TP 01 : Prise en Main

Exercice 01 : (Quelques commandes Matlab)

Commençant par tester les commandes suivantes :

% pour commencer dans un environnement propre
clear all % supprime toutes les variables de la mémoire
close all % ferme toutes les fenêtres graphiques
clc % nettoie la fenêtre de commande
% pour un affichage plus lisible par la suite
% ne change rien au stockage des variables
format short g

- **clock** : affiche l'année, le mois, le jour, l'heure, les minutes et les secondes.
- **date** : Affiche la date.
- **ans** : quand on introduise des instructions anonymes (sans variables en sortie), le matlab considère une variable 'ans' par défaut pour enregistrer le résultat.
- **input** : permet de lire une valeur à partir du clavier (l'instruction habituelle lire) Exemple :x = input ('taper un nombre : ')
- **disp** : permet d'afficher un tableau de valeurs numériques ou de caractères. L'autre façon d'afficher un tableau est de taper son nom. La commande 'disp' se contente d'afficher le tableau sans écrire le nom de la variable, ce qui peut améliorer certaines présentations. On utilise fréquemment la commande disp avec un tableau qui est une chaîne de caractères pour afficher un message. Exemple :>> disp('la valeurs saisie est erronée').
- **clear** : permet de détruire une variable de l'espace de travail (si aucune n'est spécifiée, toutes les variables seront effacées).
- **who** : donne la liste des variables définies dans l'espace de travail actuel (essayer whos).
- **whos** : donne la liste des variables définies dans l'espace de travail avec plus de détails.
- **Help** : on utilise cette commande pour obtenir l'aide sur une méthode donnée.

Exercice 02 :

Ouvrez l'´editeur matlab ("File- New- M-file", ou cliquez sur la page blanche de la barre d'outil).
Créez le script matlab suivant

```
clc , clear all ;
% Ceci est un script matlab,
% le signe "pourcent" permet de mettre des commentaires
% qui ne seront pas interpretes disp('Hello World !')
 %disp permet d'afficher ce que l'on veut a l'ecran,
%les simples cotes ' indiquent
%qu'on veut afficher du texte.
disp('Une 1ere affectation pour a et b')
a = 6
b = 7
c = a;
```

```
a = b;
b = c;
disp('Les nouvelles valeurs de a et b')
a, b %la virgule permet de mettre sur une seule ligne
%plusieurs commandes
%elle joue le role de s´eparateur d'instructions,
%comme la touche entree.
```

Enregistrez le sous le nom voulu (sans accent et sans espace), puis appelez le dans l'interpréteur matlab.

Afin de déclarer une fonction sous Matlab, on a deux méthode possible :

1. Commande fonction : Son lexique est comme suit, en considérant ses entrées et sorties :

Function [out1, out2, ...] = myfun(in1, in2, ...)

Exemple:

```
function E=EstimGauss(t)
 E=t.^(-0.5)*exp(-t/2)/sqrt(2*pi)
 end
```

En deuxième lieu, on enregistre cette fonction dans un _chier M-fil le nomme, par exemple, EstimGauss.m et pour l'utiliser, on fait appel a cette fonction par son nom:

```
  >> t=4
 t =
 4
 >> EstimGauss(t)
 E =
 0.0270
```

Commande inline : Son lexique est comme suit : INLINE (EXPR, ARG1, ARG2, ...), Ou EXPR représente la fonction à déclarer alors que ARG1, ARG2 représentent les variables ou les entrées.

Exemple :

```
f=inline('sin(x)-log(x)-sqrt(x)')
%evaluation de la fonction f avec des valeurs compris entre 1 et 10
for x=1:10
fprintf( 'f(%f)=%f\n' ,x , f ( x ) )
end
```

Exercice 03 :

Considérons le nombre complexe suivant : z1= 1+2i et z2=3-5j

```
1) z1+z2=4.0000 -3.000i
2) z1-z2=-2.0000+7.000i
3) z1/z2=-0.2.059+0.3235i
4) z1*z2=13.000+1.0000i
```

2- Calculez z2 à la puissance 2, `z2^2=-16.0000-30.0000i`

Exercice 04:

Considérons le nombre complexe suivant :

z=[1+j 2-3j ; 4+2j 5-4j]

Donnez la partie réelle, la partie imaginaire, le conjugué et le module de z sous matlab produit de z par son conjugué produit scalaire de z par son conjugué.

Exercice 05 :

Donnez le résultat MATLAB pour chacune des commandes suivantes :

```
>> a=8 ; b=a+6 ; c=b-9 ; clear a, who Your variables are:
>> temp=35.48;poids=28.63;floor(temp),ceil(poids);round(poids)
>> Format rat, sin (pi/5)
```

Exercice 06 :

Donnez des commandes **`function`** permettant d'évaluer les expressions suivantes :

```
a) >> x=2; -x^3-(2/3)*x^2+x^9-4
b) >> x=exp(7); (x^2*sin(9*pi/11)^2)/cos(5*pi+7)
c) >> x=2i; -9*log(8*x+9)+sqrt(5*x^5-6)
```

Exercice 07 :

Donner la suite de commandes **`function`** pour calculer les formules suivantes : $V = 4/3\pi R^3$

*O*ù $R = 22.5cm$.

Exercice 08 :

Soit à résoudre l'équation

$x^2 - 2x = 1$

1. Utiliser la fonction *fzero* pour trouver la première solution au voisinage de x1 = 1, puis utiliser *fzero* une nouvelle fois pour trouver la deuxième racine au voisinage de x2 = -3.
2. Utiliser la fonction *fsolve* pour trouver simultanément les deux racines sur l'intervalle [-3, 1].
3. Utiliser la fonction *solve* pour trouver les racines de cette équation.

TP 02 : Vecteurs et Matrices

Exercice 1 :

Soit la série de nombres [17 8 12 15 6 11 9 18 16 10 13 19].

1. Entrer ces valeurs dans le vecteur x ;

x= [17 8 12 15 6 11 9 18 16 10 13 19]

2. Calculer la longueur N de ce vecteur ;

3. Calculer la somme S des éléments ;

4. Calculer la moyenne $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$

5. Calculer l'écart-type $\sigma = \sqrt{\sum_{i=1}^{N} \frac{1}{N-1}(x_i - \bar{x})^2}$.

6. Calculer le vecteur dx = $\{x_{i+1} - x_i\}$ pour i = 1; 2; :::; N -1.

Exercice 02 :

Soit les trois matrices **A**, **B** et **C** : $A = \begin{pmatrix} 2 & 3 \\ 8 & 2 \end{pmatrix}, B = \begin{pmatrix} 3 & -1 \\ 0 & 5 \end{pmatrix}, C = \begin{pmatrix} -4 & -3 \\ 0 & 1 \\ 1 & 1 \\ -3 & -6 \end{pmatrix}$.

1) Calculez les expressions suivantes :

- **A*B-6**
- **C*B+1-zeros(4,2)**
- **C(end:-1:1,2).\22**

2) Créez la matrice **M** qui contient les matrices **A** et **B** l'une sur l'autre pour définir la 1ère et la 2ème colonne, et la matrice **C** pour définir la 3ème et la 4ème colonne(`M = [[A; B] C])`

$$M = \begin{pmatrix} 2 & 3 & -4 & -3 \\ 8 & 2 & 0 & 1 \\ 3 & -1 & 1 & 1 \\ 0 & 5 & -3 & -6 \end{pmatrix}$$

3) Donnez le résultat MATLAB pour chacune des commandes suivantes :

- **M(3,2) = 3**
- **M([2,3],:) = [ ]**
- **T=tril(M,-1)+triu(M,2)**

Exercice 03 :

Soient les vecteurs colonnes et la matrice suivants

$$\vec{V_1} = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}, \vec{V_2} = \begin{pmatrix} -2 \\ 1 \\ 6 \end{pmatrix}, \vec{V_3} = \begin{pmatrix} -1 \\ 2 \\ -3 \end{pmatrix}, A = \begin{pmatrix} 1 & 2 & 4 \\ 0 & 5 & 4 \\ -6 & 2 & 8 \end{pmatrix}.$$

1. Structures Matlab

1.1 Entrer ces données sous Matlab.

1.2 Calculer $\vec{V_1} + 4\vec{V_2} - \frac{\vec{V_3}}{8}$.

1.3 Calculer le produit scalaire entre les vecteurs$\vec{V}_1 et \vec{V}_3$.

1.4 Calculer le produit A$\vec{V}_1$.

2. Commandes Matlab

Trouver les commandes Matlab permettant de :

2.1 calculer$\|\vec{V}_1\|_2, \|\vec{V}_2\|_1, \|\vec{V}_3\|_\infty$.

2.2 déterminer les dimensions de la matrice A, en extraire le nombre de colonnes ;

2.3 calculer le déterminant et l'inverse de A.

3. Résolution de systèmes linéaires proposer deux méthodes permettant de résoudre le problème $A\vec{x} = \vec{V}_1$, et déterminer les commandes Matlab associées.

Exercice 04:

Soit la matrice carrée A :

$$A = \begin{pmatrix} 10 & 10 & 0 \\ 0 & 10 & 10 \\ 10 & 19 & 10 \end{pmatrix}$$

1. Créer une matrice A.
2. Calculez le determinant.
3. Trouver la matrice inverse de A.
4. Générer un vecteur colonne t qui va de 1 à 10 par pas de 0; 5.
5. Extraire la première ligne.
6. Extraire la deuxième colonne.
7. Extraire la diagonale.
8. Extraire le bloc contenant la deuxième et la troisième ligne avec la première et la deuxième colonne.

Exercice 05:

Soient la matrice et les vectrices colonnes suivantes

$$\vec{V}_1 = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}, \vec{b}_2 = \begin{pmatrix} -2 \\ 1 \\ 6 \end{pmatrix}, \qquad A = \begin{pmatrix} 1/2 & 2/3 & 4/5 \\ 5/4 & 5/2 & 4/3 \\ -6/7 & 2/7 & 8/7 \end{pmatrix}.$$

On définit, pour n ≥1, la suite de vecteurs $\vec{v}_{n+1} = A\vec{v}_n + \vec{b}_2$.

1. Construire une fonction suite .m calculant les premiers termes de la suite$\vec{v}_n$. Cette fonction aura comme arguments d'entrée les données suivantes : la matrice A, le second membre$\vec{b}_2$, le terme initial$\vec{V}_1$, et le nombre de termes voulus nb_{it}.
2. Représenter graphiquement l'´evolution de chacune des composantes. Qu'observe-t-on ?

Exercice 06 : Écrire un script MATLAB qui permet de calculer les éléments de la matrice C, la somme de deux matrices A et B de dimensions 1*3 chacune.

Exercice 07 : Écrire un programme Matlab qui déterminer :

1. le max et sa position de chaque linge
2. le min et sa position de chaque collons

TP 03 : Éléments de programmation

Exercice 01 : ***Les scripts***

Afin de pouvoir réutiliser les lignes de calcul, il est utile de les mettre dans un *script*. Un script est un fichier texte que Matlab pourra lire et exécuter.

1. Ouvrez l'éditeur de scripts de Matlab soit en cliquant sur la page blanche de la barre d'outils, soit en allant dans le menu "File!New!M-file".

Créez le script suivant :

```
% Ceci est un script matlab,
% le signe "pourcent" permet de mettre des commentaires
% qui ne seront pas interpretes
disp('Salut') % disp permet d'afficher ce que l'on veut à
l'écran, les '
% permettent d'indiquer que l'on veut afficher du texte.
a = input('entrez a : ') % input demande à l'utilisateur
d'entrer une
Si vous y arrivez bravo... ça sert à pas grand chose de
le faire
b = 6
b=b+a
a, b % la virgule permet de mettre plusieurs commandes
sur une seule
% ligne, elle a le meme role que la touche entrée.
```

Enregistrez le fichier et appelez le dans l'interpréteur.

2. Écrivez un programme qui demande deux valeurs a et b à l'utilisateur et qui les affiche, qui intervertit leurs contenus et qui les affiche à nouveau.

Exercice 02 : **INSTRUCTION DE CONTROLE (TEST)**

L'instruction **if** exécute un ensemble de commande si une condition est satisfaite.

La syntaxe générale est ***if****(expression ou condition logique)*

instruction

end

If *(expression ou condition logique)*

instruction 1

else

instruction 2

end

Calculer la valeur de y si x= -5

$$y = \begin{cases} \exp(x) + \sin(2x) & ; x < 1 \\ 3x^2 - \ln(x) & ; x \geq 1 \end{cases}$$

Exercice 03: **INSTRUCTION DE REPETITION (BOUCLES)**

On peut répéter des actions grâce aux boucles : la boucle for permet de changer la valeur d'une variable de manière régulière. La syntaxe pour la boucle for est la suivante :

```
for i=1: n
disp(i);
end
while(expression ou condition logique )
Instruction
end
```

Le code entre le for et le end est exécuté n fois : une première fois avec la variable i à 1, une deuxième fois avec la variable i à 2, etc jusqu'à n.

1. Écrivez un programme qui demande deux entiers *a* et *b* et qui affiche le résultat de la somme suivante $\sum_{K=1}^{b} K^a$

Exercice 04 :

En utilisant la boucle while ou la boucle for, écrire un programme en Matlab qui:

1. Calculer la somme : $\sum_{K=1}^{N} K = 1 + 2 + 3 + \cdots . N$.
2. Calculer la factorielle de N $N!$ = 1*2*3*....*N

Exercice 05: ***« chotomie ! »***

Le jeu *dichotomie* est un jeu qui se joue à deux joueurs (et qui n'est pas très amusant en fait). Le premier joueur choisit un nombre entre 1 et 100, et le second joueur essaie de deviner ce nombre. À chaque essai du second joueur, le premier indique si le nombre est plus grand ou plus petit que le nombre à deviner (ou égal, et dans ce cas le jeu est terminé).

1. Écrivez un script qui permet de jouer à *dichotomie* contre l'ordinateur (le script choisit un nombre, et le joueur essaie de le deviner).

Indications :

- Utilisez la fonction randi(n) qui renvoie un nombre entier tiré aléatoirement entre 1 et *n* pour choisir le nombre ;
- Utilisez l'instruction a = input('Entrez un nombre : ') qui demande à l'utilisateur d'entrer un nombre et enregistre la valeur dans la variable a.

Exercice 06 :

Écrire un script Matlab où l'on définit le vecteur N = 1 :2 :12, puis on calcule le factoriel (le factoriel de l'entier k est 1*2…*k) de chacun des éléments de N à l'aide de trois méthodes :

1. en utilisant la fonction built-in *factorial* de Matlab,
2. en utilisant la boucle *for*,
3. en utilisant la boucle *while*.

A l'aide des fonctions *tic* et *toc*, calculer les temps d'exécution de chacune de ces méthodes. Comparez dans un même graphe ces temps.

Exercice 07 :

Nous avons déjà vu comment utiliser une boucle for pour répéter plusieurs instructions. Cependant il fallait connaître à l'avance le nombre de fois que l'on voulait répéter la boucle.
Le mot-clé while sert à exécuter une suite d'instructions *jusqu'à* ce qu'une ce qu'une condition soit vérifiée.

1. Exécutez le script suivant :

```
x = 1;
while x < 1000
x
x = 2*x;
end
```

Que fait-il ? Avez-vous compris la signification du mot-clé while ?

2.Écrivez une fonction cube(n) qui renvoie le plus grand cube inférieur ou égal à *n*.

3. Écrivez une fonction somme_carres(n) qui renvoie le plus grand entier *k* vérifiant

$$\sum_{i=1}^{K} i^2 \leq n$$

Exercice 08 :

Exponentielle réelle négative Programmer en Matlab une fonction qui prend en entrée un réel t et un entier N et qui retourne la somme partielle

$$\sum_{K=0}^{N} x^K / k!$$

Tester votre fonction avec t = 14 et diverses valeurs de N.

TP04 : Polynômes

Exercice 01 :

En utilisant les commandes Matlab : définie le polynôme $P(x) = 3x^4 - 2x^2 + x$:

1. Calculer P(1), P'(3.5) et P''(0). (polyval et polyder)
2. Définie le vecteur V qui contient 100 valeurs compris entre 0 et 2 (linspace).
3. Évaluer le polynôme P(x) sur les points de V.
4. Tracer la courbe du polynôme P dans l'intervalle [1,3]. (plot)
5. Soit le polynôme$S(x) = x^4 - x^3 + 1$, calculer la somme de S et P.

Exercice 02 :

Dans cette exercice, on a choisi quelque problème mathématique résolu directement en langage Matlab sans faire discrétisation numérique comme :

1. équation non linéaire analytiquement :$f(x) = x^2 + 5x + 6$
2. calcule la dérive : $diff(f, x) = \frac{df}{dx} = d/dx(1/(x^2 + 2x + 2))$
3. calcule $\int f(x)dx = \int (1/(x^2 + 2x + 2))\, dx$
4. résoudre équation différentielle du 1^{ere} ordre : $\frac{dy}{dx} = F(x, y) = (x^2 - 1).y$
5. résoudre équation différentielle du 2^{eme} ordre $\frac{d^2y}{dx^2} - 5\frac{dy}{dx} + 6y = x^2$

Exercice 03 :

Dans cet exercice, nous allons voir comment utiliser *Matlab* pour représenter et manipuler des polynômes à coefficients réels ou complexes.

1. En considérant les polynômes comme des suites presques nulles de coefficients, comment proposez vous de les représenter simplement en *Matlab* ?

Il suffit de les représenter par des vecteurs : en effet, un polynôme étant déterminé par ses coefficients, il suffit de stocker ceux-ci. La méthode la plus intelligente consiste à stocker le coefficient du monôme de degré 0 dans la première coordonnée du vecteur et le coefficient du monôme de plus haut degré dans la dernière. On décide ici d'utiliser des vecteurs lignes, mais cela ne change pratiquement rien, sinon des détails d'implémentation.

2. Ecrivez une fonction qui, étant donné un polynôme *P* sous la forme décidée précédemment, renvoie son degré.

3. Ecrivez une fonction qui, étant donné un polynôme *P* et un entier *k*, renvoie le coefficient de *Xk* dans *P*.

4. Écrivez une fonction poly_somme(P, Q) qui renvoie la somme des polynômes *P* et *Q*.

Indication : Si les représentations de *P* et *Q* ont la même taille, la somme est très facile à faire. Si les dimensions ne coïncident pas, vous pouvez utiliser la fonction zeros(x, y) et la notation [tab1, tab2]. Ou alors vous pouvez aussi faire des boucles for...

5. Ecrivez une fonction poly_mult(P, Q) qui renvoie la multiplication des polynômes *P* et *Q* (c'est un peu plus difficile que la somme, et vous n'échapperez probablement pas aux boucles for).

6. Écrivez une fonction poly_exp(P, n) qui renvoie l'exponentiation P^n du polynôme *P*.

TP05 : Graphisme en Matlab

Exercice 01 :

Nous allons maintenant voir comment on peut utiliser *Matlab* pour représenter graphiquement des fonctions, parce qu'au fond, faire des calculs sur des matrices ça va 5 minutes mais faire des dessins c'est plus marrant.

La commande de base permettant de dessiner des graphes est plot. Sous sa forme la plus simple, plot permet de représenter les valeurs d'un tableau :

```
>> plot([2, 3, 5, 9]);
>> plot([2; 3; 2; 3]);
```

Si l'argument de plot est une matrice à plusieurs lignes, *Matlab* trace chacune des colonnes séparément:

```
>> m = [2, 3; 2, 4; 2, 3]
>> plot(m)
```

On peut également modifier l'apparence du tracé en ajoutant un argument final composé de symboles cryptiques entre guillemets simples :

```
>> plot([2, 2, 2, 3, 5, 9, 5], 'r+--')
>> plot([3, 4, 5, 6, 4, 2], 'md:')
```

La commande help plot donne plus de détails, en particulier la liste des codes permettant de choisir la couleur du tracé, la forme des marqueurs des points (croix, cercle, losange, ...) et le type de tracé (pointillé, plein, tirets, etc.).

On peut également (et c'est ce que l'on va utiliser le plus souvent par la suite) représenter les valeurs d'un tableau y en fonction d'un tableau x :

```
>> x = 0:.1:1
>> y = x.^3
>> plot(x, y)
```

1. Dans l'exemple précédent, que fait la ligne y = x.^3 ? Quelle différence y a-t-il entre x^3 et x.^3 ?

Réponse : Par défaut *Matlab* considère que tout est une matrice. Donc l'opérateur d'exponentiation ^ est l'exponentiation matricielle (qui ne fonctionne que sur des matrices carrées). Pour préciser que l'on veut utiliser l'exponentiation sur les réels (et donc l'appliquer individuellement à chaque case du tableau) il faut utiliser la notation .^.

Le tableau des abscisses (premier des deux arguments) ne doit pas nécessairement être régulier, ni même croissant. On peut par exemple utiliser plot pour tracer des courbes paramétriques... La *cycloïde* est la courbe que dessine un point sur la circonférence d'une roue de vélo lorsque celui-ci (le vélo) avance (figure 1).

Au temps *t*, l'ordonnée du point est *sin*(*-t*) (on prend *-t* parce que la roue tourne dans le sens inverse trigonométrique) et son abscisse est *cos*(*-t*) + *t* (parce que le centre de la roue a avancé de *t*).

2. Définissez un tableau t contenant les valeurs entre 0 et 20 espacées de 0, 1 (ce sera l'ensemble des temps auxquels on veut dessiner un point).

Réponse :

```
>> t = 0:.1:20;
```

3. Définissez les tableaux x et y contenant les abscisses et les ordonnées des points de la cycloïde pour chacun des temps dans le tableau t (un peu comme quand on a écrit y = x.^3 plus haut).
Réponse :

```
>> x = t + cos(-t);
>> y = sin(-t);
```

4. Tracez la cycloïde à l'aide de la commande

```
>> plot(x, y, 'o-')
```

Le problème que l'on peut remarquer sur la figure précédente est que les axes ne sont pas orthonormés, ce qui déforme le tracé de la fonction. Par défaut *Matlab* choisit les unités sur les axes de telle sorte que l'ensemble du tracé demandé tienne dans un rectangle prédéfini, mais il est possible de modifier manuellement les axes à l'aide de la commande axis.

5. Exécutez les instructions suivantes et observez l'effet qu'elles ont sur la figure précédemment tracée :

```
>> axis equal
>> axis square
>> axis normal
>> axis([2 8 0 1])
```

Vous aurez probablement remarqué qu'à chaque fois qu'on trace une nouvelle courbe, la précédente disparaît. C'est passablement gênant quand on essaie de comparer plusieurs fonctions et que l'on vou- drait donc les tracer ensemble sur la même figure. Il est cependant possible de superposer des courbes de différentes manières (au moins deux à ma connaissance). L'instruction hold on (qui s'annule à l'aide de la commande hold off) indique que les nouvelles fonctions à tracer doivent être dessinées avec les précédentes :

```
>> x = -pi:pi/20:pi;
>> y1 = cos(x);
>> y2 = sin(x);
>> y3 = -cos(x);
>> plot(x, y1, 'b')
>> hold on
>> plot(x, y2, 'g')
>> plot(x, y3, 'r')
>> hold off
```

Sinon, on peut aussi donner toutes les courbes à tracer en même temps à la fonction plot :

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

Voici quelques autres fonctions qui peuvent servir 1 :

- title : titre du graphe
- xlabel, ylabel et zlabel : étiquette sur les axes
- legend : légende (pas le film avec Tom Cruise...)
- text : permet de placer du texte

Exercice 02 :

Écrire le programme Matlab qui permet de tracer sur le même graphe les fonctions F1, F2 et F3 sur l'intervalle [-2 2].

f1=xsin(1/x), f2= x^3, f3= $2x^2$+3-x+5.

Avec comme titre ' Étude comparative des courbes '.

Exercice 03 :

Écrire un programme qui trace plusieurs courbes un seul variable En utilisant la commande **Holde on.**

Exercice 04 :

Écrire un programme qui trace les courbes à deux variables $z = f(x, y) = x^2 \mp y^2$.

Exercice 05 :

1. Dessiner la courbe 3D suivante : x = cos(t); y = sin(t); z = 4t;
2. Dessiner la surface de la fonction à deux variables : $Z = \frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$

Tp06 : Calcul symbolique

Exercice 01 :

Résolution numérique avec MATLAB de l'équation de la thermocline (océan pacifique)

$$\frac{dh(t)}{dt} = -\tanh\left[\kappa\, h(t-\tau)\right] + b\,\cos(2\pi\, t),\ t \geq 0,$$
$$h(t) = 1 \qquad \text{for } t \in [-\tau,\ 0), \quad \phi(t) \in X.$$

k=100; b=1; $\tau = 0.01$

avec t variant de 0 à 10