Corrigé TP2 MATLAB

NB/ Pour écrire une fonction dans Matlab, la première règle à respecter est de donner au fichier .m le même nom que la fonction que l'on est en train d'écrire. Par exemple, une fonction qui s'appellerait mafact devra ˆetre ´ecrite dans le fichier mafact.m.

Exercice 1

On veut écrire un programme qui calcul la racine carrée d'un nombre entré par l'utilisateur et qui affiche le résultat dans une phrase. Le résultat devra être le suivant :

Entrez un nombre: 23 (23 est entré par l'utilisateur)

La racine carrée de 23 est 4.7958 (sortie à l''ecran)

Indication: on peut convertir les nombres en chaîne de caractères en utilisant la fonction *num2str*.

Corrigé de l'exercice 1

Attention au copier-coller à partir d'un éditeur qui n'est pas l'éditeur de *Matlab*. Il y a une différence entre la quotte de Word ' et la quotte de l'éditeur de Matlab ' . En procédant par copier-coller, on récupère le message erreur suivant

```
??? a = input(‘Entrez un nombre: ‘);
          |
Error: The input character is not valid in MATLAB statements or expressions.
```

qui dit tout simplement que le symbole ' n'est pas valable comme syntaxe dans *Matlab*. Il ne faut donc pas se fier aux apparences des fonts qui peuvent être codés différemment, ça arrive aussi avec 1 (un) et l (lettre l minuscule), ou bien avec 0 et O.

La solution qui correspond à la syntaxe exacte est donc :

```
a = input('Entrez un nombre: ');

b = sqrt(a);

str = ['La racine carrée de ' num2str(a) ' est ' num2str(b)];

disp(str)
```

Exercice 2

Ecrire un M-file function appelé polaire.m qui permet de convertir les coordonnées cartésiennes d'un point en coordonnées polaires.

On sélectionne dans le menu de la fenêtre de commandes File, puis New, puis Function M-File. La fonction considérée ici et permettant de convertir les coordonnées cartésiennes d'un point en coordonnées polaires peut alors être écrite en utilisant les commandes suivantes:

```
function [rayon,theta]=polaire(x,y)
% polaire permet de convertir les coordonnées cartésiennes
% (x,y) en coordonnées polaires r et theta (en radians)
rayon=sqrt(x^2+y^2);
theta=atan(y/x);
end
```

Ici *atan* représente la fonction réciproque de la fonction tangente qui est définie sur $]-\infty,+\infty[$ **et prend ses valeurs dans l'intervalle** $]-\pi/2;\pi/2[$ **. En fait cette fonction est définie comme toute les fonction *Matlab* sur le corps des complexes. A tire d'exemple**

```
>> atan(1+i)
ans =
  1.0172 + 0.4024i
```

Noter bien le nom choisi pour cette fonction qui est en français. Attention toutefois au cas d'un anglophone. Il ne devrait pas l'appeler naïvement polar car *Matlab* dispose déjà d'une fonction intrinsèque qui porte le même nom ! Cette fonction *polar* de *Matlab* admet en sortie un autre ordre pour les coordonnées polaires car elle calcule d'abord theta avant le rayon r. On appellera donc la fonction intrinsèque *polar* de *Matlab* sous la forme [t,r]=polar(x,y) pour obtenir l'angle t et le rayon r.

En appelant notre fonction polaire pour calculer les deux coordonnées polaires d'un point défini par exemple par ses deux coordonnées cartésiennes (0,1), il convient de le faire sous la forme suivante:

```
>> [r,t]=polaire(0,1)
r =
   1
t =
  1.5708
```

implifiée

```
>> polaire(0,1)

ans =

    1
```

permet de récupérer seulement la première sortie c'est-à-dire r=1 (contrairement à la fonction *polar* de *Matlab* qui permet de récupérer sans préciser les sorties la seule theta). La deuxième coordonnée polaire n'a pas été transférée à l'espace de travail comme on peut le vérifier en tapant d'abord *clear*, puis la commande polaire(0,1), et enfin la commande *who*

```
>> who

Your variables are:

ans
```

Noter ici que *Matlab* n'a envoyé aucun message d'erreur concernant la division par 0 dans l'instruction atan(y/x) laquelle dans notre exemple correspond à atan(1/0). En effet, en tapant 1/0 dans l'invite de *Matlab*, on obtient

```
>> 1/0

ans =

  Inf
```

***Matlab* gère ainsi les divisions par zéro au moyen de la variable Inf qui représente l'infini numérique. En tapant -1/0, on obtient**

```
>> -1/0

ans =

 -Inf
```

On peut donc dire que Matlab prend 0+ pour 0. On vérifie par ailleurs que

```
>> atan(Inf)

ans =

   1.5708
```

obtenu avec l'approximation simple précision $\frac{\pi}{2} \approx 1.5708$ **.**

Pour étudier comment se comporte la variable auxiliaire Inf de *Matlab*, on peut taper les commandes suivantes :

```
ans =

    0

>> cos(Inf)

ans =

   NaN
```

L'autre variable auxiliaire qui vient d'apparaître dans le dernier résultat NaN (Not a Number), autrement dit valeur indéterminée, correspond à une indétermination comme celles qui sont rencontrées en calcul des limites. A titre d'exemple on tapera

```
>> 0/0

ans =

   NaN

>> Inf/Inf

ans =

   NaN

>> Inf-Inf

ans =

   NaN

>> 0*Inf

ans =

   NaN
```

On reconnaît dans ces exemples les quatre indéterminations fondamentales en calcul des limites. On comprend maintenant la signification du résultat correspondant à cos(Inf) car la fonction cosinus n'a pas de limite en $+\infty$. Il faut faire attention au fait que *Matlab* ne calcule pas les limites. Maple permet de le faire.

Si l'on tape à présent

```
>> help polaire

 polaire permet de convertir les coordonnées cartésiennes (x,y) en

 coordonnées polaires r et theta (en radians)
```

ıentaire inséré pour présenter ce que fait la fonction polaire tel que nous l'avons renseigné dans les deux lignes suivantes du script

```
% polaire permet de convertir les coordonnées cartésiennes
% (x,y) en coordonnées polaires r et theta (en radians)
```

Matlab **le fait cependant sans le symbole % placé en début de ligne et avec un retour à la deuxième ligne qui n'est pas exactement au même endroit où ceci a été effectué dans le commentaire vert.**

Exercice 3

1. Ecrire un M-file pour la fonction $f(x)=\dfrac{x^5-3}{\sqrt{x^2+1}}$

2. Tester la fonction sur quelques valeurs, par exemple $f(1)=-1.4142$, $f(0)=-3$.

3. Créer un tableau x d'abscisses de -5 à 5 et contenant 100 points.

4. Représenter la fonction f aux points xi, avec la commande plot(x,f(x))

Corrigé de l'exercice 3

1. **On peut proposer le M-file suivant pour définir la fonction f :**

```
function [fdex]=fonction_f(x)
fdex=(x^5-3)/sqrt(x^2+1);
end
```

Nous n'avons placé ici aucun commentaire spécial concernant cette fonction (pas donc de partie en vert).

2. **On peut tester la fonction avec les commandes**

```
>> [fdex]=fonction_f(1)
fdex =
  -1.4142
>> [fdex]=fonction_f(0)
fdex =
   -3
```

fonction. L'appeler tout simplement f est possible, mais il vaut mieux ne pas le faire pour s'éloigner exprès de la notation mathématique. Il ne faut pas l'appeler f(x) comme on peut avoir envie de le faire !

On n'est pas obligé de mettre les crochets autour de fdex, la variable qui sert ici à stocker le résultat de sortie après l'appel de la fonction, et on peut tester la fonction pour 0 avec la commande

```
>> fdex=fonction_f(0)

fdex =

   -3
```

En cas de plus de deux arguments en sortie, on doit placer nécessairement les crochets. La sortie est alors un tableau dont la dimension varie entre 1 et le nombre maximum des variables calculées comme arguments de sortie de la fonction.

3. **Le tableau est crée par la commande suivante**

```
>>x=linspace(-5,5,100) ;
```

Il est utile de placer le symbole point-virgule ; en fin de la commande pour ne pas afficher les 100 termes du tableau x créé par cette commande.

La commande *linspace* permet d'initialiser la variable x et ne modifie pas seulement les valeurs d'un tableau x déjà créé, elle est équivalente à la séquence suivante

```
>> clear x;

>> x=linspace(-5,5,100);
```

4. **Pour faire la représentation graphique de la fonction f aux points xi du tableau x créé dans la question précédente, il faut adapter le M-file qui a servi à la créer. Le M-file de la question 1 définit une fonction qui agit sur un scalaire et qui n'a pas la possibilité d'agir sur un tableau contenant plus de deux termes.**

L'adaptation du M-file est naturelle et est toute simple avec *Matlab*. Il suffit de transformer les opérations arithmétiques qui apparaissent dans le M-file en opérations terme à terme qui ont la possibilité d'opérer sur des tableaux. Voici la transformation pertinente qu'il faut effectuer au niveau du M-file qui définit la fonction f.

```
function [fdex]=fonction_f(x)

fdex=(x.^5-3)./sqrt(x.^2+1);

end
```

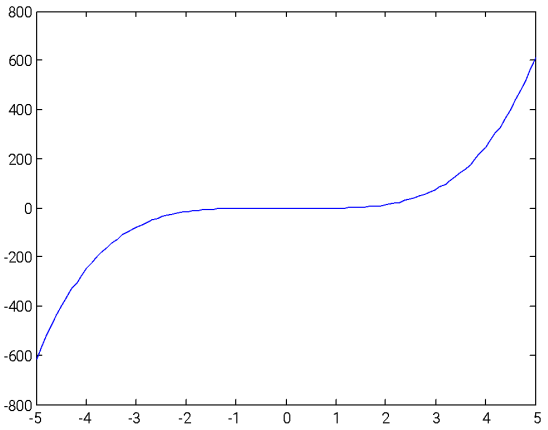
Il y a seulement trois points qui ont été rajoutés, mais à présent la fonction peut calculer l'image d'un tableau sans aucun problème.

ait se faire avec la structure contrôle boucle for, mais ce n'est pas élégant eu égard à l'esprit de *Matlab* !

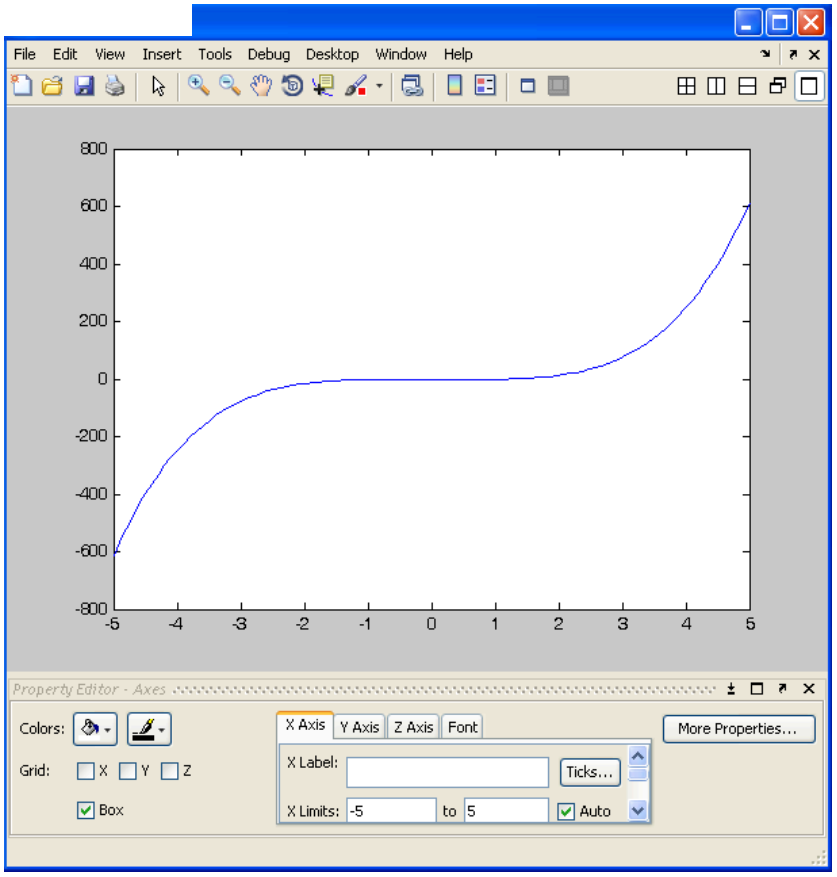
Finalement le tracé de la fonction peut se faire à l'aide des commandes suivantes lesquelles de préférence devraient être écrites dans un script qui appelle la fonction et qui demande à *Matlab* de faire la représentation graphique (on pourra l'appeler trace_de_la_fonction):

```
%trace_de_la_fonction

clear all;

close all;

clc;

x=linspace(-5,5,100);

[y]=fonction_f(x);

plot(x,y)
```

On obtient alors la figure suivante (réduite à 50% par rapport à la figure récupéré sous Word)

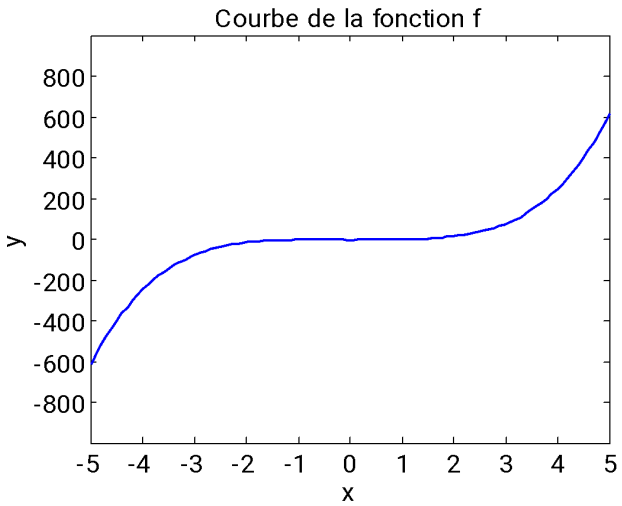


Nous pouvons remarquer que la taille des caractères est petite. En règle générale, la taille doit dépasser 6 points pour être lisible (c'est un standard dans le monde de l'édition). Il est possible de modifier le graphique produit par *Matlab* pour améliorer sa présentation. On peut le faire au niveau du script que nous avons appelé trace_de_la_fonction ou de préférence en choisissant dans la fenêtre graphique le menu Edit, puis Figure Properties. On obtient alors la figure graphique sous l'aspect suivant :



Il suffit de modifier la taille des caractères en l'augmentant de 12 à 16, on peut aussi introduire le label des axes…

Voici un exemple des résultats de modifications opérées à l'aide des menus de la figure précédente



Cette dernière figure semble mieux que la première figure de la page précédente (malgré les avis contraires de certains étudiants qui ont une vue très saine !).

Exercice 4

Ecrire un script pour la fonction : $f(x) = \exp(\sin(x))$ sur l'intervalle $[-\pi/2, \pi/2]$.

1. Calculer $f(0)$, $f(\pi/4)$ et $f(1)$.

2. Tracer la fonction $f(x)$ sur l'intervalle $[-\pi/2, \pi/2]$.

3. Calculer le maximum de $f(x)$ sur l'intervalle $[-\pi/2, \pi/2]$.

Corrigé de l'exercice 4

Nous allons donc donner d'office à la fonction f de cet exercice la possibilité d'agir sur des tableaux en transformant les opérations ordinaire en opérations terme à terme.

Le M-file définissant la fonction f est alors :

```
function [y]=fonction_exo4(x)

y=exp(sin(x));

end
```

Nous n'avons en fait rien fait pour donner à la fonction de cet exercice la faculté d'agir sur des tableaux. Son expression ne faisant appel qu'à des fonctions intrinsèques de *Matlab*, à savoir les fonction sin(x) et exp(x) qui sont déjà dotées de cette faculté (grâce aux programmeurs de Matworks).

2.

>> [y]=fonction_exo4 (0)

y =

1

>> [y]=fonction_exo4 (pi/4)

y =

2.0281

>> [y]=fonction_exo4 (1)

y =

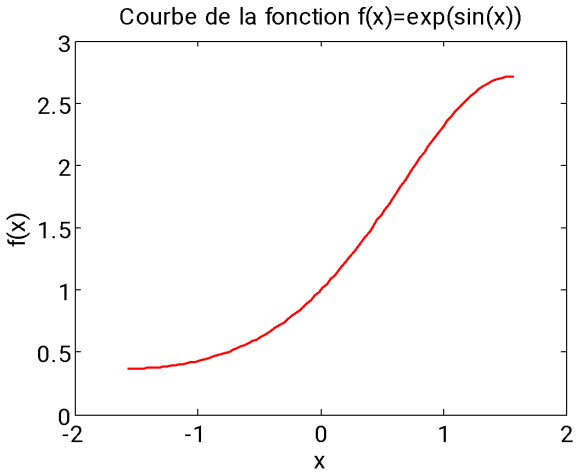
2.3198

3. Le script suivant permet de tracer la courbe de la fonction f sur l'intervalle $[-\pi/2, \pi/2]$.

```
%tracé de la courbe de la fonction f(x)=exp(sin(x))

clear all;

close all;
```

```
x=linspace(-pi/2,pi/2,100);

[y]=fonction_exo4(x);

plot(x,y,'r-')

xlabel('x')

ylabel('f(x)')

title('Courbe de la fonction f(x)=exp(sin(x))')
```

On obtient alors le graphe suivant



3. Pour calculer de manière approchée la valeur maximale prise par la fonction $f(x)$ sur l'intervalle $[-\pi/2,\pi/2]$, il suffit de taper la commande suivante :

>> max(y)

ans =

2.7183

Si l'on veut calculer de manière approchée le maximum de f, c'est-à-dire l'abscisse où la fonction f(x) prend sa valeur maximale, on peut procéder de la façon suivante :

>> [ymax,imax]=max(y)

ymax =

2.7183

imax =

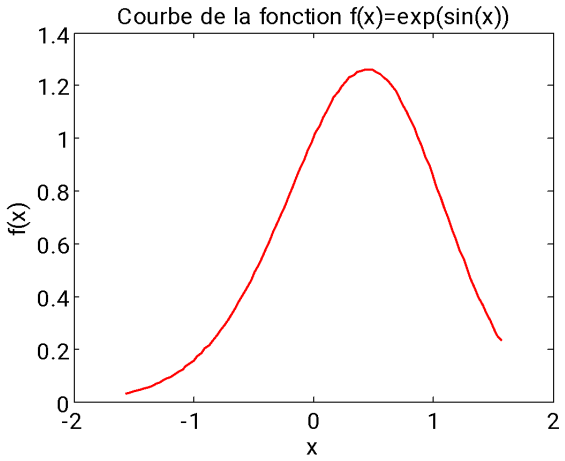
100

>> x(100)

ans =

1.5708

ıum calculé ainsi constitue seulement une approximation du maximum mathématique (maximum exact) de la fonction dont on cherche le maximum. Ce dernier ne peut être obtenu de manière exacte numériquement (sauf dans certains cas particuliers). Dans le cas de la fonction de cet exercice. Le maximum exact est $\pi/2$ car la fonction est strictement croissante sur l'intervalle $[-\pi/2,\pi/2]$. Le maximum numérique obtenu à l'aide de la discrétisation à 100 points est **1.5708** en simple précision ou bien **1.570796326794897** en double précision, et c'est exactement la valeur de $\pi/2$ évaluée en double précision. On n'aurait pas toujours la chance de tomber sur un résultat aussi bon dans le cas général. On peut tester la chose avec par exemple $f(x)=\exp(\sin(x)-x^2)$ et dont la courbe est la suivante.




```
>> [ymax,imax]=max(y)

ymax =

  1.261554787199085

imax =

   65

>> x(65)

ans =

  0.460132257343960
```

Cependant, une discrétisation plus fine à 1000 points, au lieu de 100, produit le résultat suivant

```
>> [ymax,imax]=max(y)

ymax =

  1.261705198823119
```

644

>> x(644)

ans =

0.451269815605741

Le maximum dans le deuxième cas est plus précis que celui obtenu dans le premier cas. Le maximum exact (mathématique) lui n'a pas bougé ! Augmenter le nombre de points ou ce qui revient au même réduire le pas de discrétisation permet d'améliorer la qualité de l'approximation.

Exercice 5

Ecrire un script pour la fonction: $f(x) = e^{-\alpha x^2} \cos\left(\frac{\pi x}{2\alpha}\right)$ sur l'intervalle $[-\alpha,\alpha]$

1. Calculer $f(0)$, $f(\alpha)$, $f(-\alpha)$, $f(-5)$ et $f(5)$.

2. Tracer la fonction $f(x)$ sur l'intervalle $[-\alpha,\alpha]$ en utilisant la commande *plot*

Corrigé de l'exercice 5

1.

Pour définir la fonction $f(x) = e^{-\alpha x^2} \cos\left(\frac{\pi x}{2\alpha}\right)$, on peut proposer le M-file suivant :

```
function [y]=fonction_exo5(x,alpha)

y=exp(-alpha*x.^2).*cos(pi*x/2/alpha);

end
```

Noter la présence du paramètre alpha en tant que deuxième argument de la fonction. Il faut noter aussi que les opérations terme à terme ont été introduites là où elles sont nécessaires pour donner la possibilité à la fonction de calculer en bloc les images d'un tableau contenant plusieurs antécédents.

2. Bien sûr le paramètre α doit être spécifié en indiquant sa valeur pour calculer par exemple les images suivantes: $f(0)$ et $f(\alpha)$. Si l'on force Matlab à calculer l'image de 0, on récupère le message erreur suivant :

>> [y]=fonction_exo5(0,alpha)

??? Error using ==> alpha

Too many output arguments.

Par contre, en procédant de la façon suivante pour $\alpha=1$ par exemple

>> alpha=1; [y]=fonction_exo5(0,alpha)

y =

1

ça marche. Tenter de calculer l'image sans spécifier la valeur du paramètre α revient à requérir de *Matlab* d'effectuer un calcul formel (dit aussi symbolique) comme on le fait à la main sous forme littérale en maths. *Matlab* permet de faire le calcul symbolique, mais on ne le verra pas dans ce cours qui est réservé aux facultés de *Matlab* dans le domaine du calcul numérique. Le logiciel Maple est lui spécialisé en calcul formel et il faut le dire, il est dans ce domaine beaucoup plus performant que *Matlab*.

3.

>> [y]=fonction_exo5(0,alpha)

y =

1

>> [y]=fonction_exo5(alpha,alpha)

y =

2.252611900513619e-017

>> [y]=fonction_exo5(-alpha,alpha)

y =

2.252611900513619e-017

>> [y]=fonction_exo5(-5,alpha)

y =

4.251956500241578e-027

>> [y]=fonction_exo5(5,alpha)

y =

4.251956500241578e-027

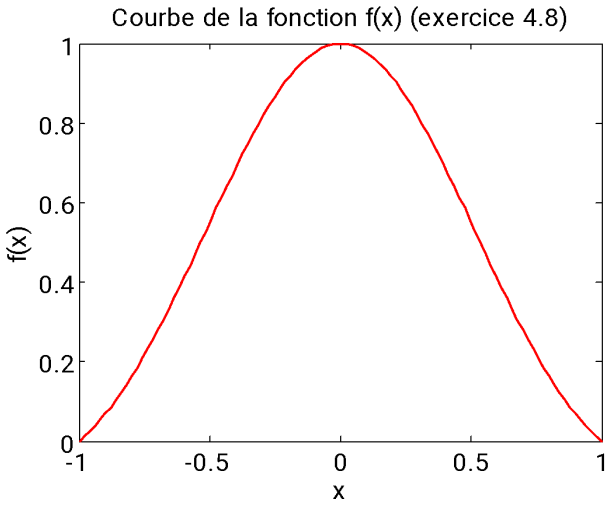
Les valeurs obtenues sont petites à cause de l'exponentielle et on vérifie qu'il y a symétrie par rapport à l'axe $y=0$ car la fonction est paire.

4. Pour tracer la courbe de la fonction $f(x)$ sur l'intervalle $[-\alpha,\alpha]$, on peut utiliser le script suivant :

```
%tracé de la courbe de la fonction de l'exercice 4.8

clear all;

close all;

clc;

alpha=1;
```

```
                        ha,100);
[y]=fonction_exo5(x,alpha);
plot(x,y,'r-')
xlabel('x')
ylabel('f(x)')
title('Courbe de la fonction f(x) (exercice 5)')
```

On obtient alors la courbe de la figure suivante pour $\alpha=1$ **:**



Pour obtenir la courbe qui correspond à $\alpha=5$ **, il suffit de modifier la ligne en surbrillance jaune** `alpha=1;` **et la remplacer avec** `alpha=5.` **On obtient alors la courbe suivante**