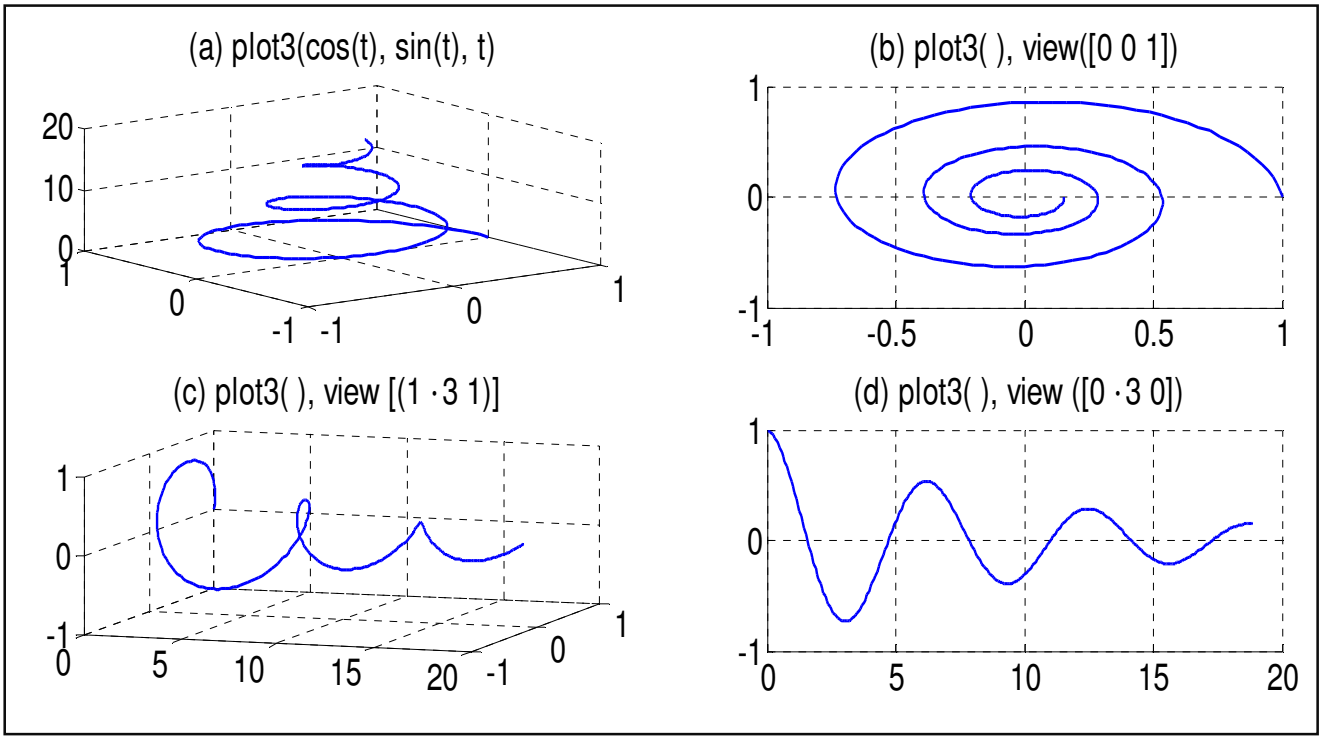


Fig. 5 - Graphiques dessinés par la commande ***plot3()*** avec des vues différentes.

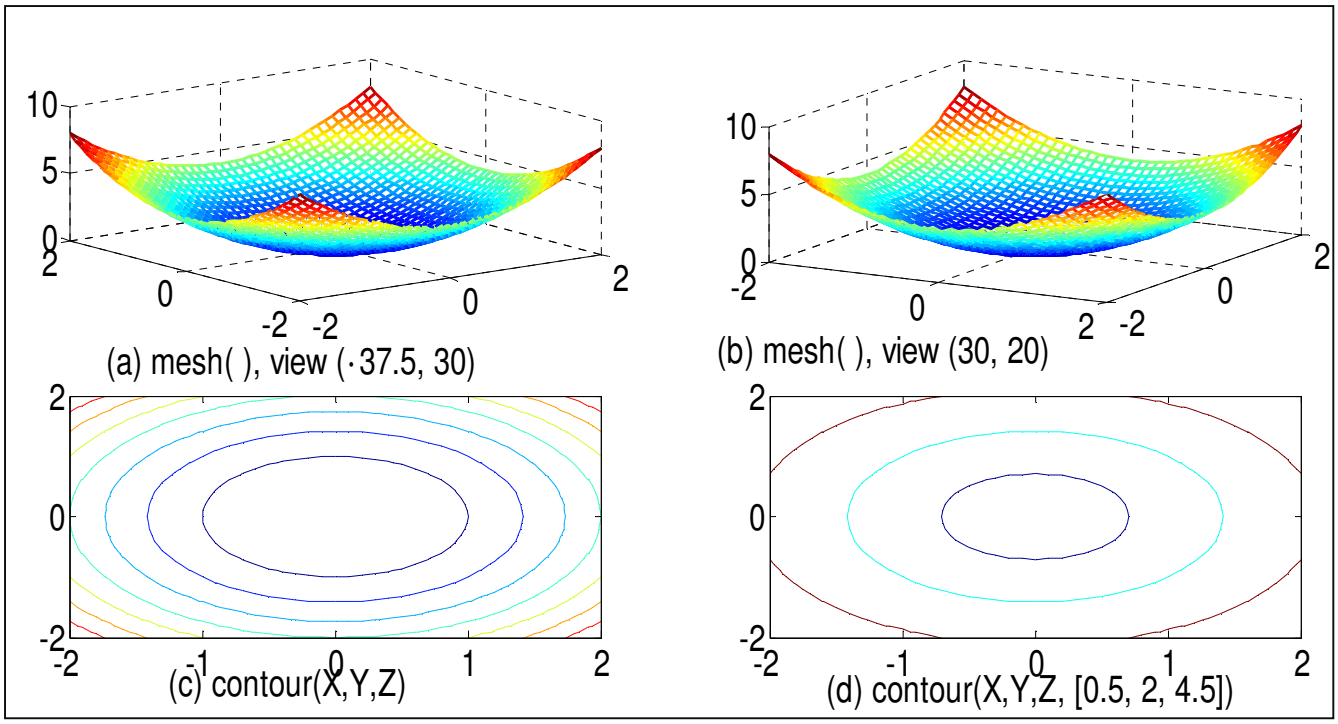


Fig.6 - Graphiques dessinés par les commandes de ***maille ()*** et de ***contour()***

TP N° 2 Résolution des équations non linéaires

1. But du TP

L'objet de ce TP est l'approximation des racines d'une fonction réelle d'une variable réelle, c'est-à-dire la résolution approchée du problème suivant : $\alpha \in R$ telles que $f(\alpha) = 0$.

Il existe plusieurs méthodes numériques telles que la dichotomie, point fixe et Newton, permettant de calculer seulement une seule racine sur un intervalle bien choisi. La majorité de ces méthodes sont itératives, distinguent principalement par leurs vitesses de convergence.

La résolution de l'équation $f(x) = 0$ consistent à :

- localiser grossièrement le zéro de *f* en utilisant le plus souvent la méthode graphique.
- construire à partir d'une valeur initiale x_0 une suite x_n convergente vers la solution exacte telle que $\lim_{n\to\infty} x_n = \alpha$ où α vérifié $f(\alpha) = 0$.

Si l'équation $f(x) = 0$, possède plus d'une racine, il est nécessaire de les localiser dans des intervalles bien choisis et de faire le calcul pour chaque racine à part.

2. Localisation des solutions

Supposant qu'on a confirmé l'existence d'une unique racine dans l'intervalle $[a, b]$ de R (moyen d'une étude graphique, une raison physique ou un théorème). La clé de la recherche des racines d'équations repose sur l'existence d'un encadrement préalable de cette racine.

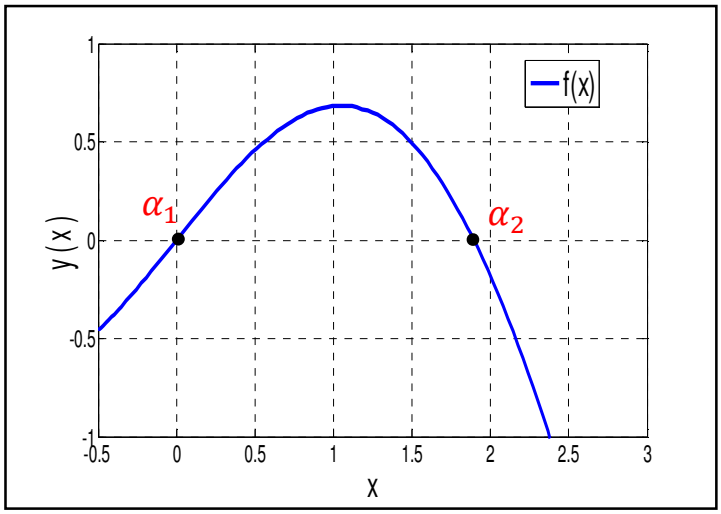


Fig. 1 : Localisation des racines

D'après le théorème des valeurs intermédiaires : si le produit $f(a)f(b) < 0$ et si la fonction f est continue, f doit s'annuler au moins une fois à l'intérieur de cet intervalle.

La figure 1 montre le graphe d'une fonction *f* qui coupe l'axe des *x* en deux points, c'est-à-dire que l'équation $f(x) = 0$ possède deux racines, α_1 et α_2.

Dans le cas où la fonction $f(x)$ est compliquée, on peut décomposer l'équation $f(x) = 0$ (s'il est possible) en deux parties simples $g(x) = h(x)$. Par exemple l'équation $f(x) = exp(x) - 4x$, est peut être décomposée en : *g(x)=h(x)* avec $g(x) = exp(x)$ et $h(x) = 4x = exp(x)$.

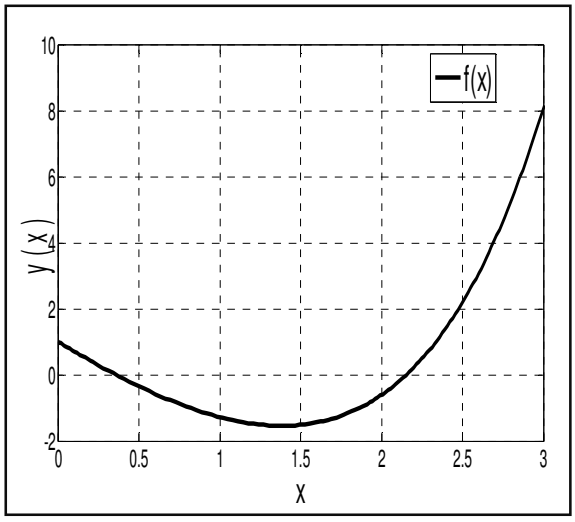


Fig. 2 – Graphe de fonction *f*

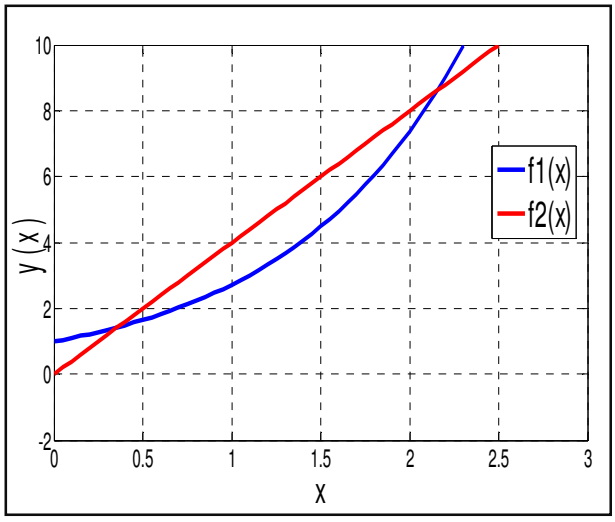


Fig. 3 – Graphe des fonctions f_1 et f_2

On note que cette équation a deux racines (figure 2) qui appartiennent par exemple aux intervalles [0 0.5] et [2 2.5]. On peut aussi réécrire la fonction $f(x) = 0$ sous une forme plus simple. Par exemple : $exp(x) - 4x = 0 \Leftrightarrow exp(x) = 4x$ ou bien $f_1(x) = f_2(x)$; avec $f_1(x) = exp(x)$ et $f_2(x) = 4x$. Ensuite on trace ces deux fonctions (figure 3), qui sont faciles sur les même axes, leurs intersections représentent les racines de l'équation $f(x) = 0$.

Remarque

Les systèmes non linéaires, qu'on va étudier dans ce chapitre, doivent être considérés unidimensionnels pour lesquels plusieurs algorithmes sont mises en point, afin de choisir l'algorithme le mieux adapté, compte tenu des informations que l'on dispose sur la fonction f.

3. Exemples de motivation

3.1. L'équation d'équilibre d'un pendule simple soumis à l'énergie potentielle de gravitation et l'´energie potentielle électrostatique

Un pendule simple de masse m porte à son extrémité une charge q. On place à proximité une autre charge q' (**Fig. 4**). On veut trouver l'angle d'´equilibre θ du pendule en fonction de la valeur de q'.

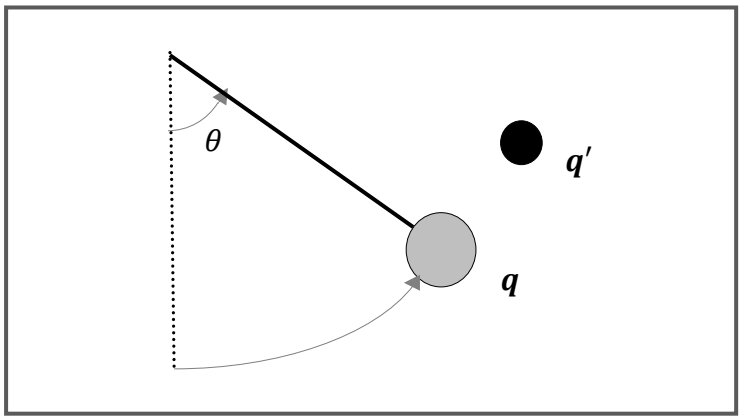


Fig. 4 – Le pendule chargé

L'énergie potentielle de gravitation du pendule est :

$$V_g = mgz = -mg\ell \cos\theta$$

où ℓ est la longueur du pendule et θ l'angle qu'il fait avec la verticale.

L'´energie potentielle électrostatique des charges est :

$$V_e = \frac{1}{4\pi\varepsilon_0} \frac{qq'}{(\ell \sin\theta - x_q)^2 + (\ell(1 - \cos\theta) - y_q)^2}$$

où x_q et y_q sont les coordonnées de la charge q'. L'´equilibre est donné par :

$$\frac{d(Vg + Ve)}{d\theta} = 0$$

D'après un calcul simple, on obtient:

$$\sin\theta + \frac{qq'}{4\pi\varepsilon_0 mg} \frac{x_q \cos\theta + (yq - \ell)\sin\theta}{[(\ell \sin\theta - x_q)^2 + (\ell(1 - \cos\theta) - y_q)^2]^{\frac{3}{2}}} = 0$$

La résolution analytique de ce problème n'est pas évidente. Il s'agit de trouver numériquement les racines de la fonction :

$$f(\theta) = \sin\theta + \frac{qq'}{4\pi\varepsilon_0 mg} \frac{x_q \cos\theta + (y_q - \ell)\sin\theta}{[(\ell \sin\theta - x_q)^2 + (\ell(1 - \cos\theta) - y_q)^2]^{\frac{3}{2}}}$$

3.2. L'équation d'´etat d'un gaz

On veut déterminer le volume V occupé par un gaz de température T et de pression p. L'´equation d'´etat (c'est-`a-dire l'´equation qui lie p, V et T) est :

$$\left[p + a\left(\frac{N}{V}\right)^2\right](V - Nb) = kNT\,,$$

où a et b sont deux coefficients dépendants de la nature du gaz, N est le nombre de molécules contenues dans le volume V et k est la constante de Boltzmann. Il faut donc résoudre l'équation non linéaire d'inconnu V.

$$\left[p + a\left(\frac{N}{V}\right)^2\right](V - Nb) - kNT = 0$$

4. Les méthodes les plus utilisées pour résoudre l'équation $f(x) = 0$

4. 1. Algorithme de Newton

Le principe de la méthode de Newton s'agit, à partir d'un point de départ x_0 judicieusement choisi, d'approcher la fonction par sa tangente et de chercher l'intersection de celle-ci avec l'axe des x (figure 5). Si l'on part d'un point d'abscisse x_0, on doit pouvoir calculer $f(x_0)$ et la pente de la tangente $f'(x_0)$. L'intersection de la tangente avec l'axe des x est donnée par :

$$x_1 = x_0 - \frac{f(x0)}{f\prime(x0)}$$

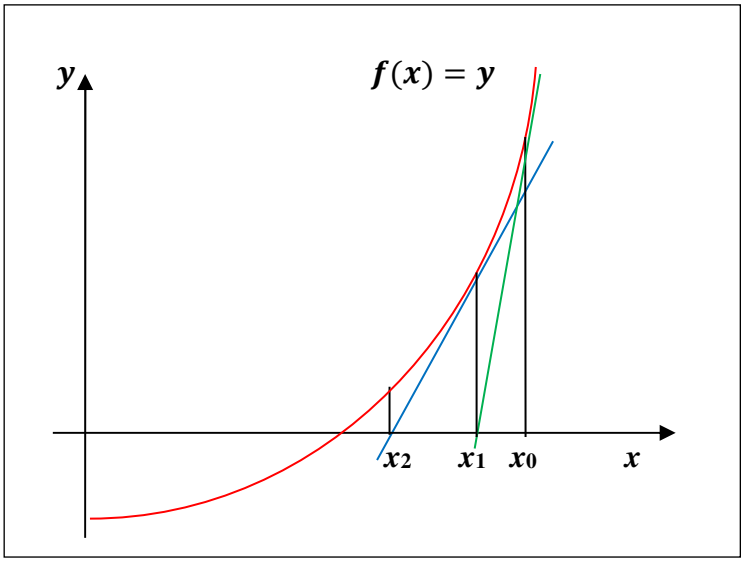


Fig. 5 – Principe de la méthode de Newton

Il suffit de répéter l'opération pour x_1, x_2,… jusqu'à ce que le critère de convergence que l'on aura choisi soit vérifié.

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

On remarque que $f'(x)$ doit être non nulle.

Convergence de la méthode de Newton

Soit une fonction $\boldsymbol{f}$ définie sur $[a, b]$ telle que :

- $f(a)f(b) < 0$.
- $f'(x)\ et\ f''(x)$ sont non nulles et gardent un signe constant sur l'intervalle donné.

4. 2. Algorithme de dichotomie ou bissection

Cette méthode est appliquée seulement dans le cas où nous connaissons un certain intervalle $[a\ , b]$ sur lequel $f\ (x)$ est continu et la solution racine α existe de façon unique et, plus important encore, $f\ (a)$ et $f\ (b)$ ont les signes opposés (théorème des valeurs intermédiaires). L'algorithme de la méthode de dichotomie est le suivant :

- on calcule $f\ (x_0)$ avec $x_0 = (a + b)/2$,
- si $f(x_0) = 0$ alors x_0 est le zéro cherché.
- si $f(x_0) \neq 0$:

* soit $f(x_0)f(a) > 0$ et alors le zéro $\alpha \in (x_0, b)$ et on définit $a = x_0$ et $x_1 = (a + b)/2$ pour ce nouveau a.

* soit $f(x_0)f(a) < 0$ et alors $\alpha \in (a, x_0)$ et on pose $b = x_0$ et $x_0 = (a + b)/2$ pour ce nouveau b.

En réitérant ce processus de division de l'intervalle jusqu'à la convergence et l'obtention de la précision souhaitée (tolérance considérée). Ainsi, pour la $n^{ième}$ itération, on divise : $[a_n\ b_n]$ en $[a_n\ c_n]$ et $[c_n\ b_n]$, avec à chaque fois :

$$c_n = (a_n + b_n)/2$$

En construisant la suite $x_0, x_1 \ldots, x_n$ qui vérifie pour tout n :

$$|x_n - \alpha| \leq (b - a)/2^{n+1}$$

5. Critère d'arrêt

Si la condition de convergence est vérifiée, le procédé itératif doit converger. Donc chaque nouvelle itération est meilleure que la précédente, de ce fait, si on a une précision ε, on arrête

le calcul lorsque la différence absolue entre deux approximations successives est inférieure à la précision donnée, c'est-à-dire :

$$|\boldsymbol{x_{n+1}} - \boldsymbol{x_n}| \leq \boldsymbol{\varepsilon}, \qquad \boldsymbol{n} = 1, 2, 3 \ldots$$

Si cette condition est vérifiée on prend x_{n+1}comme solution de $f(x) = 0$.

On peut utiliser le critère d'arrêt : $|\boldsymbol{f}(\boldsymbol{x_n})| \leq \boldsymbol{\varepsilon}$.

6. L'ordre de convergence et la vitesse de convergence

Le procédé est dit convergent si la suite x_n tend vers un nombre fini lorsque n tend vers l'infini : $\lim_{n\to\infty} x_n = \alpha$.

Soit p un entier positif. On dit qu'une méthode convergente est d'ordre p s'il existe une constante C telle que, si $\lim_{n\to\infty} x_n = \alpha$

$$|\alpha - x_{n+1}| \leq C|\alpha - x_n|^2$$

$$\lim_{n\to\infty} \frac{|\alpha - x_{n+1}|}{|\alpha - x_n|^2} = C$$

Si $p = 1$, on parle de convergence linéaire, si $p = 2$, on parle de convergence quadratique. Dans le cas où $p = 1$, il est nécessaire que $C < 1$ pour que x_n converge vers la racine α.

L'ordre de la méthode Newton est quadratique ($p = 2$), étant donné que l'ampleur de l'erreur d'approximation est proportionnelle au carré de l'erreur d'approximation précédente.

Pour un nombre n suffisamment élevé, la vitesse de convergence d'une méthode itérative est donnée par :

$$v_p(x, n) = \frac{x_{n+2} - x_{n+1}}{(x_{n+1} - x_n)^p}$$

- La méthode de Newton a plusieurs avantages par rapport à la méthode de la dichotomie. Elle est en général beaucoup plus rapide. De plus, le très gros avantage de cette méthode est de pouvoir se généraliser aisément à plusieurs dimensions.
- La convergence de la méthode de bissection n'est pas la meilleure par rapport aux autres méthodes. Cette méthode, nécessite une connaissance préalable de la fonction puisqu'elle suppose qu'il y ait un et un seul zéro dans l'intervalle de recherche : il faut donc pouvoir choisir celui-ci judicieusement.

Durant ce TP, nous allons mettre en œuvre les algorithmes de résolution des équations non linéaires : de Newton et de la bissection, et en déduisant à la fin l'algorithme le mieux adapté, en considérant la vitesse de convergence de chaque méthode utilisée.

7. Énoncé du TP

Partie (1) :

On veut résoudre l'équation : $f(x) = \frac{x}{2} - sin(x) + \frac{\pi}{6} - \frac{\sqrt{3}}{2} = 0$.

1. Tracer les graphes de f et de la dérivée f' pour $-\frac{\pi}{2} \leq x \leq \pi$ dans une même figure.
2. Le critère d'arrêt est : $|x_{n+1} - x_n| < \varepsilon$, x_n étant la solution approchée et ε, la tolérance considérée ($\varepsilon = 10^{-6}$).

Ecrire un code Matlab qui permet de calculer la solution approchée x_n située à l'intervalle $\left[\frac{\pi}{2}, \pi\right]$ en affichant la solution approchée x_n.

3. Afficher le nombre d'itérations conduisant à cette solution.
4. Refaire la 2ème question en utilisant l'algorithme de dichotomie en représentant le graphe de f et la solution approchée dans la même figure. Commenter les résultats.
5. Calculer l'ordre et la vitesse de convergence de chaque méthode.

Partie (2) :

On cherche le volume occupé par 500 molécule de dioxyde de carbone (CO_2) à une température de $T = 400\ K$ et à une pression de $P = 4.10^7\ Pa$ sachant que la constante de *Boltezman* est $K = 1.3806503\ 10^{-23}\ Joule\ K^{-1}$ et l'équation d'état d'un gaz est donnée comme suit : $\left[P + a\left(\frac{N}{V}\right)^2\right](V - Nb) = KNT$

Avec a et b sont deux coefficients dépendants de la nature du gaz : $a = 0{,}401\ Pa\ m^6$ et $b = 42.7\ 10^{-6}\ m^3$ et N le nombre des molécules.

- Résoudre numériquement le problème en utilisant la méthode graphique puis la méthode algorithmique de Newton et de dichotomie avec une précision (tolérance $e = 10^{-8}$). Comparer les résultats.

8. Solution

Partie (1)

```
1/ close all; clear all; clc
   x=[-pi/2:pi/10:pi]
   f=x/2-sin(x)+pi/6-sqrt(3)/2; df=1/2-cos(x);
   plot(x,f,'b','MarkerSize',2,'LineWidth',2)
   grid on
```

```
    hold on
    plot(x,df,'r','MarkerSize',2,'LineWidth',2)
    xlabel('x')
    ylabel('y(x)')
    legend('f(x)','df(x)')
    title('Graphes des fonctions f(x)et df(x)/dx dans
l'intervalle [-pi/2, pi]');
```

D'après la figure (**Fig. 6**), nous remarquons que l'équation, $f(x) = \frac{x}{2} - sin(x) + \frac{\pi}{6} - \frac{\sqrt{3}}{2} = 0$, possède 2 racines appartiennent aux intervalles $\left[-\frac{\pi}{2}, 0\right]$ et $\left[\frac{\pi}{2}, \pi\right]$.

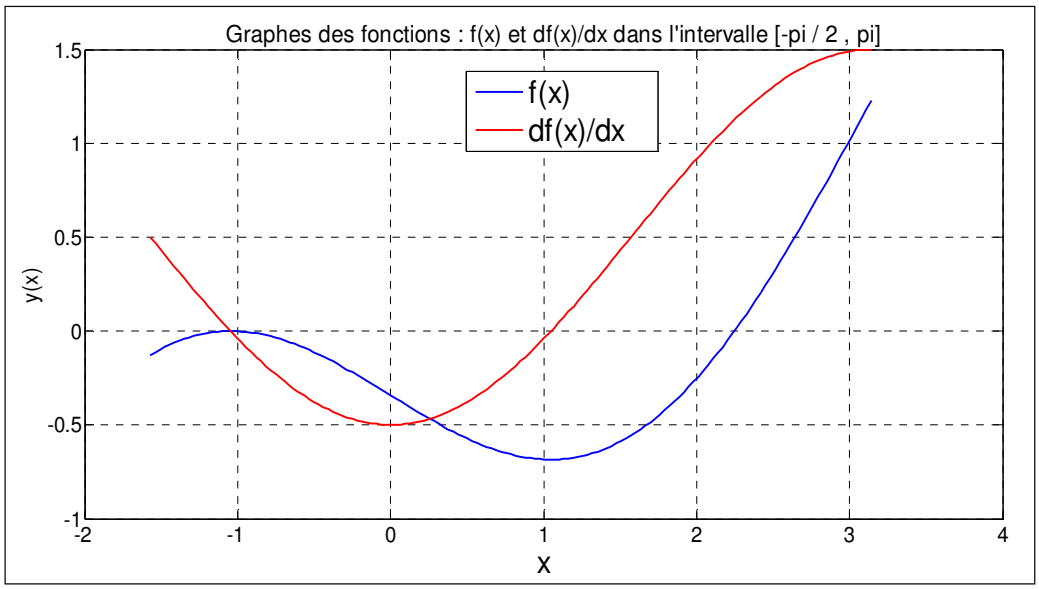


Fig. 6

2/ Pour calculer la solution approchée x_n de l'équation $f(x) = \frac{x}{2} - sin(x) + \frac{\pi}{6} - \frac{\sqrt{3}}{2} = 0$, dans l'intervalle $\left[\frac{\pi}{2}, \pi\right]$**.** On applique l'algorithme de Newton et pour avoir une convergence rapide, nous exécutons le programme suivant en choisissant la valeur initial $x_0 = 2.2300$ ($f(2.2300)$ est proche de zéro). On peut utiliser la commande $[x, y, butkey] = ginput$ qui nous a permet d'obtenir les valeurs x_0 et $f(x_0) = y \approx 0$) :

```
close all; clear all; clc
f=@(x)x/2-sin(x)+pi/6-sqrt(3)/2; df=@(x)1/2-cos(x);
e=10^-6;iter=0;x0=input('donner x0');xi=0;
while abs(xi-x0)>e
    x1=x0-f(x0)/df(x0);
    xi=x0
    x0=x1;
    iter=iter+1
```

```
end
x1
```

Les résultats de calcul sont affichés comme suit :

iter = 4

x1 = 2.2460

4/ Pour calculer le zéro de f en utilisant la méthode dichotomie pour la même tolérance ($e = 10^{-6}$) en exécutant le code suivant :

```
close all; clear all; clc
a=pi/2; b=pi ; c=(a+b)/2; tol=10^-6; iter=0;
while abs(c/2-sin(c)+pi/6-sqrt(3)/2) > tol
if (a/2-sin(a)+pi/6-sqrt(3)/2)*(c/2-sin(c)+pi/6-sqrt(3)/2) <0
b=c;
end
if (c/2-sin(c)+pi/6-sqrt(3)/2)*(b/2-sin(b)+pi/6-sqrt(3)/2)<0
a=c;
end
c=(a+b)/2; iter=iter+1;
f=@(x)(x/2)-sin(x)+(pi/6)-sqrt(3)/2
fplot(f,[pi/2 pi])
grid on
hold on
end
c
iter
x=pi/2:(a-b)/10:pi;
f=@(x)(x/2)-sin(x)+(pi/6)-sqrt(3)/2
plot(c,f(c),'ro','MarkerSize',12,'LineWidth',2)
plot(c,f(c),'rx','MarkerSize',12,'LineWidth',2)
xlabel('x')
ylabel('y(x)')
title('Racine approchée de f par la méthode de dichotomie');
text('Interpreter','latex','String','$f=(x/2)-sin(x)+(pi/6)-
sqrt(3)/2$','Position',[1.8 1.2],'FontSize',14);
text(c,2*c,['c= ',num2str(c)],'Position',[2.3 -0.1]);
```

On obtient les résultats qui sont affichés comme suit :

c = 2.2460

iter = 17

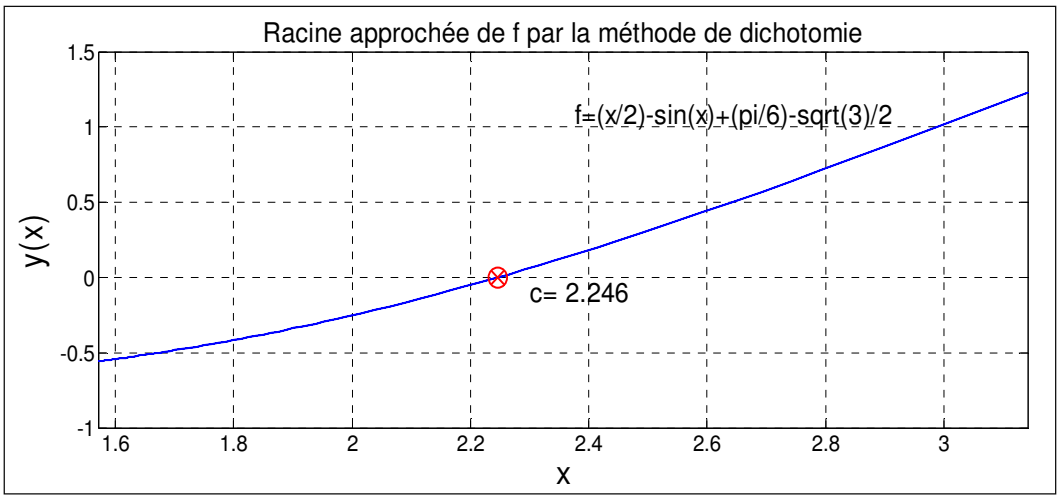


Fig. 7

Donc la méthode de dichotomie converge vers la racine approchée $x_n = c = 2.2460$, au bout de 17 itérations. Alors que celle de Newton converge vers la même solution seulement au bout de 4 itérations. La convergence est 4 fois plus rapide avec l'algorithme de Newton.

Partie (2)

Pour trouver le volume occupé par $N = 500$ molécule de CO_2 à T= 400 K et $P = 4.10^7\ Pa$:

a) On va appliquer la méthode de Newton pour l'équation non linéaire suivante :

$$f(V) = \left[P + a\left(\frac{N}{V}\right)^2\right](V - Nb) - KNT = 0 \ldots\ldots (*)$$

- La 1ère étape, on trace le graphe de la fonction $f(V)$ à fin de localiser la racine de l'équation (*) pour $V > 0$. Donc on ne considère pas les valeurs de $V < 0$, qui n'ont pas de sens physique, puisque V représente le volume du gaz.

Pour cette étape, nous exécutons le programme Matlab suivant :

```
close all; clear all; clc
p=4*10^7;N=500; a=0.401; T=400; b=42.7*10^-6;
k=1.3806503*10^(-23);
f=@(x)(p+a*((N/x)^2))*(x-N*b)-k*N*T
fplot(f,[0.0210 0.0215])
grid on
xlabel('V')
ylabel('f(V)')
title('Localisation de la solution de l'équation (*)');
```

Les résultats sont illustrés dans le graphe suivant :

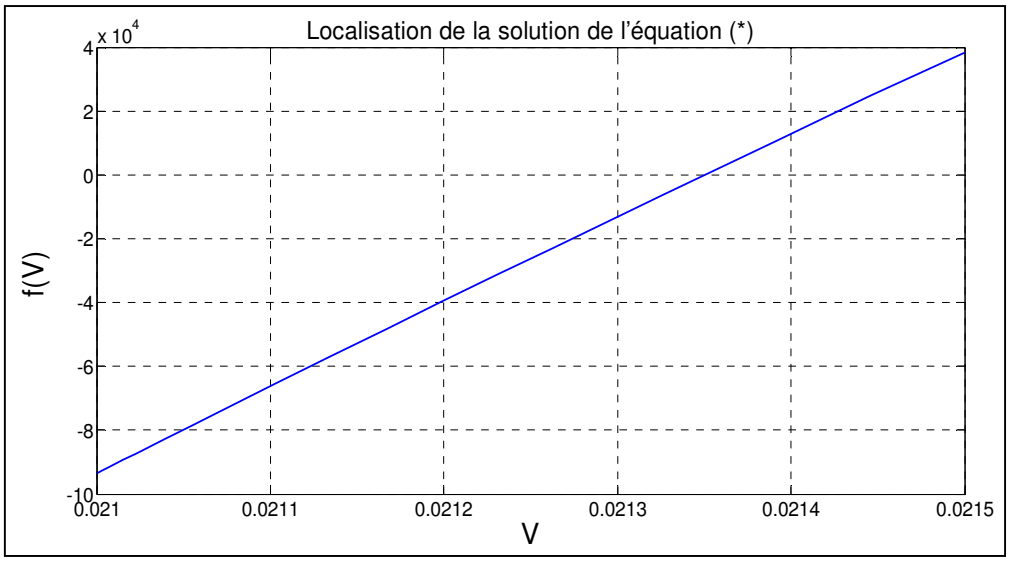


Fig. 8

- Dans la 2ème étape, on va utiliser l'algorithme de Newton pour calculer la solution approchée. D'après la courbe de la fonction $f(V)$ (**Fig. 8**), il est judicieux de choisir une valeur initiale $V_0 = 0.0213(m^3)$ pour avoir une convergence rapide ($f(0.0213 \approx 0)$. Nous exécutons le programme Matlab suivant :

```
close all; clear all; clc
p=4*10^7; N=500; a=0.401; T=400; b=42.7*10^-6;
k=1.3806503*10^(-23);
f=@(v)(p+a*((N/v)^2))*(v-N*b)-k*N*T
df=@(v)((-2)*a*N^2/(v^3))*(v-N*b)+(p+a*((N/v)^2))
e=1e-8; iter=0; v0=input('donner v0'); vi=0;
while abs(vi-v0)>e
    v1=v0-f(v0)/df(v0);
    vi=v0;
    v0=v1;
    iter=iter+1;
end
v1
iter
```

Remarque

On peut calculer et afficher la dérivée $df(V)/dV$ de la fonction $f(V)$ via un code Matlab simple et en même temps tracer $f(V)$ et $df(V)/dV$ une fois pour toutes (la commande `dfunsym` nous a donné $df(V)/dV$).

```
clc;clear all;close all;
p=4*10^7;N=500;a=0.401; T=400; b=42.7*10^-6; k=1.3806503*10^(-
23);
```

```
syms v real
funsym =(p+a*((N/v)^2))*(v-N*b)-k*N*T
n=100;lB=0.02;uB=0.03;h=(uB-lB)/n;
dfunsym = diff(funsym,v);pretty(dfunsym);vvar = lB:h:uB;
dfun = subs(dfunsym,v,vvar);fun = subs(funsym,v,vvar);
figure('Color',[1 1 1]);
[av,ih1,ih2]=plotyy(vvar,fun,vvar,dfun,'plot');grid on;
set(get(av(1),'Ylabel'),'interprete','latex','String',['$$f(v)
$$'],'FontSize',12);
set(get(av(2),'Ylabel'),'interprete','latex','String',['$$\fra
c{df(v)}{dv}$$'],'FontSize',12);
xlabel('\fontsize{12}\fontname{Tex}v');
hold on;
legend('f(v)','df(v)/dv')
```

L'exécution du code précédent nous a donné les résultats suivants :

```
dfunsym =
100250/x^2 - (200500*(x - 427/20000))/x^3 + 40000000
```

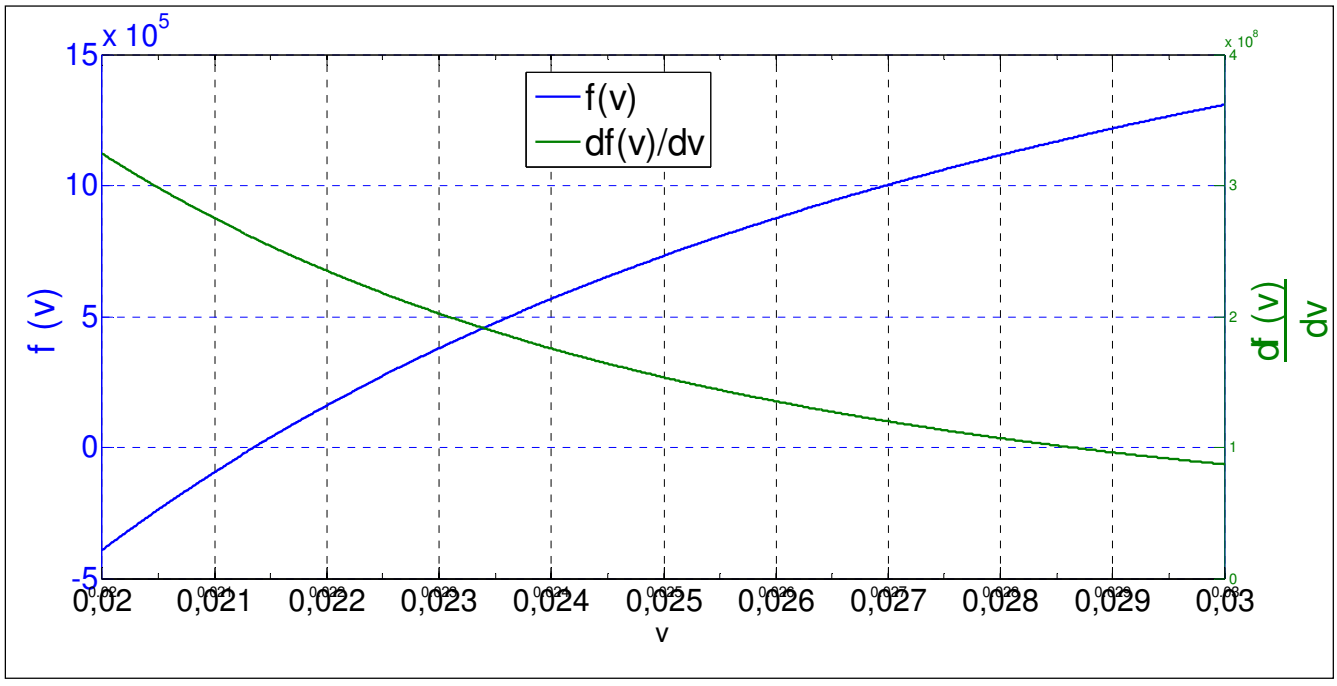


Fig. 9

b) Pour recalculer la solution approchée de l'équation (*) en utilisant la dichotomie, on exécute l'algorithme au dessous pour l'intervalle $[V_1 \ V_2] = [0.021 \ \ 0.0215]$) :

```
close all; clear all; clc
p=4*10^7; N=500; a=0.401; T=400; b=42.7*10^6;
k=1.3806503*10^(-23); v1=0.0210; v2=0.0215 ; v0=(v1+v2)/2;
tol=1e-8; iter=0;
```

```
while abs((p+a*((N/v0)^2))*(v0-N*b)-k*N*T) > tol
if      ((p+a*((N/v1)^2))*(v1-N*b)-k*N*T)*((p+a*((N/v0)^2))*(v0-
N*b)-k*N*T) <0
v2=v0;
end
if   ((p+a*(N/v0)^2)*(v0-N*b)-k*N*T)*((p+a*((N/v2)^2))*(v2-N*b)-
k*N*T)<0
v1=v0;
end
v0=(v1+v2)/2; iter=iter+1;
end
v0
iter
```

Le résultat de calcul de la méthode de Newton nous a donné la solution : $V = 0.0214\, m^3$ après seulement 4 itérations, par contre, pour la méthode de dichotomie avec la même tolérance, le même résultat est obtenu qu'après 41 itérations. Donc la convergence vers la solution exacte de l'algorithme de Newton est plus rapide par rapport à l'algorithme de dichotomie.

On peut conclure que :

- La méthode de dichotomie est simple à mettre en œuvre mais, elle converge lentement parce que son critère d'arrêt consiste à contrôler la longueur de l'intervalle $[V_1\ V_2]$ à chaque itération.
- La méthode de Newton est rapide et très utile, à condition que le choix de la valeur initiale soit la plus proche possible de la racine recherchée. Mais malheureusement dans certains cas nous n'avons pas accès à la dérivée de la fonction f. C'est pourquoi on s'est orienté vers d'autres méthodes numériques.

Exercice

On considère le problème cité précédemment ; résolution de l'équation d'équilibre d'un pendule simple soumis à l'énergie potentielle de gravitation et l'´energie potentielle électrostatique :

$$\boldsymbol{f(\theta) = \sin\theta + \frac{qq'}{4\pi\varepsilon_0 mg}\ \frac{x_q\cos\theta + (y_q - \ell)\sin\theta}{[(\ell\sin\theta - x_q)^2 + (\ell(1-\cos\theta) - y_q)^2]^{\frac{3}{2}}} = 0} \ \ldots\ldots (*)$$

On prend par exemple : $\frac{qq'}{4\pi\varepsilon_0 mg} = c$, avec $0.25 \le c \le 2$, $x_q = 0.1$, $y_q = 0$ et $\ell = 1$

1. Pour trouver les angles d'équilibres positifs θ_i (i=1, 2,…8), solutions de l'équation non linéaire (*) dans l'intervalle $-1.57 \leq \theta \leq 1.57$, on exécute l'algorithme de Newton pour chaque valeur de c, avec $c = [0.25\ \ 0.5\ \ 0.75\ \ 1\ \ 1.25\ \ 1.5\ \ 1.75\ \ 2]$ et :

$$\frac{df}{d\theta} = cos(\theta) + c * \frac{G(\theta)}{((\sin(\theta) - \ 0.1)^2 + (1 - \cos(\theta)^2)^3}$$

Avec : $G(\theta) = \big(0.1\ sin(\theta) - cos(\theta)\big)((sin(\theta) - 0.1)^2 + (1 - \cos(\theta))^2)^{\frac{3}{2}} -$

$1.5\ \big(0.1\ cos(\theta) - sin(\theta)\big)((sin(\theta) - 0.1)^2 + (1 - cos(\theta))^2)^{0.5}\ \Big(\Big(2\ (sin(\theta) -$

$0.1)\ cos(\theta) + 2\ \big(1 - cos(\theta)\big)sin(\theta)\Big)\Big)$

Remarque :

On peut calculer facilement la dérivée de la fonction $f(\theta)$ via le même code exécuté dans l'exercice précédent.

- Donner l'algorithme exécuté pour trouver chaque angle d'équilibre θ_i (Calculer les 8 valeurs positives de θ_i).

2. Après avoir calculé les racines positives θ_i , on trace la variation d'angle d'équilibre en fonction de $c = \frac{qq'}{4\pi\varepsilon_0 mg}$ (en fonction de charge). Les résultats sont représentés dans la figure suivante :

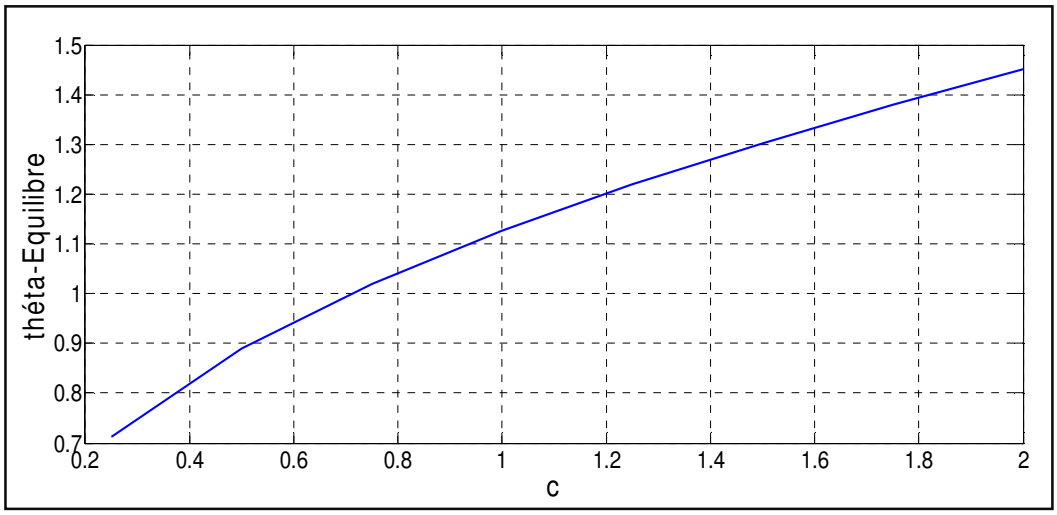


Fig. 10 – Variation d'angle d'équilibre du pendule chargé en fonction de charge.

- Donner l'algorithme qui correspond à cette figure.

3. De la même manière, calculer les angles d'équilibres négatifs θ_i (i=1, 2,…8), solutions de l'équation (*) sur l'intervalle $-1.57 \leq \theta \leq 1.57$. En prenant les mêmes valeurs de c dans la 1ère question.
4. Donner l'algorithme qui nous a permis de tracer l'angle d'équilibre en fonction de $c = \frac{qq'}{4\pi\varepsilon_0 mg}$ (en fonction de charge) en deux courbes différentes (en sens positif et négatif), sur une même figure. Commenter les résultats.
5. Refaire les mêmes étapes précédentes en utilisant la méthode de dichotomie. Comparer les résultats.

TP N° 3 - L'interpolation polynomiale d'une fonction

1. Introduction

L'approximation d'une fonction f évaluée initialement sur un segment $[a, b]$, et connue uniquement par ses valeurs en certains points consiste à remplacer ou à approcher f par une fonction plus simple, le plus souvent par un polynôme, en utilisant par exemple la procédure d'approximation par interpolation.

Donc les polynômes sont à priori de bons candidats pour approximer des fonctions ; nous allons voir ci-dessous qu'il existe des situations où le fait de prendre un polynôme de degré de plus en plus élevé détériore la qualité de l'approximation.

2. But du TP

Substitution d'une fonction $f(x)$ connue analytiquement mais difficile à manipuler par une fonction polynomiale plus simple. Dans le cas où f est non connue, l'interpolation polynomiale nous a permis de construire une représentation synthétique surtout, quand le nombre de points devient très élevé (points de mesures expérimentaux ou théoriques).

3. Exemples de motivation

Exemple 1

En relevant la vitesse de d'écoulement de l'eau dans une conduite cylindrique, dans chaque 10 secondes, on a obtenu :

t	0	10	20	30
v	2.00	1.89	1.72	1.44

Grace à une interpolation polynomiale de degré < 4 de la fonction vitesse v, on peut trouver des approximations de cette fonction, dans n'importe quel instant dans l'intervalle de mesure $[0\ \ 30\ s]$. Par exemple, pour $t = 15\ (s)$, via un polynôme interpolant de degré 2, on va trouver $v(15\ s) \approx 1.0105\ m/s$ et via un polynôme interpolant de degré 3, on va trouver $v(15\ s) \approx 1.8320\ m/s$. L'exécution du code suivant dans le Matlab nous a permet de faire l'estimation :

```
close all; clear all; clc
n=4;x=[0 10 20 30];y=[2 1.89 1.72 1.44];
p=polyfit(x,y,n)
polyval(p,15)
```

Avec n le nombre de points de mesures choisi.

Exemple 2

Dans le domaine de la physique des matériaux, l'étude théorique des propriétés électroniques d'un alliage AB_xC_{1-x}, repose sur l'analyse de la variation du gap énergétique en fonction de la concentration x de l'élément dopé. Généralement, cette étude est effectuée grâce à une interpolation polynomiale de degrés 2 de la variation du gap en fonction de la concentration *x*. Sachant que, le gap énergétique d'alliage AB_xC_{1-x} est décrit en termes de gaps des composés binaires E_{AB} et E_{AC} par la formule:

$$E_g = xE_{AB} + (1-x)E_{AC} - x(1-x)b$$

Où la courbure *b* est connue sous le nom de 'paramètre de '*bowing*'. Ce '*bowing*s' du gap représente l'écart à la linéarité et indique le désordre dans l'alliage. Il est obtenu numériquement, en ajustant les courbes de la variation du gap en fonction de la concentration *x* à l'aide d'une fonction polynomiale de la forme : $E = E_0 + ax + bx^2$ où *b* est le paramètre de désordre.

En outre, plusieurs paramètres physiques peuvent être déduits, en utilisant l'approximation des fonctions et l'ajustement des courbes via l'interpolation polynomiale.

4. Rappel sur les méthodes d'interpolation polynomiale

Le problème qui se pose est de savoir si l'ensemble des données doit être utilisé en une seule fois pour déterminer une unique fonction sur tout l'intervalle considéré ou si on doit construire une succession de fonctions approchantes en utilisant des données locales. La réponse est adaptée en fonction des propriétés de régularité de la fonction que l'on veut interpoler ainsi que le type de fonctions approchantes utilisées. On cite dans ce qui suit trois types de ces fonctions polynomiales : Lagrange, Newton et Chebyshev.

4.1. Polynôme de Lagrange

Le polynôme de Lagrange est donné par :

$$\boldsymbol{P_n(x) = \sum_{i=0}^{n} y(x_i)\, L_i(x),} \qquad \text{avec} \qquad \boldsymbol{l_i(x) = \prod_{\substack{i=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j} =}$$

$\frac{(x-x_0)(x-x_1).....(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1).....(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$ et on a $\boldsymbol{l_i(x_i)}$=1 et $\boldsymbol{l_i(x_j)} = 0$ si $i \neq j$

Cette relation est appelée formule d'interpolation de Lagrange et les polynômes $\boldsymbol{l_i}$ sont ses polynômes caractéristiques. Le degré de $L_i(x)$ est *n* et le degré de $P_n(x)$ est inferieur ou égal à *n*.

Exemple

On donne les points (-1 , 8) , (0 , 4) , (1 , 2).

$P_2(x) = \sum_{i=0}^{n=2} y_i L_i(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x)$,

$$L_0(x) = \frac{(x-0)(x-1)}{(-1-0)(-1-1)} = \frac{1}{2}(x^2 - x), L_1(x) = \frac{(x+1)(x-1)}{(0+1)(0-1)} = -(x^2-1)$$

$$L_2(x) = \frac{(x+1)(x-0)}{(1+1)(1-0)} = \frac{1}{2}(x^2 + x)$$

$P_2(x) = 8\left[\frac{1}{2}(x^2 - x)\right] + 4\left[-(x^2-1)\right] + 2\left[\frac{1}{2}(x^2 + x)\right] = x^2 - 3x + 4$. C'est le polynôme d'interpolation.

Algorithme de calcul pour le polynôme de Lagrange :

Le calcul analytique montre ses limites pour des degrés n élevés. Donc, il est nécessaire de développer un algorithme permettant l'implémentation de la méthode de Lagrange afin de déterminer les polynômes quelque que soit le degré n.

Les données de l'algorithme de Lagrange :

- Le vecteur $x = (x_1, x_2, \ldots x_n)$ formé des points d'interpolation.
- Le vecteur $y = (y_1, y_2, \ldots y_n)$ formé des valeurs d'interpolations.
- Le point x en lequel on veut calculer $Pn(x)$.

Le résultat est dans Pn :

- $Pn = 0$
- Pour $i \in [1, n]$ fair

  $Lag = 1$

  Pour $j \in [1, n]$ et $i \neq j$, $Lag = Lag * (t - x(j))/(x(i) - x(j))$

  $Pn = Pn + Lag * y(i)$

4.2. Polynôme de Newton

Le polynôme de newton pour $N+1$ points $x_0, \ldots, x_n$ ainsi que $N+1$ valeurs associées $y_0, \ldots . y_n$ est donné par :

$P_n(x) = f(x_0) + (x - x_0) f[x_1, x_0] + (x - x_0)(x - x_1) f[x_2, x_1, x_0] + \cdots + (x - x_0)(x - x_1) \ldots (x - x_{n-1}) f[x_n, x_{n-1}, \ldots, x_0]$

Avec la différence divisée: $f[x_n, x_{n-1}, \ldots, x_0] = \frac{f[x_n, x_{n-1}, \ldots, x_1] - f[x_{n-1}, x_{n-2}, \ldots, x_0]}{x_n - x_0}$

Exemple

On donne les points (-1 , 8), (0 , 4) et (1 , 2). Le tableau des différences divisées est le suivant :

x_i	$f(x_i)$	Ordre 1	Ordre 2
-1	8	$[x_0, x_1] = -4$	$[x_0, x_1, x_2] = 1$
0	4	$[x_2, x_1] = -2$	
1	2		

$$P_2(x) = f(x_0) + [x_0, x_1](x - x_0) + [x_0, x_1, x_2](x - x_0)(x - x_1$$
$$= 8 + (-4)(x + 1) + 1(x + 1)(x - 0) = x^2 - 3x + 4$$

Algorithme de calcul pour le polynôme de Newton :

Pour construire le polynôme de Newton, on peut utiliser un programme en exécutant les étapes suivantes :

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

$D(i,j) = 0$ pour $i = 1 \ldots (n)$ et $j = 1 \ldots . (n)$ avec n le nombre des points x_i.

$D(i,1) = f(x_i)$ pour $i = 1 \ldots (n)$

$$D(i,j) = \frac{D(i,j-1) - D(i-1,j-1)}{x(i) - x(i-j+1)} \quad \text{pour } j = 2 \ldots (n) \text{ et } i = j \ldots (n)$$

- Extraire en suite la diagonale de D dans un vecteur M :

$M(i) = D(i,i)$ pour $i = 1 \ldots (n)$

- Ecrire l'expression du polynôme de Newton :

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

4.3 Polynôme de Tchebychev

Dans le cas de l'interpolation de Lagrange et celle de Newton, on a la liberté de choisir les nœuds d'interpolation $\{x_i\}$ dans l'intervalle considéré $[a, b]$. Par contre, l'interpolation de Tchebychev, impose un choix des nœuds appelés points de Tchebychev. L'expression des nœuds est montrée dans relation suivante :

$$x_i = \frac{(b+a)}{2} + \frac{(b-a)}{2} \times cos\left(\frac{2(n-i)+1}{2n}\right)\pi \qquad \text{avec } i = 0, 1, \ldots n$$

Il s'agit donc d'une interpolation de Lagrange menée en des points particuliers.

L'intérêt de choisir les nœuds de Tchebychev est de se prémunir contre le phénomène de Runge, qu'on va expliquer ultérieurement.

5. Effet de Runge

Il existe des situations où le fait de prendre un polynôme de degré de plus en plus élevé détériore la qualité de l'approximation. Dans le cas des fonctions du type $g_\alpha = \frac{1}{1+\alpha^2 x^2}$ où $h_\alpha(x) = \frac{1}{\alpha^2+x^2}$, on constate que l'interpolation via les points équidistants ne donne pas un résultat proche de la fonction interpolée. En augmentant le degré du polynôme, on voit clairement que l'approximation est meilleure au centre mais se détériore gravement sur les bords de l'intervalle. On constate des oscillations de grandes amplitudes près des bords.

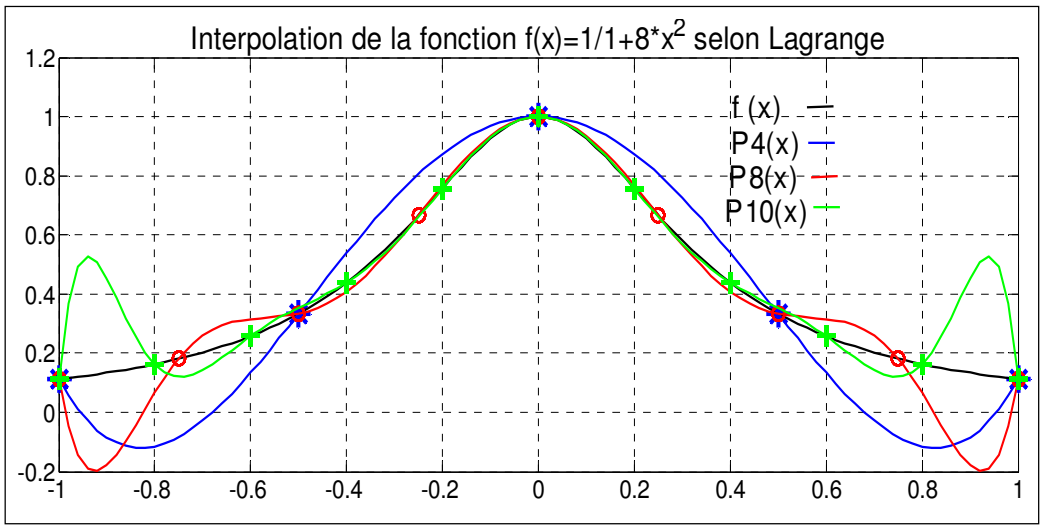


Fig. 1 – Phénomène de Runge - Interpolation via Lagrange

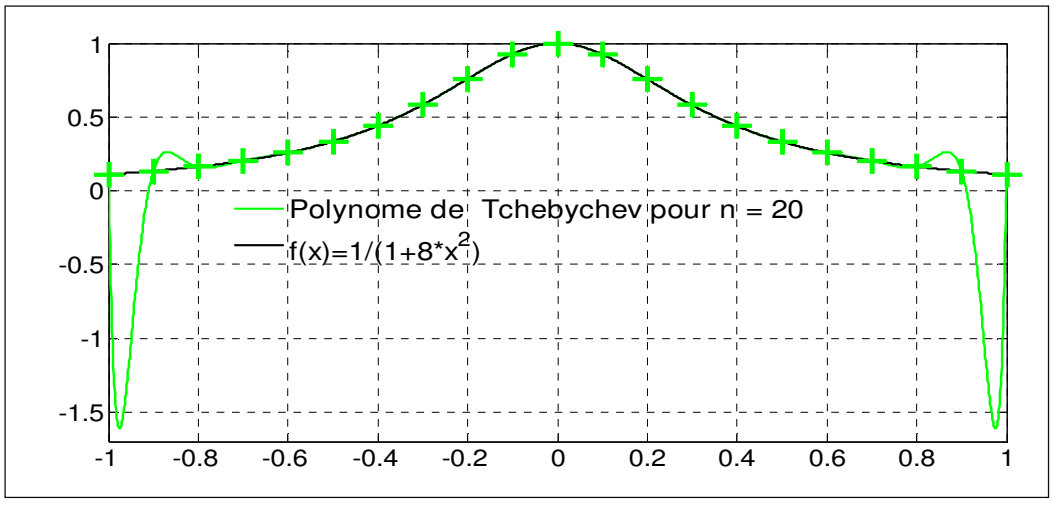


Fig. 2 – Phénomène de Runge - Interpolation via Tchebychev

Ce phénomène (dit de Runge) est bien montré dans les figures (Fig.1) et (Fig.2), pour l'interpolation de Lagrange de 4ème, 8ème et 10ème degré et pour l'interpolation de Tchebychev de 20ème degré, respectivement. Nous allons voir dans ce TP comment choisir convenablement

les nœuds pour établir la convergence uniforme du polynôme d'interpolation vers la fonction *f*.

6. L'erreur d'interpolation

Lorsqu'on remplace la fonction f par le polynôme d'interpolation équivalent, on va commettre une erreur notée par $\varepsilon(x)$ qui est une fonction varie dans l'intervalle d'interpolation. Cette erreur doit être nulle aux points d'interpolation, $\varepsilon(x_i) = 0$, $(i = 0, \dots n)$.

$$\varepsilon(x) = |f(x) - P_n(x)|$$

7. Enoncé du TP

Partie (a)

1. Construire le polynôme d'interpolation P_3 (x) de degré trois qui interpole les points : (x_0, y_0) = (-2, -6) ; (x_1, y_1) = (-1, 0), (x_2, y_2) = (1, 0) et (x_3, y_3)= (2, 6).
2. Écrire un algorithme Matlab permettant l'implémentation de la méthode de Lagrange. Déterminer ce polynôme.
3. Tracer, sur la même figure, le polynôme et les points d'interpolation.
4. Écrire un algorithme Matlab permettant de donner le polynôme de Newton P_3 (x) de degré trois qui interpole les points : (x_0, y_0) = (-2, -6) ; (x_1, y_1) = (-1, 0), (x_2, y_2) = (1, 0) et (x_3, y_3)= (2, 6) et (x_4, y_4)= (4, 60).

Partie (b)

Voyons maintenant le problème de l'interpolation du point de vue de l'approximation. On donne un exemple concret de fonction analytique *f* pour laquelle les polynômes d'interpolation ne forment pas une suite convergente. Supposons que nous devons nous rapprocher la fonction :

$$f(x) = \frac{1}{1 + 8x^2}$$

1. On cherche les fonctions polynomiales des degrés, $4, 8$ et 10 ($P4(x)$, $P8(x)$ et $P10(x)$) en utilisant la méthode de Lagrange, Newton et Tchebychev ($x_i = \{-1, -0.5, 0, 0.5, 1\}$), en calculant la fonction d'erreur dans chaque cas. Comparer les résultats.

2. Tracer la fonction $f(x)$ et les autres fonctions polynômiales pour plusieurs degrés dans le même graphe puis faire la même chose pour tracer la fonction d'erreur dans chaque cas. Commenter les résultats.

8. Solution

Partie (a)

1. Pour obtenir le polynôme d'interpolation de degré $n < 4$, on peut exécuter le code suivant :

```
x = [-2 -1 1 2]; y = [-6 0 0 6];
p=polyfit(x,y,n)
```

2. L'exécution du code suivant nous a permis de calculer le polynôme de Lagrange 3ième degré (**Fig. 3**) :

```
close all; clear all; clc
x = [-2 -1 1 2]; y = [-6 0 0 6];
n=4;
h=[x(1):0.02:x(4)];
Pn=0;
for i=1:n
    lag=1;
    for j=1:n
        if i~=j
            lag=lag.*(h-x(j))/(x(i)-x(j))
        end
        figure(1);
       plot(x(i),y(i),'r*','MarkerSize',16,'LineWidth',2)
       grid on
       hold on;
    end
    Pn=Pn+lag.*y(i);
end
plot(h,Pn,'b-','LineWidth',1)
hold on; xlabel('x'); ylabel('y');
title('Interpolation-selon Lagrange')
text('Interpreter','latex','String','$Pn= X^3-
X$','Position',[-0.5 0.5],'FontSize',16);
polyfit(h,Pn,3)
p=polyfit(x,y,3)
```

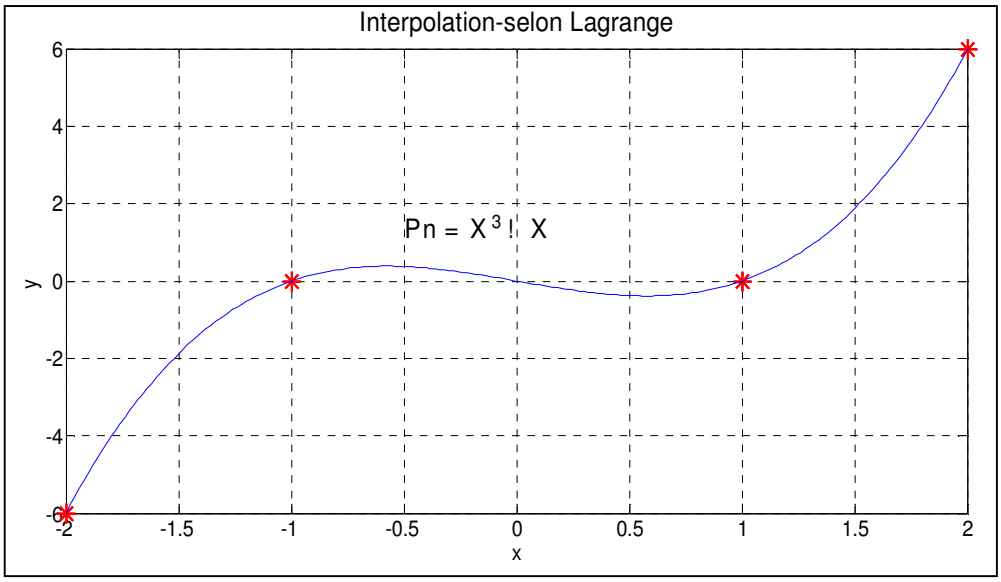


Fig. 3 - Le graphe de polynôme d'interpolation de degré 3 selon Lagrange avec les point d'interpolation.

4. On peut exécuter le code suivant pour trouver le polynôme de Newton 3^{ième} degré.

```
close all; clear all; clc
x = [-2 -1 1 2 4]; y = [-6 0 0 6 60];
D=zeros(5,5);
D(:,1)=y';
for j=2:5
    for i=j:5
        D(i,j)=(D(i,j-1)-D(i-1,j-1))/(x(i)-x(i-j+1));
    end
end
m=diag(D)
t=[-2:0.02: 4]
s=m(1);
for i=2:5
    p=1;
    for k=1:i-1
        p=p.*(t-x(k));
    end
    p
    s=s+p.*m(i);
end
s
polyfit(t,s,3)
```

Les réponses affichées **:**

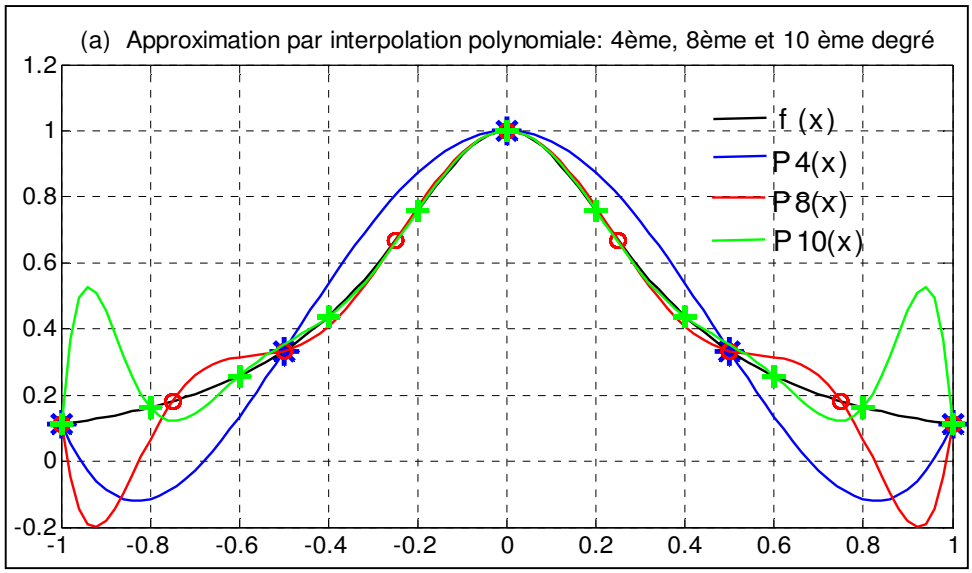
```
m =
   -6
```

6
-2
1
0

On trouve le même polynôme 3ième degré ($P_3 = x^3 - x$).

Partie (2)

```
close all; clear all; clc
n=5; x=[-1 -0.5 0 0.5  1]; y=1./(1+8*x.^2);
xChe=(x(5)+x(1))/2+((x(5)x(1))/2)*cos((2*(n-
(1:n))+1)*pi/(2*n);
xChe=x;
D=zeros(5,5); D(:,1)=y';
for j=2:5
    for i=j:5
        D(i,j)=(D(i,j-1)-D(i-1,j-1))/(xChe(i)-xChe(i-j+1));
    end
end
m=diag(D);
xvar=[-1:0.001:1];
s=m(1);
for i=2:5
    p=1;
    for k=1:i-1
        p=p.*(xvar-xChe(k));
    end
    p
    s=s+p.*m(i);
end
s
figure (1);
hold on
plot(xvar,s,'b-')
grid on
hold on
plot(xvar,1./(1+8*xvar.^2),'-k')
hold on
plot(x,y,'b*','MarkerSize',18,'LineWidth',3)
title('Interpolation aux noeuds de Tchebychev pour n = 4')
legend('$Polynome de Tchebychev pour n =
4$','$f(x)=1./(1+8*x.^2)$')
```



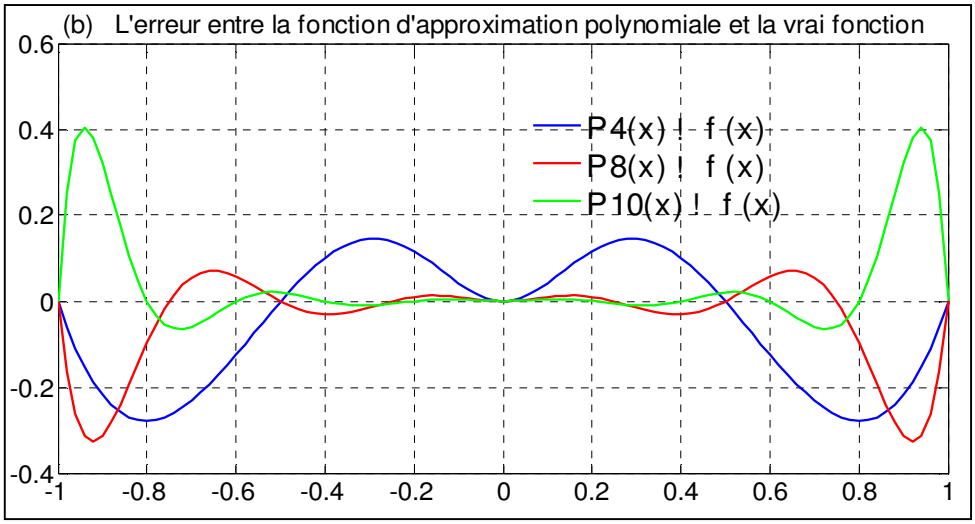


Fig. 4 (a / b) - Interpolation polynomiale de point de vu approximation

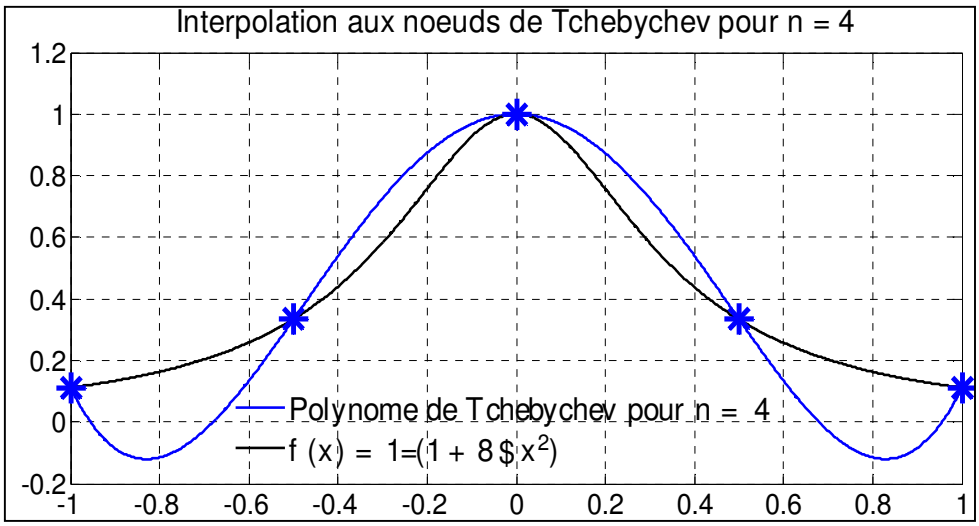


Fig. 5

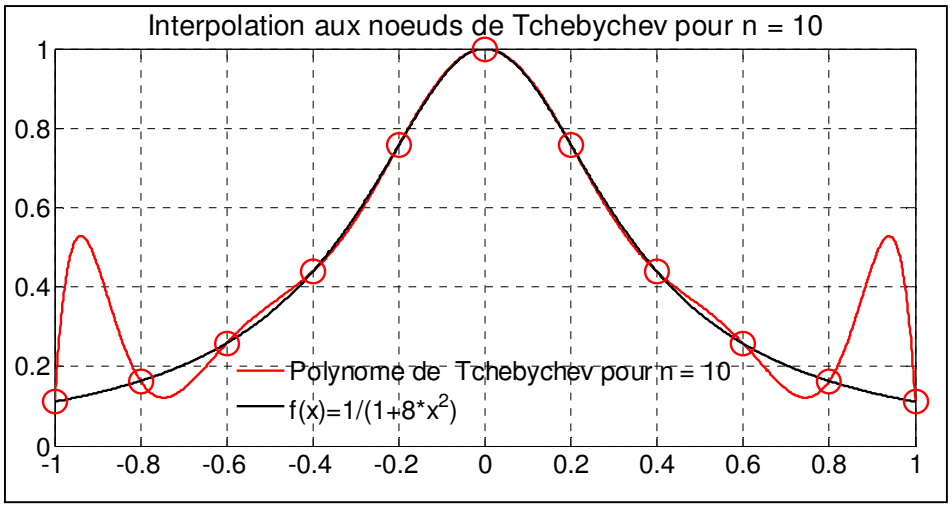

Interpolation aux noeuds de Tchebychev pour n = 10
Polynome de Tchebychev pour n = 10
f(x)=1/(1+8*x^2)

Fig. 6

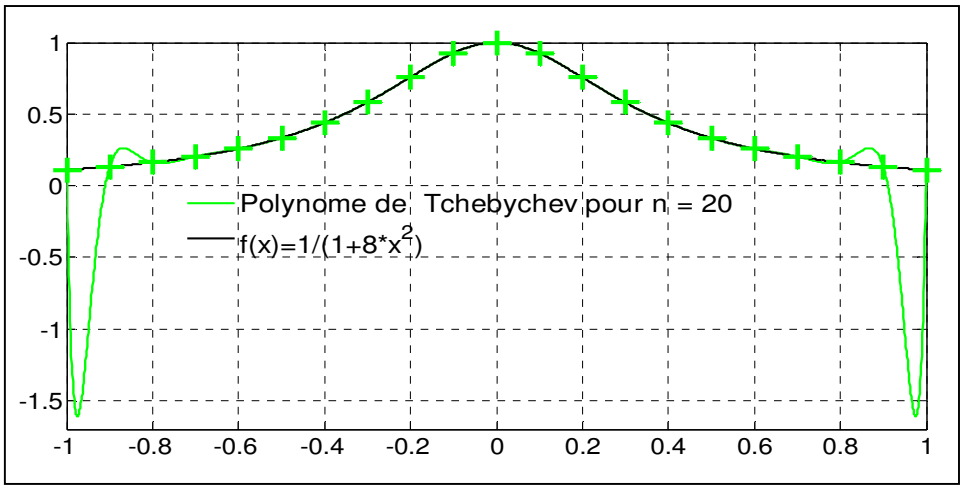
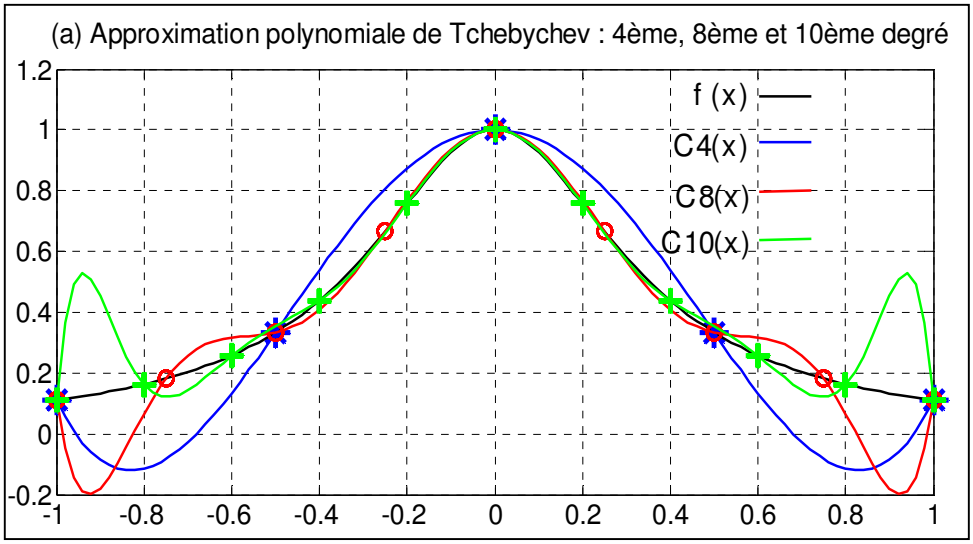

Polynome de Tchebychev pour n = 20
f(x)=1/(1+8*x^2)

Fig. 7


(a) Approximation polynomiale de Tchebychev : 4ème, 8ème et 10ème degré
f (x)
C4(x)
C8(x)
C10(x)

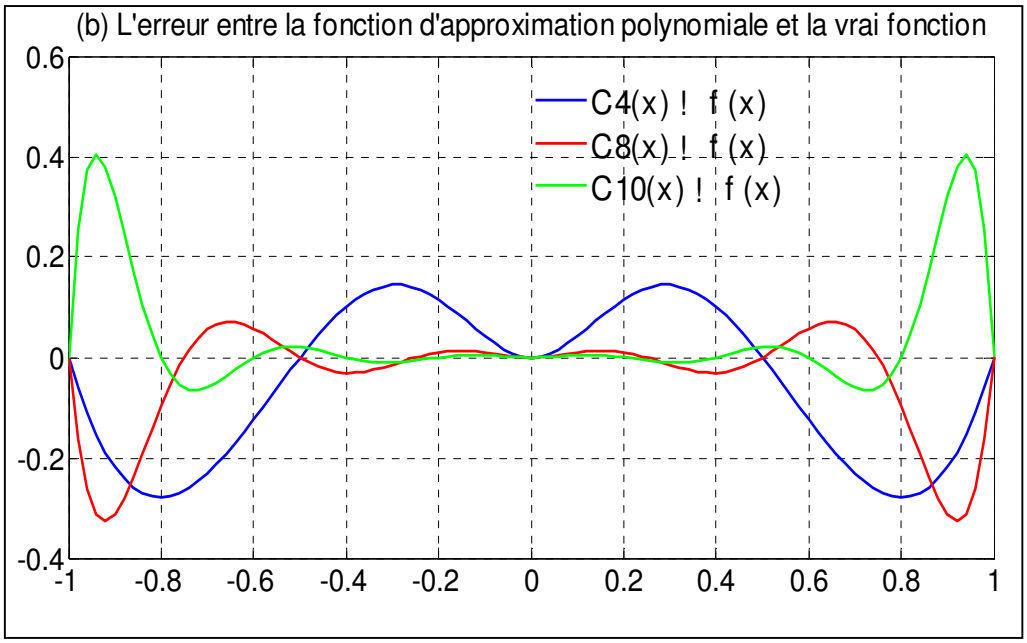


Fig. 8 (a/b)

- À partir de la figure 4(a), on remarque une divergence au voisinage des bornes de l'intervalle entre le polynôme interpolateur de Lagrange et la courbe $f(x)$. Par ailleurs, plus le degré n de ce polynôme est élevé plus la divergence (oscillations sur les bords de $[a,b]$) aux bords de l'intervalle est grande. C'est le phénomène de Range.
- L'interpolation de Lagrange diverge pour $x > 0.45$, au contraire pour le polynôme de Tchebychev, l'interpolation diverge pour $x > 0.6$ dans le cas de polynôme 10ième degré et diverge pour $x > 0.8$ dans le cas de polynôme $20^{\text{ième}}$ degré. Ce phénomène est particulièrement évident au voisinage des extrémités de l'intervalle d'interpolation, comme le montre la figure 4(a).
- On voit clairement que l'approximation est meilleure au centre mais l'erreur devient plus grande dans les pièces proches des deux extrémités comme on peut le voir dans les figures 4(b) et 8(b), c'est pourquoi les polynômes de degré 5 ou plus sont rarement utilisés à des fins d'interpolation.

Exercices

Exercice 1

Le tableau suivant présente la conductivité thermique de la vapeur d'acétone en fonction de la température :

(a) Estimer la conductivité thermique de l'acétone à $300\ ^{o}F$.

(b) Estimer la température en ^{o}F qui correspond à la conductivité thermique $k = 0.008\ Btu/hr\ ft\ ^{o}F$.

$T\ (^{o}F)$	$k\ (Bru/hr\ ft\ ^{o}F)$
32	0.0057
115	0.0074
212	0.0099
363	0.0147

Exercice 2

Le tableau suivant présente la contrainte de cisaillement en *kilo Pascal* (kPa) dans une strate d'argile varie avec la profondeur h en mètre (m).

$h(m)$	2	3	5	7
$\tau(kPa)$	18	35	75	163

1. Utiliser les mesures expérimentales ci-dessus pour évaluer la contrainte de cisaillement τ au point $h = 4.5\ m$ en utilisant l'interpolation polynomial de Lagrange et puis vérifier le résultat en utilisant seulement la commande $'polyfit\ '$ (Donner le programme Matlab correspondant).

2. Pour obtenir d'autres approximations de la fonction τ au point $h = 4.5\ m$:
   - Ecrire un programme pour obtenir à la fois le polynôme d'interpolation de degré 2 en utilisant la commande $'polyfit\ '$ et les deux polynômes de Lagrange de degré 2 et de degré 3.
   - Calculer les trois valeurs approximatives de $\tau\ (h = 4.5\ m)$ puis comparer ces dernières avec la valeur théorique $\tau_{exacte}\ (h = 4.5\ m) = 62.5500\ (kPa)$. Conclure.

Compléter le programme utilisé précédemment pour tracer tout les points d'interpolation et le polynôme de Lagrange trouvé de degrés 3 dans la même figure.

TP N° 4 - Résolution des équations différentielles ordinaires

1. But du TP

Ce TP consiste à résoudre les équations différentielles ordinaires en utilisant des algorithmes des méthodes de base : méthode d'**Euler**, méthode de **Range Kutta2** d'ordre 2 et la méthode de **Range Kutta4** d'ordre 4. Dans ce TP, nous allons comparer les trois solutions approchées avec la solution exacte.

2. Exemple de motivation

Mouvement du pendule :

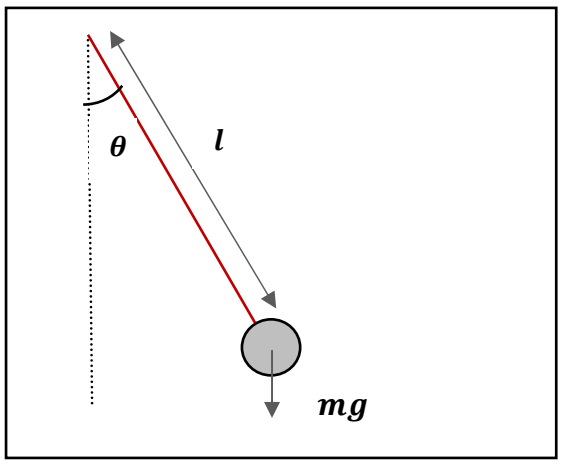


Fig. 1 - Mouvement d'un pendule simple

- On cherche la solution de l'équation différentielle suivante qui d'écrit le mouvement d'un pendule simple :

$$\begin{cases} \dfrac{d^2\theta(t)}{dt^2} + \dfrac{g}{l}\sin(\theta(t)) = 0 \\ \theta(0) = \theta_0 \quad , \quad \dfrac{d\theta(0)}{dt} = \theta_1 \end{cases}$$

- Pour simplifier la résolution, on pose : $\theta \ll 1$ et la solution approchée sera :

$$\begin{cases} \dfrac{d^2\theta(t)}{dt^2} + \dfrac{g}{l}\theta(t) = 0 \\ \theta(t) = \ \theta_0 \ \cos\sqrt{\dfrac{g}{l}}t + \sqrt{\dfrac{l}{g}}\ \theta_1 \sin\sqrt{\dfrac{g}{l}}t \end{cases}$$

- Chercher d'autres solutions approchées en utilisant des méthodes d'approximation numérique ?

3. Rappel sur les méthodes de résolution numériques

On considère une équation différentielle d'ordre 1 sous forme résolue $y' = dy/dx = f(x,y)$ avec la condition initiale $y(x_0) = y_0$.

Méthode d'Euler

Le principe est d'approcher la solution y sur $[a\ b]$ par une fonction affine par morceaux, en opérant une discrétisation du paramètre, on pose : $x_i = a + i\ h$ où $h = (b-a)/n$ est le pas.

La fonction affine par morceaux joindra donc les points de coordonnées (x_i, y_i) et il s'agit de proposer un algorithme pour construire les $y_{\ i}$ à partir de y_0 :

$$\begin{cases} h = x_i - x_{i-1} \\ y_{i+1} = y_i + h\, f(x_i\ ,\ y_i)\ ;\ f(x\ ,y) = \dfrac{dy}{dx} \end{cases}$$

Cette méthode est d'ordre 1, cela veut dire que l'erreur est proportionnelle au carré du pas (h) de discrétisation. Donc il suffi de réduire le pas h pour améliorer la précision.

Méthode de Range Kutta d'ordre 2 (formule de Heun)

Le schéma numérique de cette méthode résulte de l'application de la formule de quadrature du trapèze. Notons également que la méthode de Heun fait partie des méthodes de Runge-Kutta explicites d'ordre deux.

$$y_{i+1} = y_i + \frac{h}{2} f(x_i\ ,\ y_i) + \frac{h}{2} f\big(x_i + h, y_i + h f(x_i\ ,\ y_i)\big)\ ;\ f(x\ ,y) = \frac{dy}{dx}$$

Méthode de Range Kutta4 d'ordre 4

$$\mathrm{k}_1 = f(x_0\ ,y_0) \quad , \quad \mathrm{k}_2 = f\left(x_0 + \tfrac{h}{2}\ , y_0 + \tfrac{h}{2} * \mathrm{k}_1\right)$$

$$\mathrm{k}_3 = f\left(x_0 + \tfrac{h}{2}\ , y_0 + \tfrac{h}{2} * \mathrm{k}_2\right) \quad , \quad \mathrm{k}_4 = f(x_0 + h\ ,\ y_0 + h * \mathrm{k}_3)$$

$$y_1 = y_0 + \frac{h}{6} * (\mathrm{k}_1 + 2 * \mathrm{k}_2 + 2 * \mathrm{k}_3 + \mathrm{k}_4)$$

4. Travail demandé

Partie (1)

Soit l'équation différentielle :

$$\begin{cases} \dfrac{dy(x)}{dx} = 2\ y(x) \\ y(0) = 5 \end{cases}$$

1. Vérifier que la solution analytique (exacte) est $y(x) = 5\ e^{2x}$.

2. Résoudre cette équation avec un pas d'intégration $h = 0.1$ en écrivant un programme permettant de calculer 10 itérations de la méthode d'Euler, puis de la méthode de Range Kutta d'ordre 2 (méthode de Heun), en comparant à chaque fois les solutions approximatives par la solution exacte ($y(x_i) = 5\exp(2\,x_i)$).
3. Compléter le programme précédent pour faire tracer et comparer les courbes ; solution exacte, solution approximative d'Euler et solution approximative de Range Kutta d'ordre 2 (formule de Heun), quel est votre commentaire ?
4. Prendre $h = 0.01$ et refaire la question 3. Quel est votre commentaire ?
5. Refaire la question 2 pour la méthode de Range Kutta d'ordre 4 et compléter le tableau suivant :

Méthode	Euler	Range Kutta2 (formule de Heun)	Range Kutta4	Analytique (solution)
Solution				
Précision				/

6. Tracer les courbes des solutions approximatives, d'Euler, de Range Kutta d'ordre 2 (formule de Heun) et de Range Kutta d'ordre 4. Commenter les résultats et conclure.

Partie (2)

On veut résoudre numériquement l'équation différentielle du mouvement $(*)$ d'un oscillateur harmonique (libre non amorti) en utilisant l'algorithme d'Euler :

$$\begin{cases} \dfrac{d^2x(t)}{dt^2} = -\dfrac{K}{m}x(t) \\ x(0) = 2 \text{ et } \dfrac{dx(0)}{dt} = 0 \end{cases} \quad \dots\dots(*)$$

1. Ecrire un programme Matlab pour trouver les valeurs approximatives de la solution de l'équation $(*)$ aux points de l'intervalle $[0,\ 15]$ avec un pas d'intégration $\Delta t = 0.1$ en prenant $m = 5\ Kg, K = 3\ N\ m^{-1}$, le nombre d'itération $N = {}^{t_f}/_{\Delta t}$ avec $t_f = 15$.
2. Etudier graphiquement la variation de la position *x(t)* en fonction du nombre d'itérations N. Commenter les résultats.

5. Solution

Partie (1)

```
clc, clear all, close all
t=zeros(10,1);
y=zeros(10,1);
```

```
t(1)=0;y(1)=5;
T=[0:0.1:1]';
for i=1:10
    t(i+1)=t(i)+0.1;f(i)=2*y(i);
    y(i+1)=y(i)+0.1*f(i);yr=5*(exp(2*T));
end
[t,yr,y]
plot(T,y,'o')
xlabel('T')
ylabel('y(T)')
hold on
plot(T,yr,'*k')
t=zeros(10,1);y=zeros(10,1);t(1)=0;y(1)=5;T=[0:0.1:1]';
for i=1:10
    t(i+1)=t(i)+0.1;
    f(i)=2*y(i);
    y(i+1)=y(i)+0.1*1.1*f(i);
end
[t,y]
figure (1)
plot(t,y,'+r','MarkerSize',2,'LineWidth',6)
legend('solution                    approximative-Euler','solution
exacte','solution approximative-Heun')
```

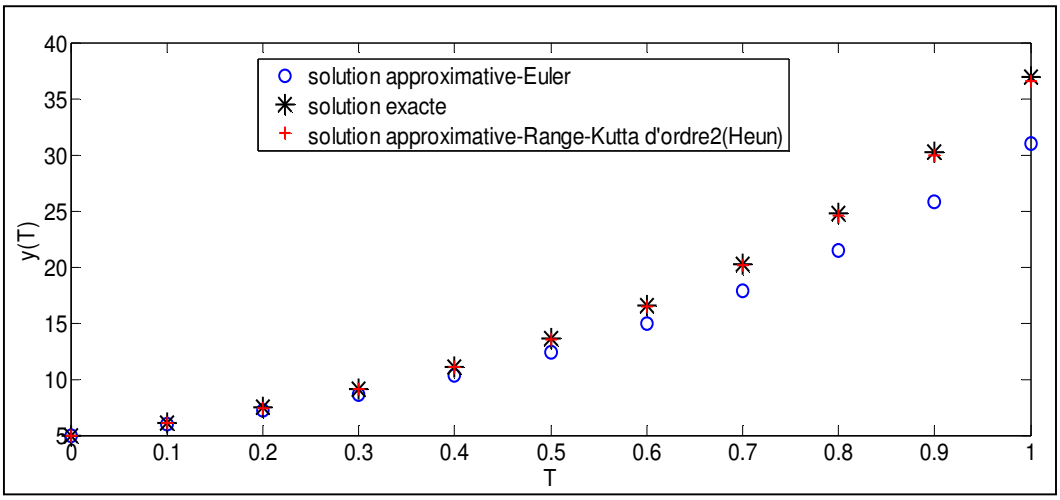


Fig. 2

D'après les courbes illustrées dans la figure (**2**), on constate que la méthode Heun donne les solutions approximatives $y(t_i)$ les mieux proches de la solution exacte. Pour améliorer la précision de la méthode d'Euler, il suffira de réduire le pas Δt et d'augmenter le nombre d'itérations ($t\epsilon$ [0 1]). Les résultats deviennent comme le montre la figure (**3**).

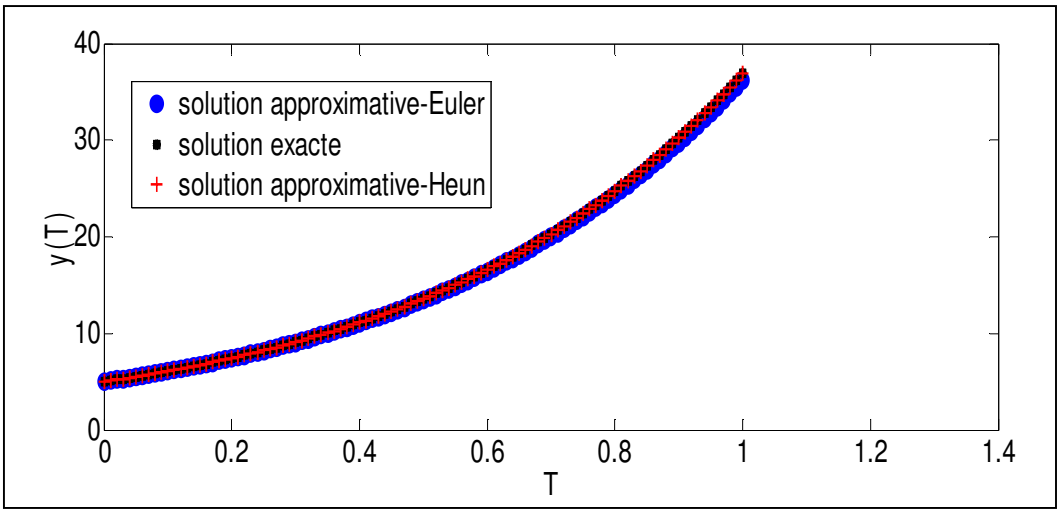


Fig. 3

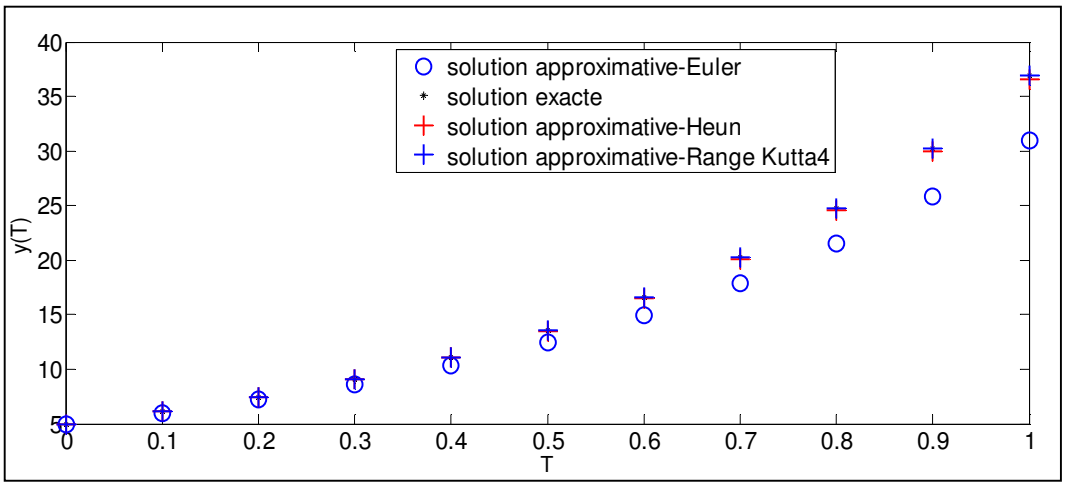


Fig. 4

```
clc, clear all, close all
t=zeros(10,1);y=zeros(10,1); t(1)=0;y(1)=5;T=[0:0.1:1]';
for i=1:10
    t(i+1)=t(i)+0.1;
    f(i)=2*y(i);
    y(i+1)=y(i)+0.1*f(i);
    yr=5*(exp(2*T));
end
[t,yr,y]
plot(T,y,'ob')
xlabel('T')
ylabel('y(T)')
hold on
plot(T,yr,'*k')
t=zeros(10,1);y=zeros(10,1);t(1)=0;y(1)=5;T=[0:0.1:1]';
for i=1:10
```

```
    t(i+1)=t(i)+0.1;
    f(i)=2*y(i);
    y(i+1)=y(i)+0.1*1.1*f(i);
end
[t,y]
figure (1)
plot(t,y,'+r','MarkerSize',2,'LineWidth',6)
t=zeros(10,1) ; y=zeros(10,1);t(1)=0;y(1)=5;
for i=1:10
    t(i+1)=t(i)+0.1;
    f(i)=2*y(i);
    k1=f(i);
    k2=2*(y(i)+k1*0.1/2);
    k3=2*(y(i)+k2*0.1/2);
    k4=2*(y(i)+k3*0.1);
    y(i+1)=y(i)+(0.1/6)*(k1+2*k2+2*k3+k4);
end
y'
figure (1)
plot(t,y,'+b','MarkerSize',2,'LineWidth',6)
legend('solution                approximative-Euler','solution
exacte','solution  approximative-Heun','solution  approximative-
Range Kutta4')
```

D'après le tableau suivant et les courbes illustrées dans les figures 2 et 4 (nombre d'itération 10), on voit clairement que la méthode de Range Kutta d'ordre 4 est la mieux adaptée, puisque elle donne l'approximation la plus proche de la valeur exacte.

Méthode	Euler	Range Kutta2 (formule de Heun)	Range Kutta4	Analytique (solution)
Solution	5.0000 6.0000 7.2000 8.6400 10.3680 12.4416 14.9299 17.9159 21.4991 25.7989 30.9587	5.0000 6.1000 7.4420 9.0792 11.0767 13.5135 16.4865 20.1136 24.5385 29.9370 36.5232	5.0000 6.1070 7.4591 9.1105 11.1276 13.5913 16.6004 20.2757 24.7647 30.2476 36.9444	5.0000 6.1070 7.4591 9.1106 11.1277 13.5914 16.6006 20.2760 24.7652 30.2482 36.9453
Précision				/

Partie (2)

1/

```
clc, clear all, close all
m=5;k=3;h=0.1;tf=15;n=tf/h;t(1)=0;x(1)=2;v(1)=0;
for i=1:n
    t(i+1)=t(i)+h;
    x(i+1)=x(i)+h*v(i);
    v(i+1)=v(i)+h*(-k/m)*x(i);
end
x'
```

2/

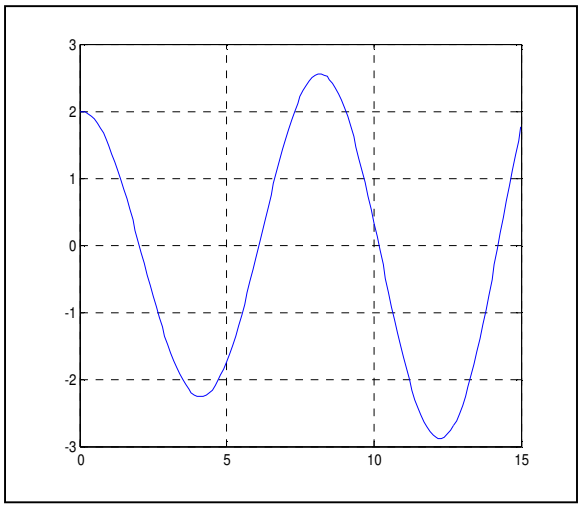


Fig 5 / h=0.1

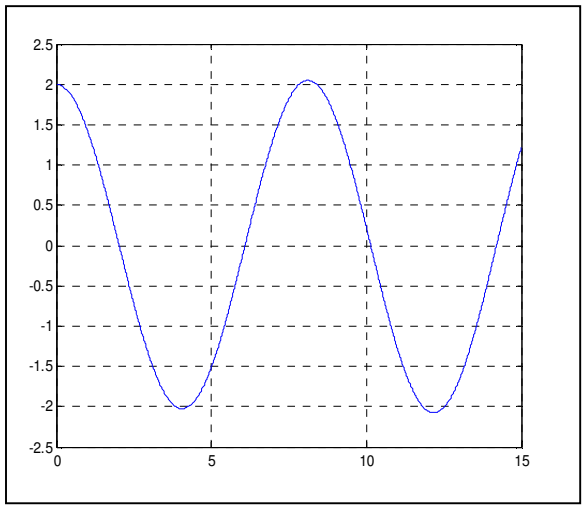


Fig 6 / h=0.01

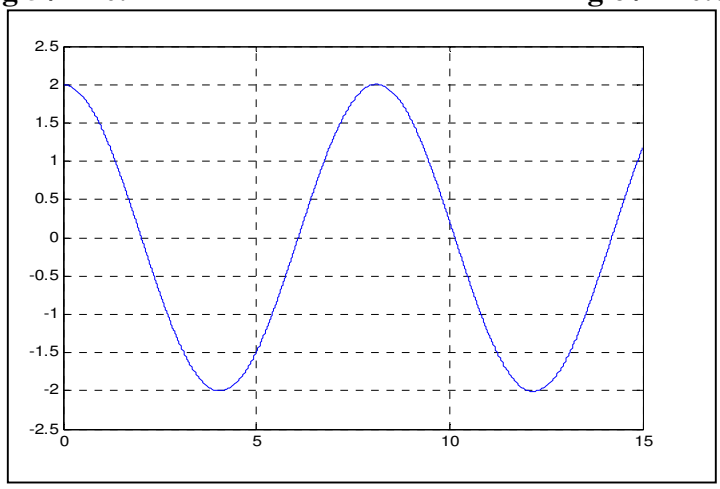


Fig. 7 / h=0.001

Dans l'intervalle $[0,\ 15]$ et pour $\Delta t\ =\ 0.1$ la valeur maximale de $x(t)$ calculée dans les 2 sens (+) et (-) augmente avec l'augmentation de t. A chaque fois on augmente le nombre d'itérations $N = \frac{t_f}{\Delta t}$ (exemple : $\Delta t\ =\ 0.1, \Delta t\ =0.01$ et $\Delta t\ =0.001$), la valeur maximale de $x(t)$ calculée ($x_{max}(t)$), dans les 2 sens, devient de plus en plus constante et par conséquence

la solution approximative d'Euler $x(t)$ s'approche de la solution analytique caractéristique de tout oscillateur harmonique libre non amorti (Fig. 7 et Fig. 10). Cette solution donne l'évolution de la position de masse m en fonction du temps. Elle est de la forme connue : $x(t) = x_{max} cos(\omega_0 t + \varphi)$, x_{max} est l'amplitude du mouvement et φ la phase.

On détermine les deux grandeurs x_{max} et φ à partir des conditions initiales suivantes : $t = 0$ $x = x(0) = 2$ et $\frac{dx(0)}{dt} = 0$.

$$\begin{cases} x(t) = x_{max} cos(\omega_0 t + \varphi) \\ \dfrac{dx(t)}{dt} = -x_{max}\, \omega_0 sins(\omega_0 t + \varphi) \end{cases}$$

A $t = 0$:

$$\begin{cases} x(0) = x_{max} cos(\varphi) = x(0) = 2 \\ \dfrac{dx(0)}{dt} = -x_{max}\, \omega_0 sins(\varphi) = 0 \Rightarrow \varphi = 0 \Rightarrow \begin{cases} x_{max} = x(0) = 2 \\ \varphi = 0 \end{cases} \end{cases}$$

Donc la solution de l'équation différentielle (*) est : $x(t) = 2\, cos(\omega_0 t)$.

Pour tracer les courbes illustrées dans les figures 5, 6 et 7, on exécute le programme suivant, en modifiant à chaque fois le pas $\Delta t = h$:

```
clc, clear all, close all
m=5;k=3;h=0.1;tf=15;n=tf/h;t(1)=0;x(1)=2;v(1)=0;
for i=1:n
    t(i+1)=t(i)+h;
    x(i+1)=x(i)+h*v(i);
    v(i+1)=v(i)+h*(-k/m)*x(i);
end
plot(t,x,'r-')
grid on
xlabel('t')
ylabel('x(t)')
```

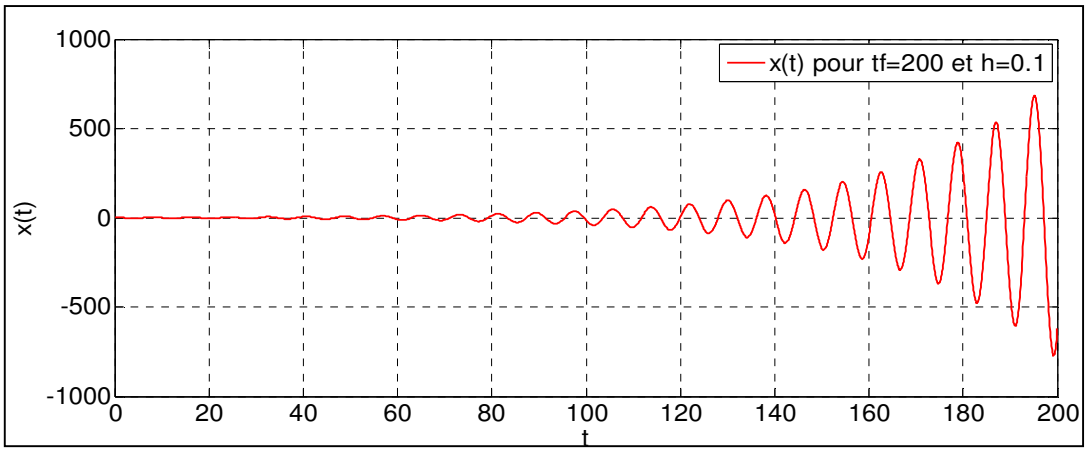


Fig. 8

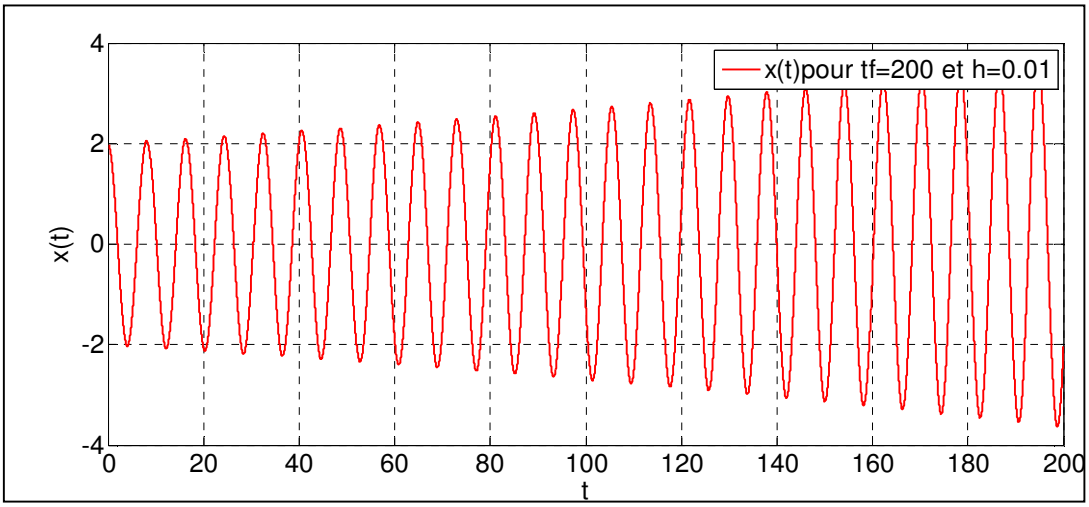


Fig. 9

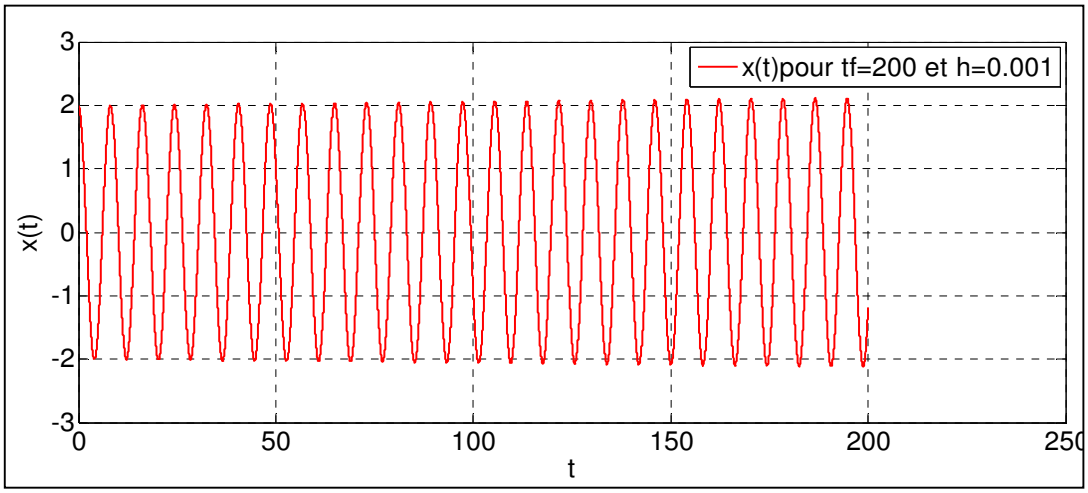


Fig. 10

Les figures 8, 9 et 10 représentées au dessus, sont obtenues à partir du code suivant :

```
clc, clear all, close all
m=5;k=3;h=0.1;tf=200;n=tf/h;t(1)=0;x(1)=2;v(1)=0;
for i=1:n
```

```
    t(i+1)=t(i)+h;
    x(i+1)=x(i)+h*v(i);
    v(i+1)=v(i)+h*(-k/m)*x(i);
end
plot(t,x,'r-')
grid on
xlabel('t')
ylabel('x(t)')
legend('x(t)pour tf=200 et h=0.1')
```

TP Nº 5 Résolution des systèmes d'équations linéaires

1. 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.

2. Introduction au calcul matriciel sous Matlab

2.1. Définition de vecteurs et de matrices

Les commandes de base sont:

- Vecteur colonne : $x = [1;\ 2;\ 3]$
- Vecteur ligne : $z = [1\ -1\ 0]$
- Une matrice : $A = [1\ 2;\ 3\ 1]$

Autres commandes :

$Size(A)$: nombre de lignes et de colonnes de A

$A(i,j)$: coefficient d'ordre i,j de A

$A(i1 : i2,\ :)$: ligne $i1$ à $i2$ de A

$A(\ :, j1 : j2)$: colonnes $j1$ à $j2$ de A

$A(\ :)$: indexation linéaire de A

$A(i)$: coefficient d'ordre i dans l'indexation linéaire

$diag(A)$: coefficients diagonaux de A

$diag(x)$: matrice diagonale dont la diagonale est le vecteur x

$zeros(m,n)$: matrice nulle de taille m,n

>> Introduire les vecteurs et les matrices suivants :

$$x = \begin{bmatrix}1\\2\\3\end{bmatrix}, \qquad y = \begin{bmatrix}1\\2\\3\end{bmatrix}, \qquad Z = [1\ -1\ 0], \qquad u = \begin{bmatrix}3\\2\end{bmatrix},$$

$$A = \begin{bmatrix}1 & 4\\6 & 3\end{bmatrix}, \qquad B = \begin{bmatrix}1 & 3 & 4\\0 & -3 & 1\end{bmatrix}, \qquad C = \begin{bmatrix}1 & -1\\0 & -2\\1 & -3\end{bmatrix}$$

2.2 Opération de base

Quelques opérations de base sur les vecteurs et les matrices :

- Transposition : x', A'
- Somme et différence : $x + y,\ \ x - y$
- Produit : $A * B$
- Puissance : $A\hat{}3$

>> Appliquer ces opérations sur les vecteurs et les matrices ci-dessus. On vérifiera en particulier que $(AB)x = A(Bx)$ et que $(AB)^T = A^T B^T$

2.3 Matrices particulières

Voici quelques exemples de matrices prédéfinies dans Matlab : $eye(n)$, $ones(n)$, $pascal(n)$, $magic(n)$, où n est un entier positif.

>> Calculer $ones(n)\text{^}m$ pour différents n et m. Calculer $ones(3) * magic(3)$. Interpréter les résultats.

>> $A = [1\ 2\ 3\ ;\ 2\ 3\ 1\ ;\ 3\ 1\ 2]$. Créer des matrices particulières.

Exemple de création d'une matrice par blocs :

$C = [A, zeros(3,2)\ ;\ zeros(2,3)\ , eye(2)]$.

3. Rappel sur la méthode de résolution

On prend par exemple la méthode de Gauss comme méthode de résolution numérique de système d'équations linéaires.

Un système d'´equations linéaires à résoudre se présente de la façon suivante :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots \ldots \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots \ldots \ldots + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots \ldots \ldots + a_{3n}x_n = b_3 \\ \qquad\qquad\qquad\qquad \ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots \ldots \ldots + a_{nn}x_n = b_4 \end{cases}$$

La recherche des solutions x_i consiste à reformuler le problème en termes matriciels :

$A \cdot x = b$ avec :

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \ldots \ldots \ldots a_{1n} \\ a_{21} & a_{22} & a_{23} \ldots \ldots \ldots a_{2n} \\ a_{31} & a_{32} & a_{33} \ldots \ldots \ldots a_{3n} \\ & & \ldots\ldots \\ a_{n1} & a_{n2} & a_{n3} \ldots \ldots \ldots a_{nn} \end{pmatrix}$$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{pmatrix} \quad , \quad b = \begin{pmatrix} b_1 \\ b_2 \\ a_3 \\ \dots \\ b_n \end{pmatrix}$$

Il s'agit de formaliser une procédure qui transforme une matrice en une matrice triangulaire supérieure en éliminant, colonne après colonne les éléments non nuls en dessous de la diagonale.

- On commence par le remplissage de la matrice augmentée A du système $\mathrm{A} = (\mathrm{A} \vdots b)$

  Donc la dimension de la matrice A sera de $'n'$ lignes et $'n+1'$ colonnes.

- Ensuite, on programme l'algorithme de Gauss qui va trianguläriser la matrice A (il va transformer A à une matrice triangulaire supérieure).

L'algorithme de Gauss qui va trianguläriser la matrice A se déroule de façons suivante :

Pour k allant de 1 à $n-1$

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

- Et enfin, on extrait la solution du système (par substitution arrière) suivant l'algorithme:

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

4. Exemple

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

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

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

5. Solution

```
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
```

Les résultats sont affichés comme suit :

```
xgauss =

    1.0000
   -1.0000
    1.0000
   -1.0000

xm =

    1.0000
   -1.0000
    1.0000
   -1.0000
```

References bibliographies

[1] Steven T. Karris Third, Numerical Analysis Using Matlab and Excel. Edition 2007.

[2] Jean-pierre Demailly, Analyse numérique et équations différentielles EDP sciences, 2006.

[3] L. Georges Rodriguez-Guisantes, Introduction à Matlab, 2008.

[4] Y. Yang, Wenwu Cao, Tae S.Chung, John Morris, Applied Numerical Methods Using Matlab Won 2005.

[5] L.F. Shampine and I. Gladwell Southern Methodist Solving ODES with Matlab, University Dallas 2004.

[6] Samir Kenouche, Méthode Numériques et Programmation Biskara université 2016/2017

[7] S. Karoui TP Méthodes Numérique, Université d'Oran.

[8] Meddahi Yousous, Méthode Numériques Université de Chlef.

[9] Chabane Foued, Cours de l'Analyse Numérique Appliquée.

[10] PH. Depondt, La boite à Outils de la Physique Numérique, Université Pierre et Marie Curie Paris-6, Année 2008-2009.

[11] Alfio. Quarteroni ; Riccardo. Sacco ; Fausto. Saleri, Méthodes Numériques Algorithmes, analyse et Applications 2007.

[12] Jean-Louis Merrien, Analyse Numérique avec Matlab 2007.

[13] Catherine Bolley, Analyse numérique, 2014.

[14] Jacques le Bourlot, Notions d'analyse numérique, Paris7 – 2006

[15] Hubert Baty, Approche numérique à l'usage du physicien pour résoudre les équations différentielles ordinaires. I. Cas des équations du premier ordre, Strasbourg. 2017

[16] Hubert Baty, Approche numérique à l'usage du physicien pour résoudre les équations différentielles ordinaires. III. Cas d'équations différentielles aux dérivées ordinaires du second ordre en dimension 2. Application aux oscillateurs harmoniques/anharmoniques, et au mouvement Képlérien dans le plan, France 2019.

[17] Eric Walter, Méthodes numériques et optimisation, un guide du consommateur, 2015.

[18] Bahram Houchmandzadeh, Mathématiques pour la Physique. Licence. Mathématiques pour la Physique., France. 2010.

[19] Gloria Faccanoni, Analyse numérique Recueil d'exercices corrigés et aide mémoire, 2013-2014

[20] Bahram Houchmandzadeh, Mathématiques pour la Physique, France, 2010.