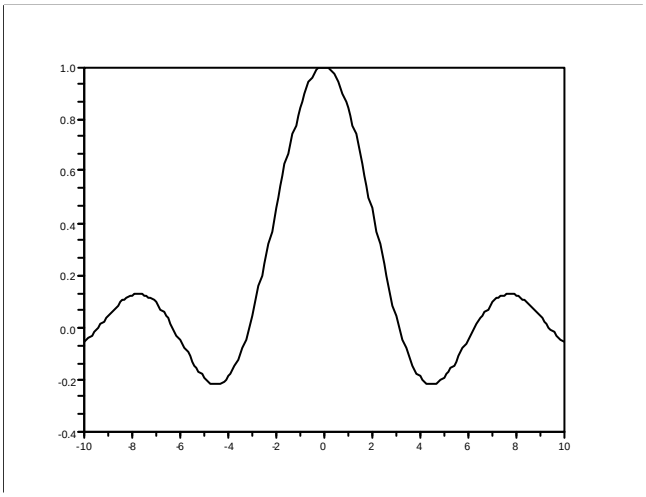
Mathématiques du Signal

Travaux Pratiques

avec

Scilab



Roger NOËL 2006-2007

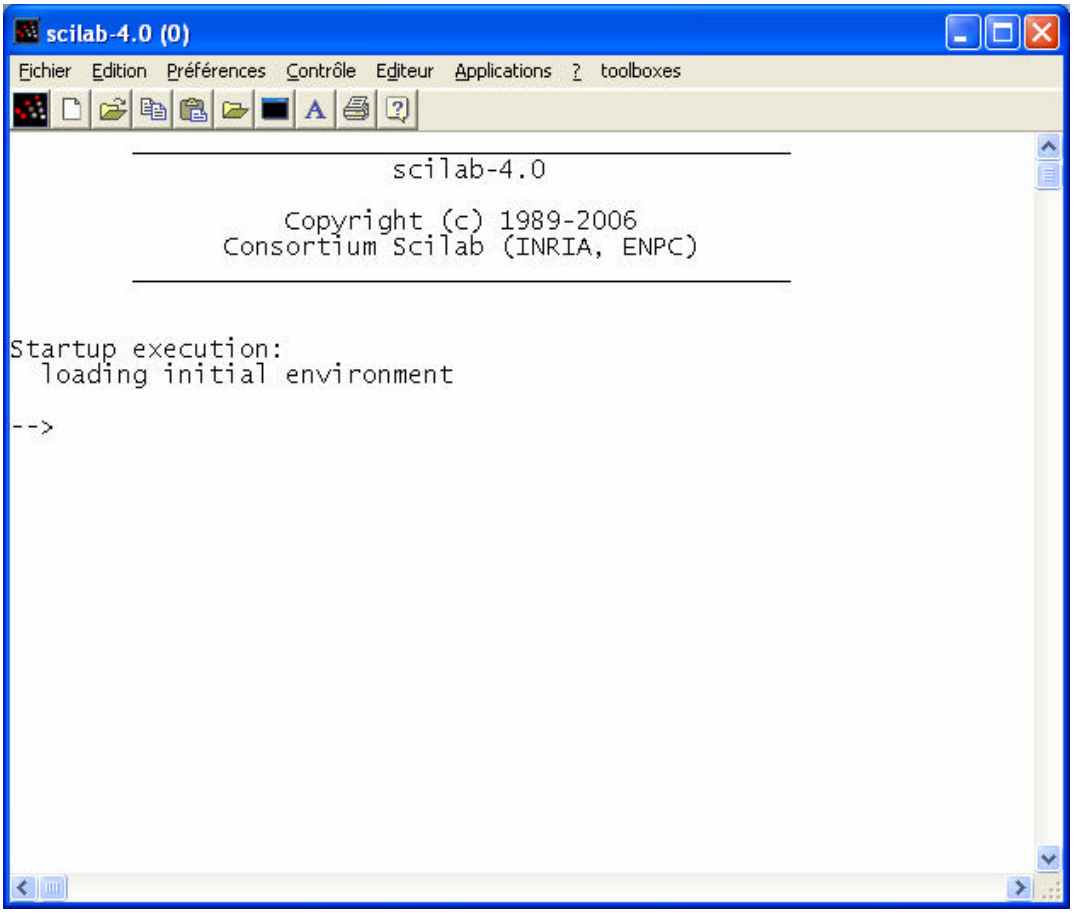
Découverte de Scilab

Question 1

Effectuer une recherche documentaire sur Scilab ; résumer en une page environ les caractéristiques de ce logiciel en insistant sur les points forts et les points faibles (ce qu'il fait et ce qu'il ne fait pas).

Le logiciel

La version dont nous disposons (dernière version « gratuite » sous Windows) se présente ainsi :



Le symbole « - - > » est le « prompt » ou « message d'accueil » du système qui nous indique que Scilab est prêt à interpréter nos commandes.

Eléments du langage

Les objets de base :

Scilab manipule

- des *nombres* (réels et complexes), des booléens, sous forme de variables et de constantes ; les noms des constantes prédéfinies commencent par le symbole « % » :
  - **%pi** pour le réel *p*
  - **%t** et **%f** pour les booléens vrai (true) et faux (false) respectivement
  - **%i** pour le complexe i
- des *chaînes* de caractères délimitées au choix par des apostrophes (‘) ou des doubles quotes (”)

Calculette réelle :

Taper (dans Scilab évidemment !) les commandes suivantes :

```
--> 2+2 ; // noter le « ; »
--> 2+2 // noter l’absence du « ; »
--> sin(pi)
--> %pi
--> %Pi
--> sin(%pi)
```

Question 2

1. quel est le rôle du symbole « // » ?
2. quel est le rôle du « ; » ?
3. qu’implique la différence entre « %pi » et « %Pi » ?
4. comment expliquer la réponse à « sin(%pi) » , Déterminer la cause de l’erreur.

```
--> help
```

Question 3

Explorer le système d’aide (remarquer qu’une partie est en anglais, une autre en français !), relever la liste des fonctions élémentaires (que vous connaissez) et repérer en particulier la fonction logarithme.

```
--> %eps
--> 1+%eps
--> 2^3 ;
--> 1/3 // noter le nombre de chiffres
--> format(12);1/3 // noter le nombre de chiffres
--> format(24);1/3 // noter le nombre de chiffres
--> format(30);1/3 // noter le nombre de chiffres
--> // Conclusion ?
--> sin(%pi)
--> help format
```

Question 4

1. Quelle est à votre avis la signification de la constante prédéfinie « %eps » ?
2. Décrire la signification de la commande « format ». Est-ce une commande d'affichage ou une commande de précision des calculs ?

Les variables

Le symbole d'affectation est « = » comme en VB (et en C), le symbole d'égalité est « = = » (comme en C).
La syntaxe des identificateurs (« nom » de variables, de fonctions, …) est la même que dans tous les langages informatiques (au symbole « % » près réservé à des constantes prédéfinies)
Scilab utilise un typage dynamique, ce qui signifie que vous n'avez pas à déclarer les variables explicitement, pas plus que leur type (qui sera déterminé implicitement au moment de l'affectation). Les variables ont néanmoins un type que vous pouvez vérifier par « typeof »

```
--> a=5 ;
--> a
--> typeof(a)
--> b=3
--> a+b
--> c=a+b ;
--> c
--> sin(a)
```

Question 5

Reconsidérer le rôle du « ; » en fin de commande : est-ce celui que vous aviez d'abord envisagé ?

Calculette complexe

```
--> a=1+2*%i  // n'oubliez pas le symbole « * »
--> a^2
--> abs(a)
--> real(a)
--> imag(a) ;
--> b=1-%i;
--> a/b
--> sqrt(a)
--> abs(a/b)-abs(a)/abs(b)
--> help atan
```

Question 6

1. Résumer les opérations sur les nombres complexes.
2. Les calculs sont-ils exacts ?
3. Comment obtenir l'argument d'un nombre complexe ?

Les vecteurs

```
--> u=[1,2,3]
--> u(1), u(2), u(3)
--> v=[-1,1,0]
--> u+v
--> u*v
--> v'
--> u*v'
--> log(u), u^2
--> u^2, sqrt(u)
```

Question 7

1. Expliquer la manière de définir un vecteur, ainsi que la manière d'accéder aux composantes (coordonnées) d'un vecteur.
2. Le « ' » après un vecteur en donne la « transposée » : expliquer en quoi cela consiste.
3. Comparer « u*v' » avec le produit scalaire de « u » et de « v » ; vérifier dans d'autres cas.
4. Remarquer que les fonctions numériques s'appliquent aux vecteurs : de quelle manière ?
5. Quel est le rôle de la virgule dans une ligne de commande ?

```
--> u.*v //noter le « .» (point) avant le « * »
--> u./v // même remarque (point avant /)
--> //constructeurs de vecteurs
--> u=0:1:10
--> v=1:2:8
--> w=2:0.5:50
--> u=[u,6,7]
--> u=[u,v]
```

Question 8

1. Indiquer le rôle du point devant un symbole d'opération lorsqu'on l'applique à des vecteurs
2. Résumer les méthodes de construction de vecteurs (que donne **n :p :q ?).** Vous pouvez évidemment faire d'autres essais pour confirmer (ou infirmer) ce que vous avez compris.
3. Construire le vecteur des 20 premiers entiers.
4. Construire le vecteur des 20 premiers entiers pairs
5. Construire le vecteur des 20 premiers entiers impairs

Les Matrices

Une **matrice** est un tableau comprenant un certain nombre de **lignes** et de **colonnes**.
On notera $\boldsymbol{A} = (\boldsymbol{a}_{i,j})$ une matrice, $\boldsymbol{a}_{i,j}$ désignant l'élément de la ligne **i** et de la colonne **j**
Les éléments d'une matrice peuvent être des nombres (réels, complexes, booléens) ou des chaînes de caractères.
Lorsque les éléments des matrices sont des nombres, on définit certaines opérations sur les matrices :

- ➔ Etant données deux matrices de mêmes dimensions (même nombre de lignes et de colonnes) on définit leur **somme** par $(\boldsymbol{a}_{i,j}) + (\boldsymbol{b}_{i,j}) = (\boldsymbol{a}_{i,j} + \boldsymbol{b}_{i,j})$
- ➔ Etant donnés une matrice $\boldsymbol{A} = (\boldsymbol{a}_{i,j})$ et un nombre l , on définit la matrice $l\boldsymbol{A}$ par

$$l\boldsymbol{A} = (l\ \boldsymbol{a}_{i,j})$$

- ➔ Etant données une matrice $\boldsymbol{A}$ à $\boldsymbol{n}$ lignes et $\boldsymbol{p}$ colonnes et une matrice $\boldsymbol{B}$ à $\boldsymbol{p}$ lignes et $\boldsymbol{q}$ colonnes, on définit le produit $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$ par

$$\boldsymbol{c}_{i,j} = \sum_{k=1}^{p} \boldsymbol{a}_{i,k}\boldsymbol{b}_{k,j}$$

($\boldsymbol{C}$ est une matrice à $\boldsymbol{n}$ lignes et $\boldsymbol{q}$ colonnes)

```
--> A=[1,2 ;3,4] // noter la différence entre “,” et “;”
--> size(A)
--> length(A)
--> matrix(A,4,1)
--> B=zeros(2,3), C=ones(2,3),B'
--> D=eye(3,3), E=eye(3,4)
--> A+B //Explication?
--> A+ones(2,2), A*C, 2*E
--> sin(A),log(A),A^2,A*A,A.^2
```

Question 9

1. Que représente « size(A) » ?
2. Quelle est la différence entre « A^2 » et « A.^2 » (notez le point avant « ^ »)
3. Si A est une matrice à 2 lignes et 3 colonnes, B une matrice à 3 lignes et 4 colonnes,
   a. peut-on faire la somme A+B ? B+A ?
   b. peut-on effectuer le produit A*B ? B*A ? Si oui, quels sont les dimensions de la matrice résultat ?
4. Sous quelles conditions peut-on calculer le carré d'une matrice ?
5. Donner la syntaxe complète de la commande « matrix »

```
--> A=rand(2,3) // matrice aléatoire
--> A(2,3),A(2, :),A( :,3),A(1,$),A( :,$)
--> A(2,2 :3),A( :,2 :3)
```

Question 10

Comment extraire

1. un élément d'une matrice ?
2. une ligne, une colonne ?
3. la dernière ligne, la dernière colonne ?
4. la matrice comprenant les éléments $a_{i,j}$ pour $p \leq i \leq q$ et $r \leq j \leq s$?

Les polynômes

Scilab n'est pas un calculateur formel (ou calculateur algébrique), mais possède quelques fonctionnalités qui s'en rapprochent, notamment en matière de polynômes.

```
--> p=poly([1,2,3,4],'X') //surprenant, le résultat !
--> roots(p) // c'est plus clair ?
--> help poly
```

Question 11

1. Comment obtenir le polynôme $X^3 - 3X^2 + 2X - 5$?
2. Déterminer les racines de ce polynôme.

```
--> q=poly([1,2,3,4],'X','c')
--> p+q, p*q,p/q
--> help poly
```

Question 12

1. Scilab peut-il manipuler les fractions rationnelles ?
2. Déterminer les racines de q et de $p+q$
3. Déterminer le PGCD de p et de q (rechercher dans l'aide)
4. Déterminer le PPCM de p et de q (rechercher dans l'aide)

```
--> X=poly(0,'X') // définit une variable X comme étant égale au
// polynôme X
--> P=X^2+3*X+5
--> degree(p)
--> roots(P)
--> derivat(p) //toujours du calcul presque formel
--> derivat(1/p)// et encore …
--> integrate(p)
--> help integrate //pas d'intégration formelle
--> help pdiv
```

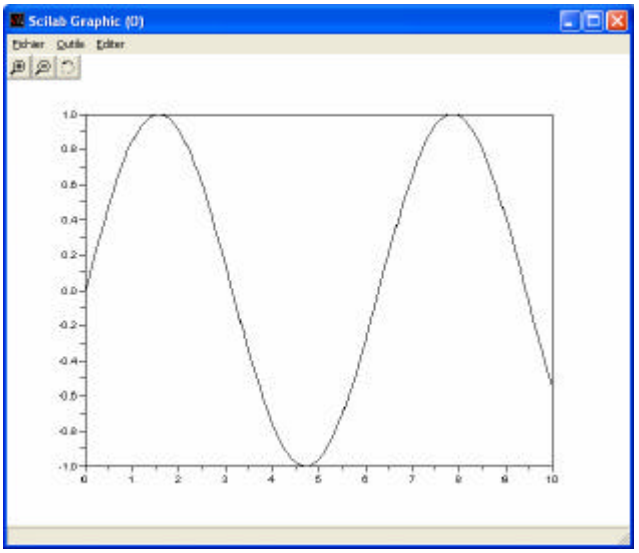
Question 13

1. Résumer les différentes opérations réalisables sur des polynômes
2. Quelles fonctions peut-on dériver formellement en Scilab ?
3. Calculer $\int_0^3 (X^3 + 1)^5 \, dX$ (en utilisant Scilab !)
4. Calculer une valeur approchée de $\int_0^5 \sqrt{1 - x^2}\, dx$

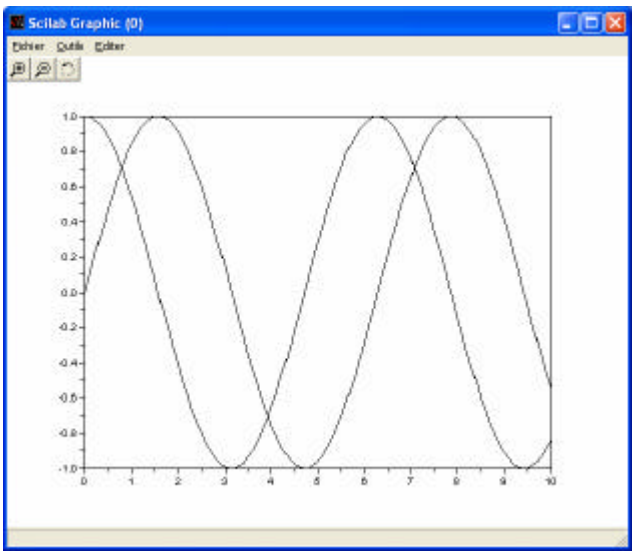
Les graphiques

Un des points forts de Scilab est la grande variété de graphiques qu'il nous propose.

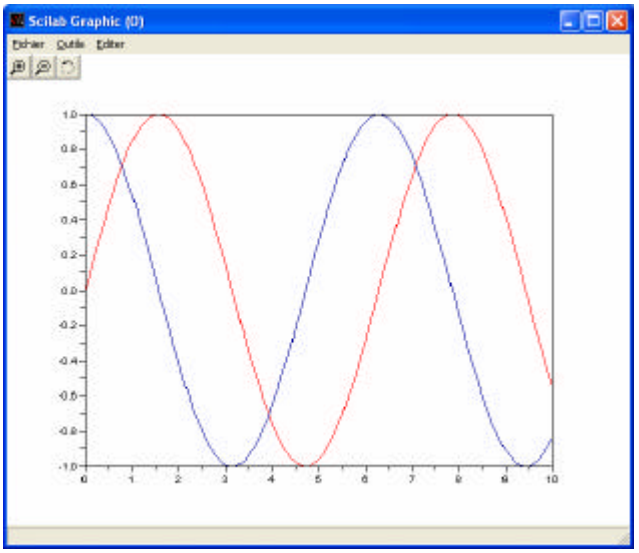
```
--> t=0 :0.1 :10 ; //Pourquoi un « ; » ?
--> plot2d(t,sin(t))
```




```
--> plot2d(t,cos(t))
```



On remarque que les deux graphiques sont superposés, et donc que, à moins de déjà bien connaître les courbes, il sera difficile de faire la différence en la courbe de la première fonction et celle de la seconde. Mettons de la couleur :



Question 14

1. Obtenir l'aide de « plot2d » et traduire cette aide
2. Obtenir le graphique « coloré » précédent ; on utilisera la commande « getcolor » pour obtenir les codes des couleurs et on dressera un tableau des codes des couleurs usuelles (bleu, rouge, vert, jaune, etc…)
3. Donner un titre au graphique (commande « titlepage » ou « xtitle »)
4. Donner une légende pour chacune des deux courbes
5. Donner la syntaxe des commandes « xsetech », « xgrid », « xbasc » , « xset »

Exercices

1. Tracer les courbes d'équations paramétriques

   a. $\begin{cases} x = \cos^3 t \\ y = \sin^3 t \end{cases}$ b. $\begin{cases} x = 2\cos 2t \\ y = 3\sin 3t \end{cases}$

2. Tracer le cercle de rayon 2
3. Calculer une valeur approchée de

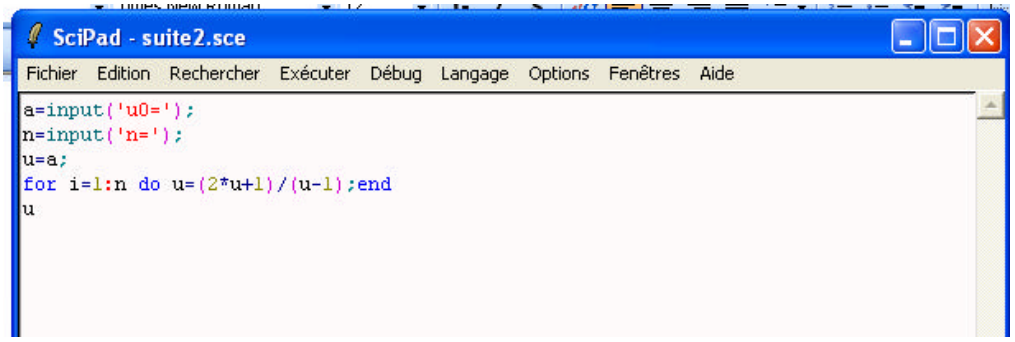
   a. $\int_0^2 \frac{2x+5}{x^2+2x-3}dx$ et $\int_2^5 \frac{2x+5}{x^2+2x-3}dx$ et $\int_0^p \sin x\,dx$

   b. $\int_0^2 \frac{x^2-2x+3}{x^2+x+5}dx$ et $\int_{-1}^1 \frac{\sin x}{x}dx$ et $\int_{-p}^p \frac{\sin x}{x}dx$

PROGRAMMATION SCILAB

Les scripts

Scilab nous permet d'écrire des suites d'instructions dans un fichier texte séparé, que nous pouvons écrire à l'aide de l'éditeur intégré (menu « Editeur »).



Nous sauvegardons ce fichier sous un nom (adapté) et une extension qui est, par convention d'usage « sce ».

Pour obtenir l'exécution du script, nous pouvons lancer depuis la fenêtre principale de Scilab
-- > exec('nom du fichier')
Par exemple
-->exec('C:\Documents and Settings\travail\Mes documents\SciLab\scripts\suite2.sce');
Plus simplement, et de manière plus conviviale, nous pouvons utiliser l'option « exec ... » du menu « file » de Scilab.

Question 15

Recopier le script suivant assurant le calcul du ***n***-ième terme de la suite ***u*** définie par

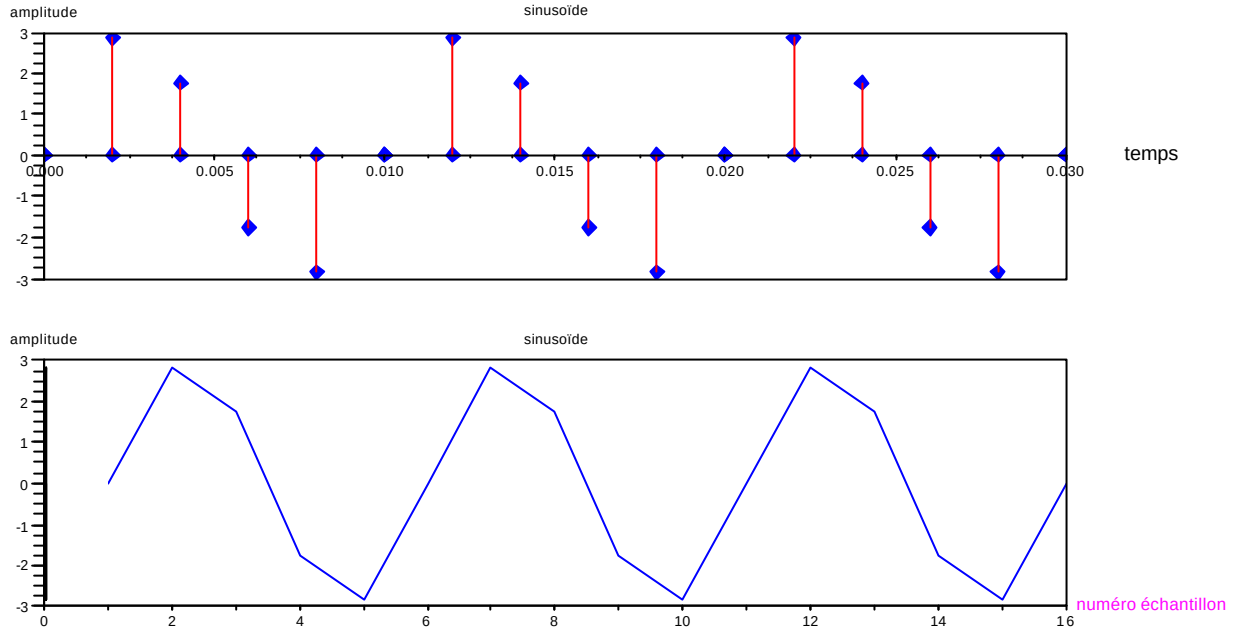
$$\begin{cases} u_0 = a \\ u_{n+1} = \dfrac{2u_n + 1}{u_n - 1} \end{cases}$$

```
a=input('u0=');
n=input('n=');
u=a;
for i=1:n do
u=(2*u+1)/(u-1); 'tester ensuite en supprimant le «;» final
end
u;
```

et lancer son exécution depuis Scilab.
Noter la fonction « input » qui permet un dialogue avec l'utilisateur.

Question 16

Taper cet autre script et modifier le graphique obtenu pour obtenir ceci :




```
clear
nb_pts=input('nombre de points : ');
pas=input('période d''échantillonnage : ');
t=pas*(0:1:nb_pts-1);
f=input('fréquence : ');
amp=input("amplitude : ");
s=amp*sin(2*%pi*f*t);
xsetech([0,0,1,1/2]);
plot2d(t,s)
xtitle("sinusoïde","temps","amplitude");
xsetech([0,1/2,1,1/2]);plot2d(s);
xtitle("sinusoïde","numéro échantillon","amplitude") ;
```

Commenter les différentes lignes de ce script.

Les conditionnelles

IF

Le langage Scilab dispose des trois variantes usuelles :

a) If ... then ... end
b) If ... then ... else ... end
c) If ... then ... elseif ... then ... elseif ... then ... else ....end

On notera que **IF** est fermé par **END** et que les instructions sont séparées par des virgules ou des «;» ou des sauts de ligne.

Question 17

Ecrire un script permettant de déterminer si une année est bissextile ou non (l'utilisateur entre l'année et le script répond 'bissextile' ou non'bissextile'). On pourra utiliser la commande « disp » ou affecter la réponse à une variable.

SELECT

Resemble beaucoup au Select Case de Visual Basic :

```
select expr,
   case expr1 then instructions1,
   case expr2 then instructions2,
   ...
   case exprn then instructions,
   [else instructions],
 end
```

LES BOUCLES

FOR

```
for variable=debut:pas:fin, ...,end
ou
for variable=u, ... , end 'u est un vecteur
```

EXEMPLE

Le calcul de ***n!*** peut se faire par :

```
n=7
u=1
for i=2:n,u=u*i,end
```

WHILE

```
while expression do instructions,end
```

ou

```
while expression do instructions1 else instruction2, end
```

EXEMPLE

Le calcul de ***n!*** peut se faire par :

```
n=7
u=1;i=2
while i<=n do u=u*i,i=i+1,end
```

LES FONCTIONS

```
function [y1,...,yn]=NomDeLaFonction(x1,...,xm)
...
Endfunction
```

Scilab manipulant par défaut des vecteurs, nous pouvons définir des fonctions renvoyant des vecteurs, donc en fait plusieurs fonctions en une !

EXEMPLE

Ecrivons une fonction qui renvoie la plus petit et le plus grand élément d'une liste :

```
Function [m,M]=MinMax(u)
m=u(1),M=u(1)
for i=u
if i<m then m=i,end
if i>M then M=i,end
end,
endfunction
```

VARIANTE

Dans le cas de fonctions définies par une expression « simple », nous pouvons utiliser une définition simplifiée d'une fonction sous la forme

```
deff('[y1,y2,...]=nom_fonction(x1,x2,....)','expression fonction')
```

Par exemple

```
deff('y=sh(x)','y=(exp(x)-exp(-x))/2')
```

FICHIERS DE FONCTIONS

Les fonctions peuvent être regroupées dans un fichier (comme un script). Dans ce cas on donnera plutôt (convention d'usage) l'extension « sci » au fichier. Pour inclure les fonctions en question dans une session Scilab on utilisera « getf(nom_du_fichier) » (au lieu de « exec »)

Question 18

a) Ecrire un script qui affiche la table de multiplication

```
!  1.   2.   3.   4.   5.   6.   7.   8.   9. !
!  2.   4.   6.   8.   10.  12.  14.  16.  18. !
!  3.   6.   9.   12.  15.  18.  21.  24.  27. !
!  4.   8.   12.  16.  20.  24.  28.  32.  36. !
!  5.   10.  15.  20.  25.  30.  35.  40.  45. !
!  6.   12.  18.  24.  30.  36.  42.  48.  54. !
!  7.   14.  21.  28.  35.  42.  49.  56.  63. !
!  8.   16.  24.  32.  40.  48.  56.  64.  72. !
!  9.   18.  27.  36.  45.  54.  63.  72.  81. !
```

b)

1. en utilisant une boucle
2. en utilisant les expressions Scilab

Ecrire une fonction indiquant si une année est bissextile ou non.

Question 19

Ecrire une fonction « *periodise(f,T,t)* » qui calcule *g(t)* où *g* représente la fonction de période *T* définie sur $[\mathbf{0;}\boldsymbol{T}]$ par $\boldsymbol{g(t)} = \boldsymbol{f(t)}$

Question 20

Ecrire une fonction renvoyant les *n* premières lignes du triangle de Pascal (en fait une matrice à *n* lignes et *n* colonnes, les éléments complétant les lignes étant nuls)

Question 21

a) Ecrire une fonction renvoyant l'impédance complexe d'un circuit *RLC* et traçant les courbes du gain et du déphasage du filtre *RLC* avec sortie sur *R.*

b) Même question pour un filtre *RL* avec sortie sur *R*, puis sortie sur *L*.

Question 22

a) Ecrire une fonction « fct »(n) calculant *n!* de deux manières différentes (avec une boucle, puis en utilisant la fonction « prod » de Scilab) dans un fichier.

b) Tester la fonction avec différentes valeurs (entières !) de *n*.

c) Vérifier que, contrairement aux fonctions prédéfinies de Scilab, cette fonction ne s'applique pas aux vecteurs (et plus généralement aux matrices).

d) Ecrire une fonction « fact » qui calcule *n!* en s'appliquant aux matrices (on pourra utiliser la fonction « fct » définie précédemment et effectuer une double boucle sur tous les éléments de la matrice n après en avoir déterminé les dimensions).

Question 23

Ecrire une fonction *« coef_fourier (f,T,n) »* qui calcule les coefficients de Fourier d'une fonction « *f* » de période « *T* » (en fait « f » est l'expression de la fonction sur $[0;T]$) pour les « n » premières harmoniques ; la fonction renverra les résultats sous forme de matrice (*n+1* lignes et *2* colonnes de réels ou *n+1* lignes et *1* colonne de complexes). Noter que les fonctions (ici *f*) peuvent être passées en paramètre.

On considère le signal *s* de période *2* défini sur $[0;2]$ par $s(x) = e^x$.

a) Déterminer les coefficients de Fourier de *s* jusqu'à la dixième harmonique

b) Calculer la puissance du signal

c) Déterminer le nombre *n* d'harmoniques nécessaires pour conserver 99% de la puissance du signal

d) Tracer la courbe représentative du signal et de la somme de ses *n* premières harmoniques

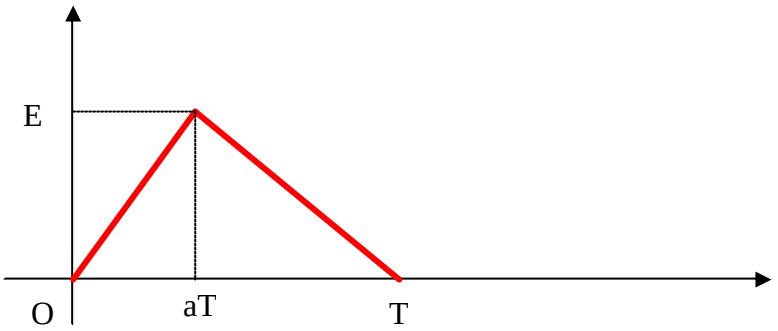
e) Automatiser ce travail dans un script et/ou une fonction

APPLICATIONS AU TRAITEMENT DU SIGNAL

Décomposition en série de Fourier

Question 24

On considère un signal « triangulaire » de période T défini sur $[O,T]$ par la représentation graphique



1. Ecrire en Scilab une fonction « triangle » prenant en argument les valeurs de T, de a ($0 < a < T$) et de t et calculant la valeur de ce signal au temps t. On fera en sorte que cette fonction puisse s'appliquer à un «vecteur» t.
2. Tester cette fonction en construisant la représentation graphique lorsque $T = 1$ et $a = \frac{1}{3}$.
3. Calcul des coefficients de Fourier de cette fonction et étude des harmoniques :
   a) écrire au choix une fonction «coef_c» ou deux fonctions «coef_a» et «coef_b» calculant soit les coefficients complexes (c_n) soit les coefficients réels (a_n) et (b_n).
   b) tracer la représentation graphique de la fonction obtenue en prenant la somme des 2 (resp. 3, resp. 4, resp. 5) premières harmoniques.
   c) écrire une fonction calculant la puissance du signal, ainsi qu'une fonction calculant la puissance de la somme des n premières harmoniques
   d) déterminer la plus petite valeur de n pour laquelle la puissance de la somme des n premières harmoniques du signal est supérieure ou égale à 95 % (resp. 99 %) de la puissance du signal.

Transformée de Fourier à Temps Discret

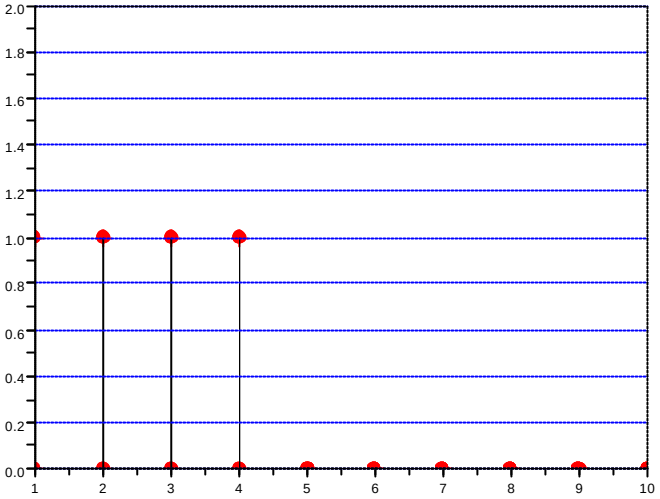
Etant donné un signal numérique x, on appelle **transformée de Fourier à temps discret** (TFTD) du signal la fonction X de f définie par

$$X(f) = \sum_{n=0}^{+\infty} x_n e^{-i2\pi n f}$$

Question 25

On considère le signal x défini par x_n = **(1,1,1,1,0,...0...)**

a) Construire la représentation graphique du signal (reproduire le graphe ci-dessous) :



b) Déterminer la transformée de Fourier à temps discret de x

c) Déterminer le spectre de fréquence de x (vérifier qu'il est égal à $2\left|\cos(3\pi f) + \cos(\pi f)\right|$) et tracer sa courbe. Vérifier que le spectre est périodique : quelle est sa période T?

Tracer la représentation graphique du spectre sur $\left[-\frac{T}{2}, \frac{T}{2}\right]$

d) Déterminer le spectre de phase de x et tracer sa courbe

e) Déterminer la densité spectrale de puissance de x et tracer sa courbe

f)

On utilisera Scilab et/ou Mupad et l'on comparera les deux logiciels sur cet exemple.

Transformée de Fourier discrète

Question 26

Etudier la transformée discrète du signal précédent et comparer à la transformée à temps discret.

Echantillonnage

Un signal numérique résulte le plus souvent d'un signal analogique que l'on a échantillonné, i.e. dont on a prélevé des échantillons à des instants discrets. Usuellement, les échantillons sont prélevés à des instants uniformément répartis.

Si $x(t)$ est un signal à temps continu, la suite $x_n = x(nT_e)$ où T_e désigne l'intervalle entre deux prélèvements constitue un échantillonnage du signal $x(t)$.

T_e est appelé **période d'échantillonnage**. Son inverse $f_e = \frac{1}{T_e}$ est appelé **fréquence d'échantillonnage** (ou encore cadence d'échantillonnage).

Théorème d'échantillonnage (Whittaker-Kotelnokov-Shannon)

Soit $x(t)$ un signal à temps continu de **largeur de bande** B (ce qui signifie qu'il est composé de sinusoïdes de fréquences au plus égales à B).
Soit f_e la fréquence d'échantillonnage, T_e la période d'échantillonnage et $x_n = x(nT_e)$ les échantillons de x.
Si $f_e > 2B$, alors $x(t)$ peut être reconstruit à partir de ses échantillons en utilisant la **formule d'interpolation** :

$$x(t) = \sum_{n=-\infty}^{+\infty} x(nT_e)\,\mathrm{sinc}(p(f_e t - n))$$

où **sinc** désigne le **sinus cardinal** défini par $\mathrm{sinc}(x) = \frac{\sin x}{x}$.
$2B$ est appelée **fréquence de Nyquist** ou **fréquence de Shannon** du signal.

Question 27 Vérification du théorème d'échantillonnage

On considère le signal défini par

$$s(x) = 2\sin x - 3\cos(p\,x)$$

1. Vérifier que ce signal n'est pas périodique
2. Construire une représentation graphique de s
3. On décide d'échantillonner à 0.1, 1 puis 2 Hz : dans quels cas pourra-t-on reconstruire le signal ?
4. Effectuer l'échantillonnage dans chacun de ces cas (on prendra 50 points d'échantillonnage – i.e. 50 **échantillons** – dans chaque cas)
5. Représenter les trois signaux obtenus.
6. Appliquer la formule d'interpolation précédente à chacun de ces trois signaux numériques (on se limitera évidemment à un nombre fini de termes de la somme répartis également autour de 0) ; construire les graphes des signaux à temps continu

ainsi obtenus. Si le graphe ne correspond pas à la fonction de départ, essayer d'augmenter le nombre de termes dans la formule d'interpolation.

Transformée en z

Question 28

Donner une représentation graphique et déterminer la transformée en Z des signaux

- $x_n = (2n - 3)U(n)$
- $x_n = 2^n U(n - 1)$
- $x_n = n^2 U(n)$
- $x_n = e^{-n}\left(U(n) - U(n - k)\right)$ où $k \in \mathbb{N}^*$
- $x_n = e^{-n} \sin wn\, U(n)$

Question 29

Déterminer les transformées en Z inverses des fonctions et représenter les signaux obtenus :

- $\dfrac{1}{1 - 3z^{-1} + 2z^{-2}}$
- $\dfrac{1}{z^2 - 9z + 20}$
- $\dfrac{z - 1}{z + 3}$
- $\dfrac{1}{z^{-3} - 1}$

Filtrage numérique

Question 30

On considère le filtre dérivateur défini par l'équation aux différences finies $y_n = x_n - x_{n-1}$.

- Déterminer la fonction de transfert du filtre
- Déterminer et représenter la réponse impulsionnelle et la réponse indicielle du filtre
- Ecrire une fonction Scilab «derive» calculant la réponse à un signal donné en argument. On remarquera que si ***x*** est défini (non nul) pour ***n*** positif ou nul, ***y*** n'a véritablement le sens de la dérivée pour ***n*** strictement positif.
- Appliquer cette fonction «derive» aux signaux
  a) $x_n = \ln(x/10)$
  b) $x_n = \cos(x/10)$
- Construire les courbes correspondantes ainsi que les courbes des dérivées des fonctions sous-jacentes ; peut-on considérer cette méthode comme un moyen efficace pour obtenir la dérivée d'une fonction ?

Question 31

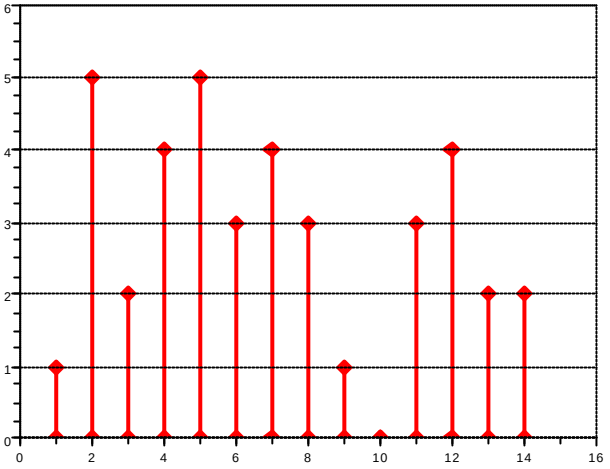
On considère le filtre **à moyenne pondérée** défini par l'équation aux différences finies
$y_n = 0{,}5\,x_n + 0{,}3\,x_{n-1} + 0{,}2\,x_{n-2}$

- Déterminer la réponse impulsionnelle de ce filtre et la représenter

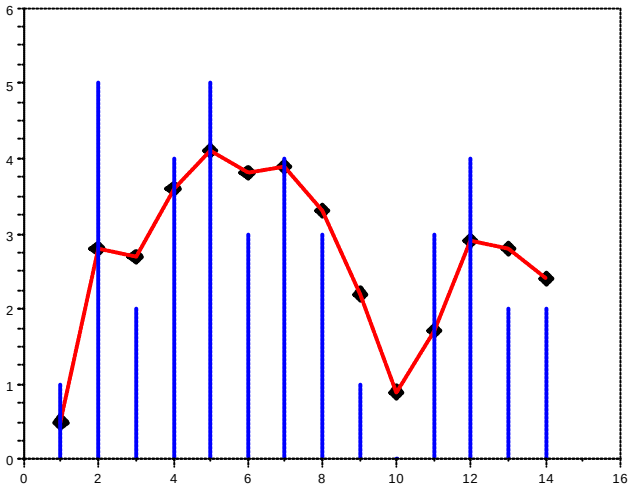
- Ecrire une fonction «fmp» donnant le signal de sortie ***y*** correspondant à un signal d'entrée causal ***x***. On remarquera que le signal d'entrée n'est donné qu'à partir de ***0***, et que par conséquent il est nécessaire dans l'écriture de la fonction d'envisager à part les premières valeurs de ***n***.
  On notera par ailleurs que les «vecteurs» Scilab sont numérotés à partir de ***1*** et non de ***0***.

- Déterminer la **réponse indicielle** (réponse à l'échelon unité) du filtre

- Déterminer la réponse du filtre au signal suivant :

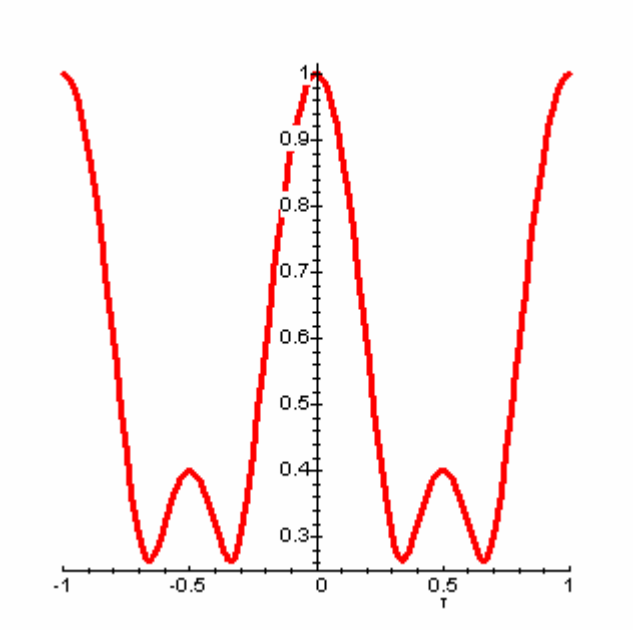


On remarquera qu'il s'agit bien d'un filtre opérant un «lissage» du signal d'entrée



(signal d'entrée en diagramme à bâtons, sortie en polygone)

- Déterminer la fonction de transfert ***H*** puis la transmittance du filtre. Déterminer les pôles et les zéros du filtre.
- Déterminer le module ***T*** de la transmittance et tracer la courbe représentative de ***T*** par rapport à la fréquence ***f*** (on montrera que $|T(f)|^2 = 0{,}38 + 0{,}42\cos(2\pi f) + 0{,}4\cos 4\pi f$)



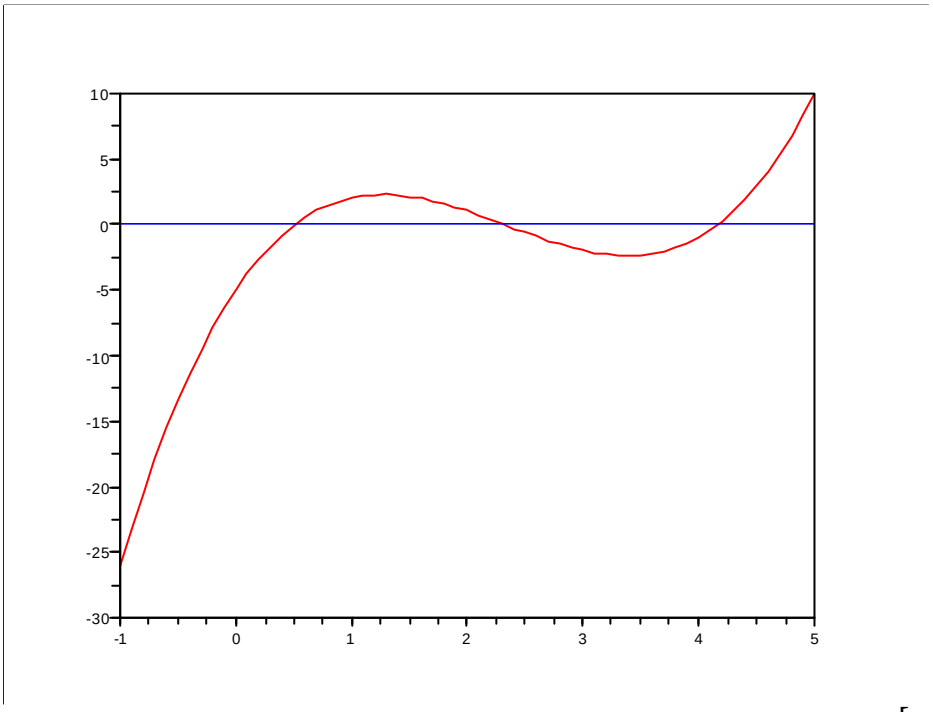
- Quelle est la période de cette fonction ?
- Quel nom porte cette fonction ?
- S'agit-il d'un filtre passe haut, passe bas, etc. ?
- Déterminer la fréquence de coupure du filtre (.1644295171 ?).

COMPLEMENTS SCILAB

Résolution d'équations : `fsolve`

Cette fonction permet de déterminer une valeur approchée d'une solution d'une équation de la forme $f(x) = 0$; on donne en argument une valeur initiale x_0 la plus proche possible de la solution cherchée, ainsi que la fonction f.

```
// définition de la fonction dont on veut déterminer les racines
function y=pol(x); y=x^3-7*x^2+13*x-5,endfunction
// tracer de la courbe
x=-15:0.1:15;
plot2d(x,pol(x)),plot2d(x,zeros(x))
```



Nous constatons que l'équation $plot(x) = 0$ admet trois solutions sur l'intervalle $[-5;5]$

```
s=[fsolve(0,pol),fsolve(2,pol),fsolve(4,pol)]
```

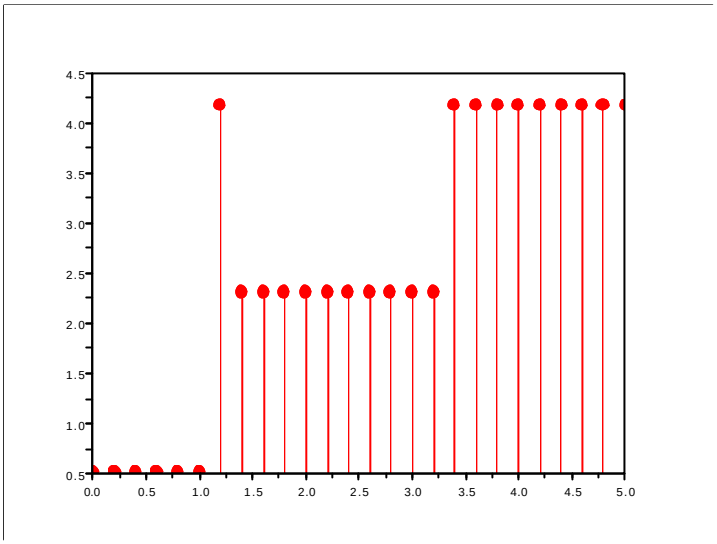
nous donne

```
!   0.5188057    2.3111078    4.1700865 !
```

La vrai question qui se pose est le choix des valeurs ***0, 2, 4*** ; essayons de déterminer la solution trouvée suivant la valeur de x_0 :

```
x0=[0:0.1:5];s=fsolve(x0,pol);
[x0',s']
```

nous donne

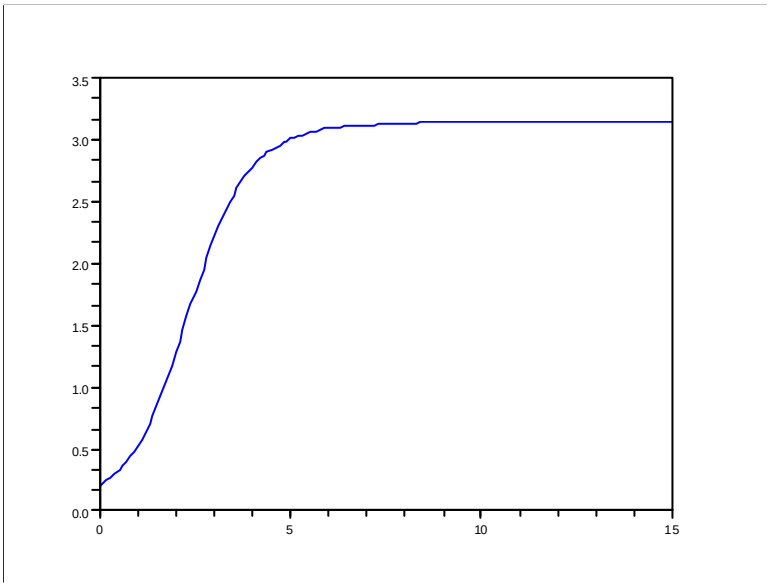


On remarquera que la solution la plus grande est atteinte pour une valeur de x_0 relativement éloignée, ce qui nous amènera à une certaine prudence dans le choix de la valeur initiale. L'utilisation de « fsolve » ne nous dispense pas de l'étude de la fonction, et notamment de la détermination du nombre de racines et d'un encadrement de ces racines.

Résolution d'équations différentielles : ode

La fonction « ode » permet de déterminer une solution approchée des équations différentielles de la forme $\begin{cases} \dfrac{dy}{dt} = f(t,y) \\ y(t_0) = y_0 \end{cases}$

```
function y=f1(t,y), y=sin(y), endfunction
y0=0.2;t0=0;
t=0:0.1:15;
y=ode(y0,t0,f1);
clf;plot(t,y);
```



EXERCICE : Conjecture de Syracuse

On considère la suite u définie par

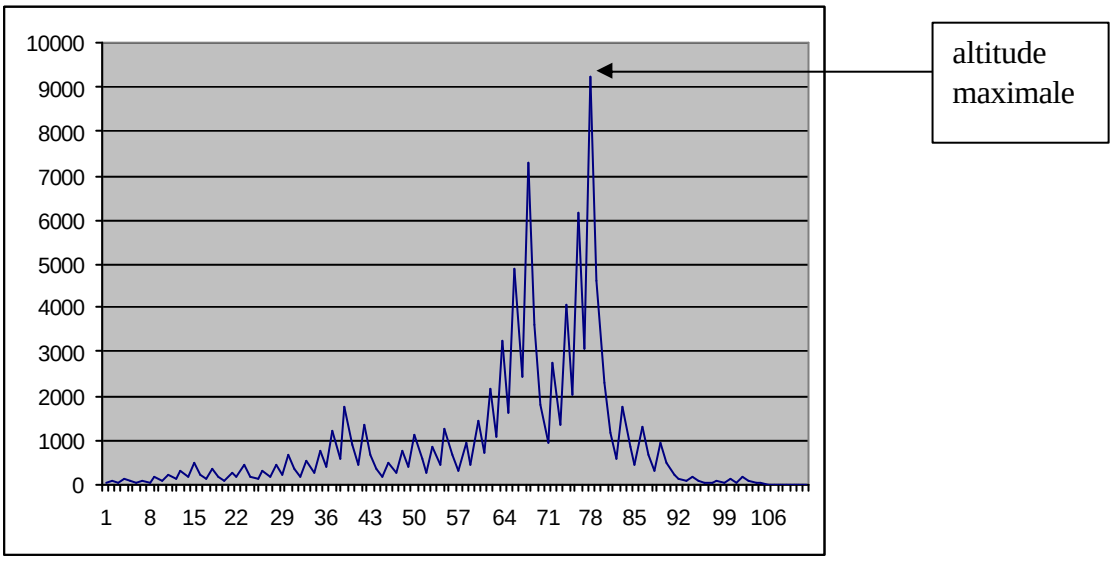
$$\begin{cases} u_0 \in \mathbb{N} \\ u_{n+1} = \begin{cases} \frac{u_n}{2} \text{ si } n \text{ est pair} \\ 3u_n + 1 \text{ si } n \text{ est impair} \end{cases} \end{cases}$$

La conjecture de Syracuse suggère que $\forall u_0 \in \mathbb{N}, \exists n \in \mathbb{N} / u_n = 1$;

- cet entier n, s'il existe, est appelé **durée du vol** de u_0.
- l'ensemble des termes de la suite jusqu'au rang n est appelé **trajectoire** de u_0
- la plus grande valeur prise par la suite est appelée **altitude maximale** (de u_0)
- la plus petite valeur de n pour laquelle $u_n < u_0$ est appelée **durée du vol en altitude** ; l'ensemble des termes de la suite du rang 0 à la durée du vol en altitude est appelé **vol en altitude**.
- on appelle **facteur d'expansion** de u_0 le rapport de l'altitude maximale sur u_0

Exemple

Représentons graphiquement la trajectoire de 27

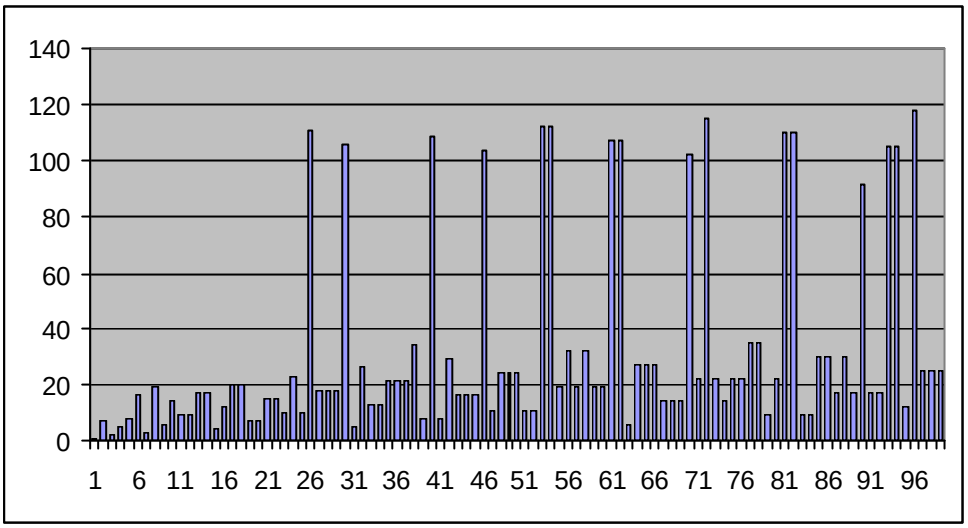


La durée du vol est 111, l'altitude maximale 9232, le vol en altitude 96

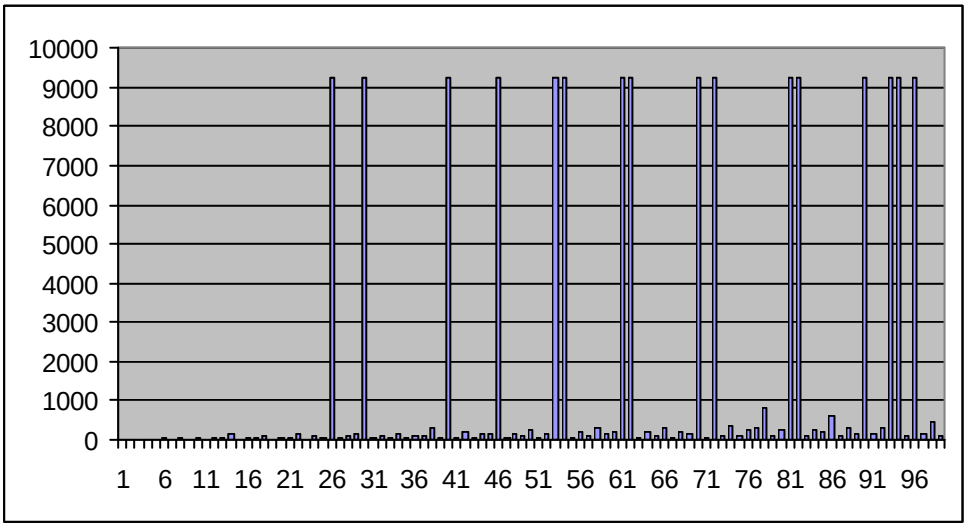
Question 32

- Ecrire une fonction « syr » qui renvoie u_{n+1} à partir de u_n

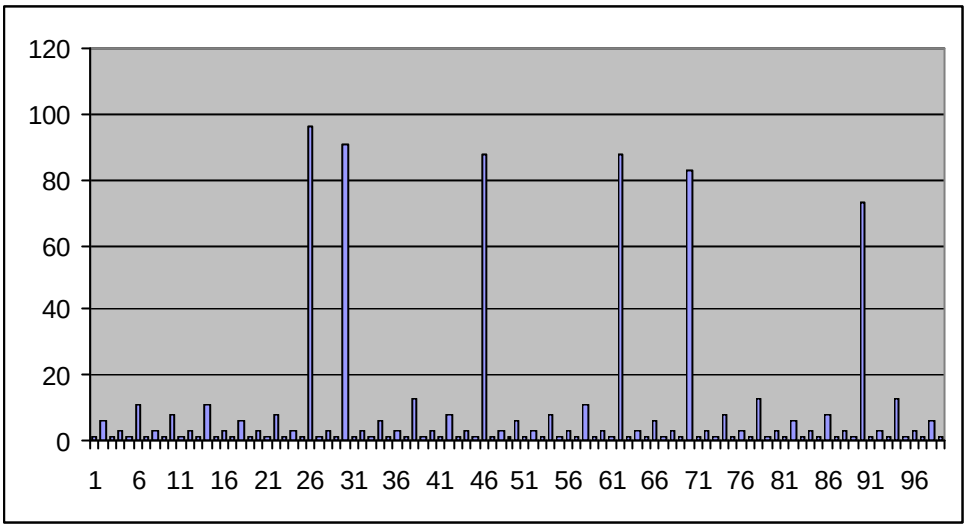
- Utiliser cette fonction pour calculer les 30 premiers termes de la suite de Syracuse en fonction de la valeur du premier terme. Déterminer ainsi le premier entier pour lequel 1 n'est pas obtenu au cours des 30 premiers termes de la suite.
- Ecrire une fonction « vol » qui renvoie la durée du vol d'un entier ***p*** (valeur de u_0)
- Déterminer le vol des 100 premiers entiers et construire une représentation graphique de ces valeurs.
-



- Ecrire une fonction « record » qui donne le nombre ayant la plus grande durée de vol parmi les entiers compris entre deux nombre ***a*** et ***b*** passés en paramètre. Déterminer le recordman parmi les nombres plus petits que 1000.
- Ecrire une fonction « alt » qui renvoie l'altitude maximale d'un entier. Donner une représentation graphique de l'altitude maximale des entiers de 2 à 100.



- Ecrire une fonction « vol_alt » qui détermine le vol en altitude d'un entier. Donner une représentation graphique du vol en altitude des entiers de 2 à 100.
-
-



- Déterminer, parmi les entiers inférieurs à 1000, ceux dont l'altitude maximale est la plus grande et ceux dont le vol en altitude est le plus long.
- Déterminer, parmi les entiers inférieurs à 1000, ceux dont le facteur d'expansion est le plus grand.

Contrôle

construction de signaux

Construire des vecteurs (limités aux dix premiers termes) représentant les signaux :

- rampe
  `u= 0:9`
- impulsion unité :
- on commencera par construire un vecteur uniligne comprenant 10 zéros :
  `u= zeros(1,10)`
- on modifie le premier terme en remplaçant le 0 par 1
  `u(1)=1`
- on construit un diagramme à bâtons («barplot») représentant ce signal
  `plot2d3(u)`
  ou de préférence
  `plot2d3([0:9],u)`
- échelon unité :
  `u= ones(1,10)`
- exponentiel 2^n (exponentiel à base 2)
- 1ère méthode : on rappelle que $2^n = \exp(n * \ln 2)$;
  `u= exp(log(2)*[0:9]) ou u=2^[0:9]`
- 2ème méthode : le signal est une suite géométrique de raison 2
  On utilise donc une programmation par boucle FOR
- on initialise **u** comme un vecteur contenant 10 zéros ou dix un
  `u= ones(1,10)`
- on utilise $u(i+1) = 2*u(i)$ dans une boucle FOR (après avoir – éventuellement – défini le premier terme
  `for i=2:10,u(i)=2*u(i-1);end`
  exponentiel à base quelconque a :
  écrire un script demandant à l'utilisateur l'entrée de la valeur de **a** et définissant le vecteur **u** associé :
  `a=input('a='),u=ones(1,10);for i=2:10,u(i)=a*u(i-1);end`

construction d'un filtre

On considère le filtre dérivateur dont l'équation aux différences finies est

$$y_n = x_n - x_{n-1}$$

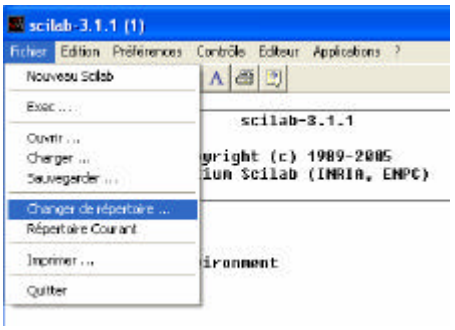
```
//Filtre dérivateur
function y=fder(x)
n=length(x)
y=zeros(1,n)
y(1)=x(1)
for i=2:n,y(i)=x(i)-x(i-1);end;
endfunction
```

Les réponses aux questions suivantes seront rédigées dans un fichier Word sauvegardé sur le disque «D» et nommé «CTRL_» suivi de votre nom ; les graphiques obtenus par Scilab y seront évidemment copiés.

- écrire un fichier contenant cette fonction
- en déduire la réponse impulsionnelle du filtre et la représenter
- en déduire la réponse indicielle du filtre et la représenter
- comparer dans ce cas la réponse du filtre à la dérivée de la fonction continue sous-jacente
- considérer le signal $x_n = \dfrac{2n+1}{3n-1}$ et étudier la réponse y_n du filtre à ce signal
- construire ce signal et sa réponse
- donner la dérivée de $f(x) = \dfrac{2x+1}{3x-1}$
- comparer y_n et $f'(n)$
- on échantillonne la fonction f en considérant le signal $a_n = f(nT_e)$ (où T_e est la période d'échantillonnage)
- écrire une fonction construisant ce signal échantillonné : on considérera un signal de longueur 100
- représenter le signal échantillonné pour différentes valeurs de la période d'échantillonnage
- construire les réponses du filtre à ce signal
- représenter, sur la même figure, le signal $f'(nT_e)$
- conclusion ?
- reprendre les questions précédentes pour les fonctions
- $\sin x$
- e^x
- $\ln x$
- $\arcsin x$
- $\arctan x$

Corrigé

La fonction «fder» est sauvegardée dans un fichier «filtres.sci» placée dans un répertoire quelconque, disons "c:\scilab\scripts".
Nous pouvons choisir ce répertoire comme répertoire par défaut en utilisant le menu



a) La réponse impulsionnelle du filtre est obtenue par

- chargement de la fonction

```
getf('filtres.sci')
```

si nous avons choisi le 'bon' répertoire comme répertoire courant ; sinon utiliser

```
getf('c:\scilab\scripts\filtres.sci')
```

- création du signal impulsion unité

```
imp=zeros(1,10) //matrice 1 ligne 10 colonnes constituée de 0
imp(1)=1 // valeur initiale à 1
```

noter que normalement c'est la valeur en 0 qui vaut 1 ; mais les vecteurs Scilab sont numérotés à partir de 1. C'est à nous d'interpréter correctement la correspondance vecteur – signal

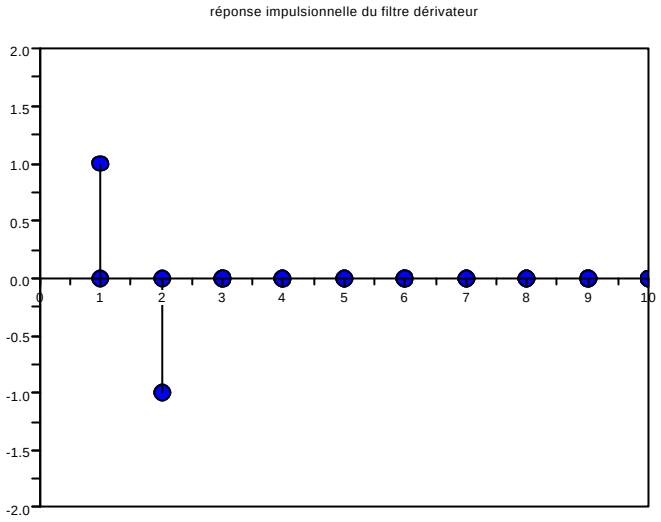
- obtention de la réponse impulsionnelle :

```
rep_imp=fder(imp)
```

- tracé de la courbe

```
t=0:9
plot2d3(t,rep_imp)
xtitle('réponse impulsionnelle du filtre dérivateur')
a=get("current_axes") //récupération de l'entité (objet) «axes»
a.x_location="middle"
a.data_bounds=[-1,-2;10,2]; // fixe les limites des axes
```

Après quelques modifications dans l'éditeur, on obtient :



- La réponse indicielle, c'est à dire la réponse à un échelon est obtenue par :

b) création du signal échelon

```
echelon=ones(1,10)
```

c) obtention de la réponse du filtre au signal et tracé de la réponse :

```
rep_ind=fder(echelon);plot2d3(t,rep_ind) //t a été défini plus haut
```

On remarque que l'on obtient l'impulsion unité (qui est donc en quelque sorte la "dérivée" de l'échelon unité !).

d)

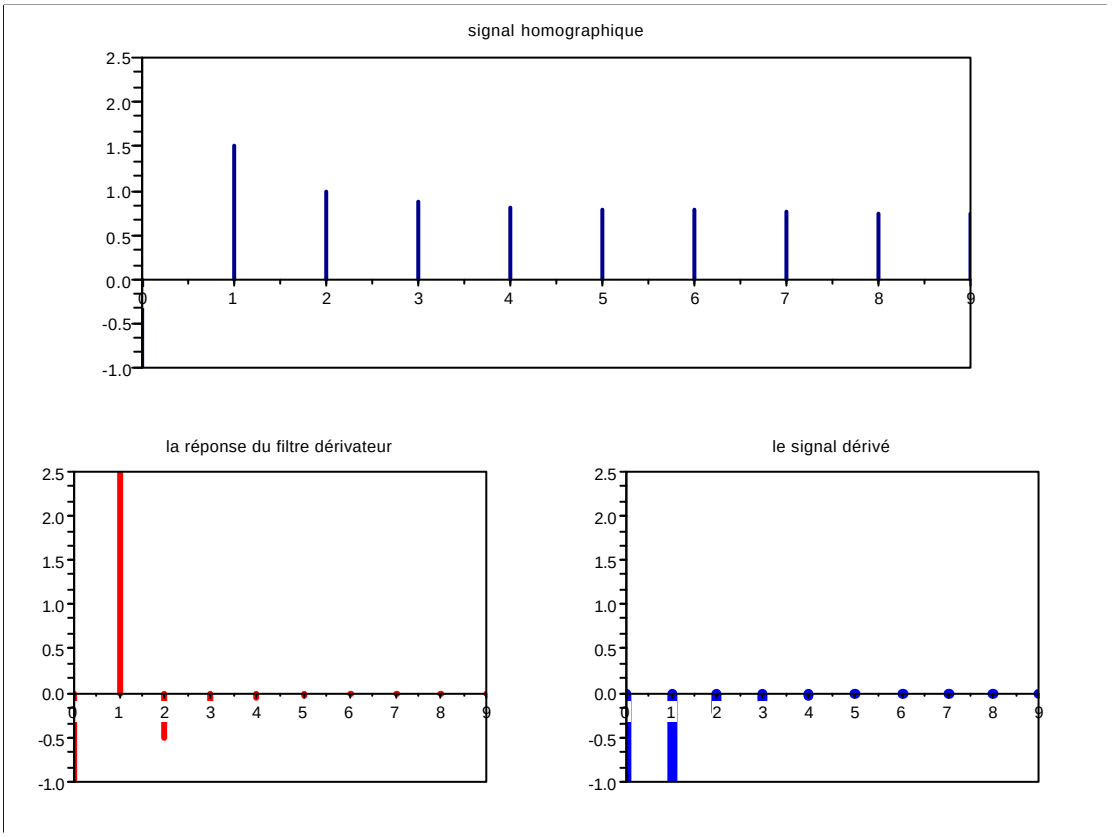
- Soit x le signal défini par $x_n = \frac{2n+1}{3n-1}$ et $f(x) = \frac{2x+1}{3x-1}$

a) Sous Mupad, nous définissons f et calculons sa dérivée :

```
f:=x->(2*x+1)/(3*x-1);
                          x -> (2*x + 1)/(3*x - 1)
diff(f(x),x):normal(%):factor(%);
                                    -5
                                ----------
                                        2
                                (3 x - 1)
```

b) Sous Scilab nous traçons les courbes ; nous écrivons un script (facilité de mise au point) :

```
t=0:9  //rappel
x=(2*t+1)./(3*t-1) //signal d'entrée
y=fder(x) // réponse du filtre
f=-5*ones(0:9)./(3*t-1)^2 //dérivée théorique échantillonnée
clf
//tracé du signal de départ
xsetech([0,0,1,1/2]);
plot2d3(t,x)
xtitle("signal homographique");
// tracé de la réponse du filtre
xsetech([0,1/2,1/2,1/2]);plot2d3(t,y);
xtitle("la réponse du filtre dérivateur") ;
// tracé de la dérivée échantillonnéé
xsetech([1/2,1/2,1/2,1/2]);plot2d3(t,f);
xtitle("le signal dérivé") ;
```



A part pour les premiers termes, nous constatons que la réponse du filtre semble correspondre à la dérivée (échantillonnée). Une représentation graphique peut cependant s'avérer trompeuse. Vérifions sur les valeurs numériques :

```
-->y'-f'
 ans  =
!   4.        !
!   3.75      !
! - 0.3       !
! - 0.046875  !
! - 0.0154959 !
! - 0.0069573 !
! - 0.0037074 !
! - 0.0022059 !
! - 0.0014178 !
! - 0.0009648 !
```

Les deux premières valeurs sont en effet très différente ! Cela s'explique aisément pour la première valeur puisque, et cela peut s'observer sur la programmation du filtre, le signal que nous traitons est supposé causal et donc nul à gauche de 0, ce qui rend la fonction sous-jacente discontinue – et donc non dérivable en 0 ! L'écart en 1 semble plus difficile à expliquer.
Mais avons nous correctement testé l'adéquation des deux signaux ? Une «petite» différence entre les deux signaux ne veut pas dire grand chose : la différence entre 0 et $\mathbf{10}^{-6}$ peut être considérée comme «petite», il n'empêche que l'erreur est de 100% !
Testons donc les différences relatives $\left|\frac{y_n - f_n}{y_n}\right|$ ou $\left|\frac{y_n - f_n}{f_n}\right|$; nous obtenons, en pourcentage

```
-->abs((y-f)./y*100)'
 ans  =
!   400.      !
!   150.      !
!   60.       !
!   37.5      !
!   27.272727 !
!   21.428571 !
!   17.647059 !
!   15.       !
!   13.043478 !
!   11.538462 !
```

ce qui montre que l'approximation n'est guère meilleure pour les dernières valeurs que pour les premières !

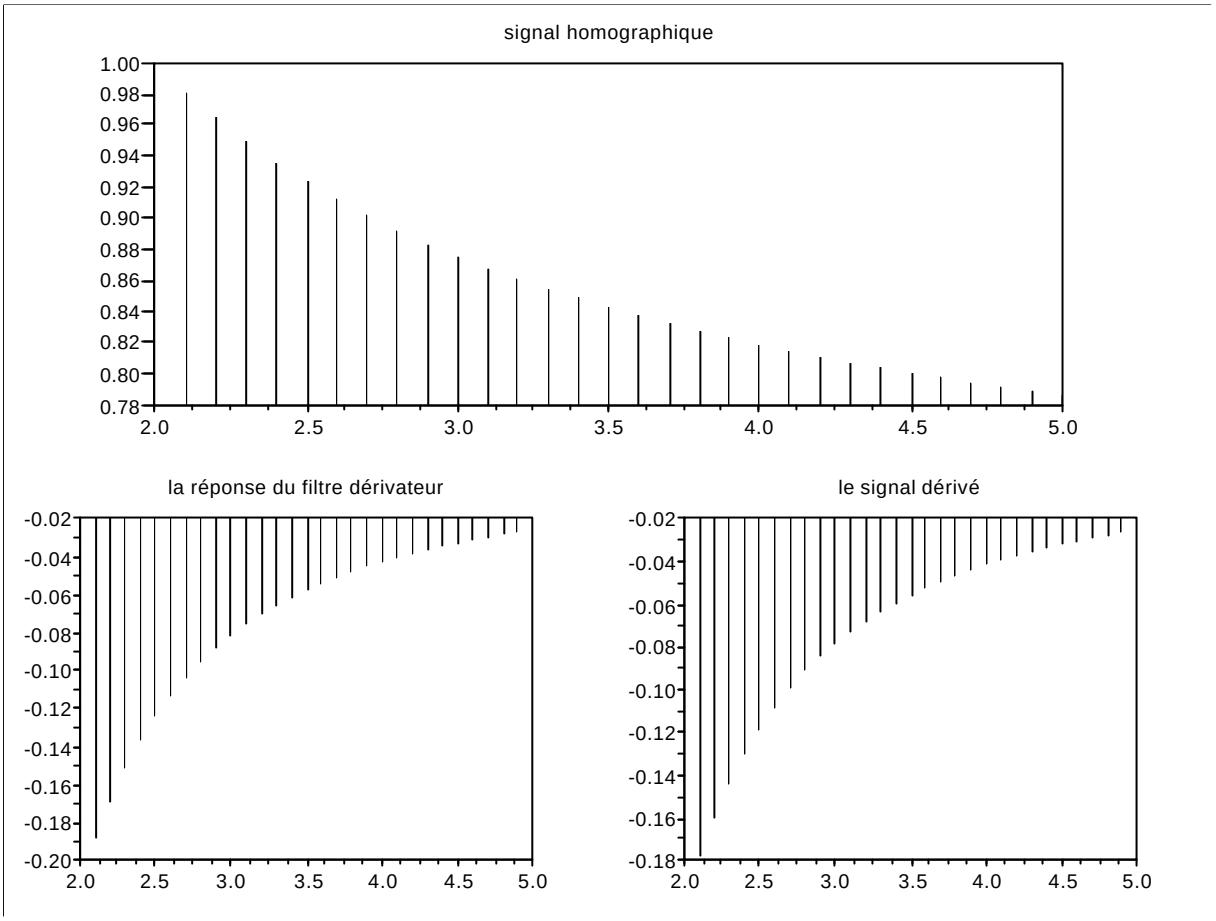
Ce qui n'est guère surprenant si l'on considère que nous avons échantillonné avec un pas de 1. Echantillonnons donc avec un pas plus petit $T_e = \mathbf{0.1}$:

```
a=input('Borne inférieure de l''intervalle d''étude : ');
b=input('Borne supérieure de l''intervalle d''étude : ');
Te=input('période d''échantillonnage');
t=a:Te:b;
n=length(t)
x=(2*t+1)./(3*t-1); //signal d'entrée
y=fder(x); // réponse du filtre
f=-5*ones(1,n)./(3*t-1)^2 ;//dérivée théorique échantillonnée
//suppression du premier terme
t=t(2:$);x=x(2:$);y=y(2:$);f=f(2:$);
clf
//tracé du signal de départ
xsetech([0,0,1,1/2]);
plot2d3(t,x)
xtitle("signal homographique");
// tracé de la réponse du filtre
xsetech([0,1/2,1/2,1/2]);plot2d3(t,(1/Te)*y);
xtitle("la réponse du filtre dérivateur") ;
// tracé de la dérivée échantillonnéé
xsetech([1/2,1/2,1/2,1/2]);plot2d3(t,f);
xtitle("le signal dérivé") ;
```

Remarquons que

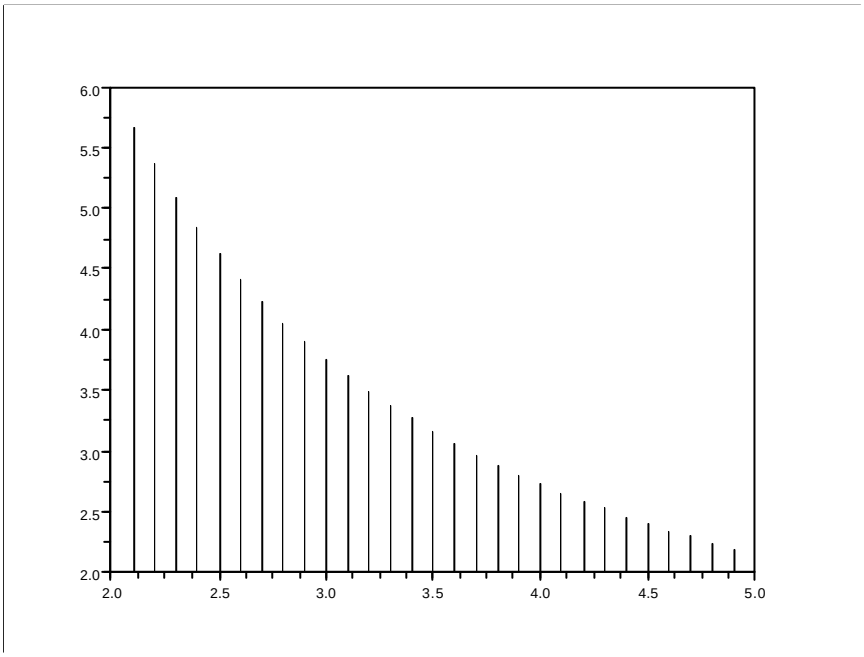
- nous avons, après calculs, supprimé les premiers termes de chaque signal, car, comme nous avons pu le constater précédemment, le premier terme de la réponse du filtre est pratiquement sans signification.
- en utilisant le script précédent avec changement de la valeur de T_e en 0.1 nous pouvons constater que les valeurs de y et celle de f sont «presque» proportionnelles et non égales ; en effet $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$ et donc $f'(nT_e) = \lim_{T_e \to 0} \frac{f((n+1)T_e) - f(nT_e)}{T_e}$. Par conséquent $f'(nT_e)$ est approché par $\frac{y_n}{T_e}$ lorsque y_n est la réponse du filtre dérivateur.
- Enfin, après exploration de ce script, nous remarquons qu'il est préférable de prendre comme intervalle d'étude un intervalle inclus dans le domaine de définition de la fonction (ce qui parait évident !).

Pour $a = 2, b = 5, T_e = 0.1$ nous obtenons



Une étude des erreurs relatives nous donne (en pourcentage) :

```
y=(1/Te)*y;
clf,plot2d3(t,(y-f)./y*100)
```



soit une erreur maximale de 6%, ce qui tout compte fait n'est pas si mal !

Le lecteur pourra se convaincre qu'en échantillonnant à ***100 Hz***, c'est-à-dire avec un pas d'échantillonnage de ***0.01***, les résultats sont encore bien meilleurs !

Pour pouvoir faire cette étude avec d'autres fonctions, il est possible de modifier le script précédent en modifier les lignes définissant le signal d'entrée (changement de la fonction ***f***) et le signal dérivé (changement de la fonction ***f'***). Il est aussi possible de modifier légèrement ce script pour en faire une fonction que nous nommerons «efd» pour «étude du filtre dérivateur» admettant pour paramètres (ou arguments) les valeurs a et b des bornes de l'intervalle d'étude, la période d'échantillonnage, les fonctions ***f*** et ***f'*** (on notera que si Mupad sait calculer une dérivée formellement, il n'en est pas de même pour Scilab) et le nom du signal (pour l'affichage du titre).

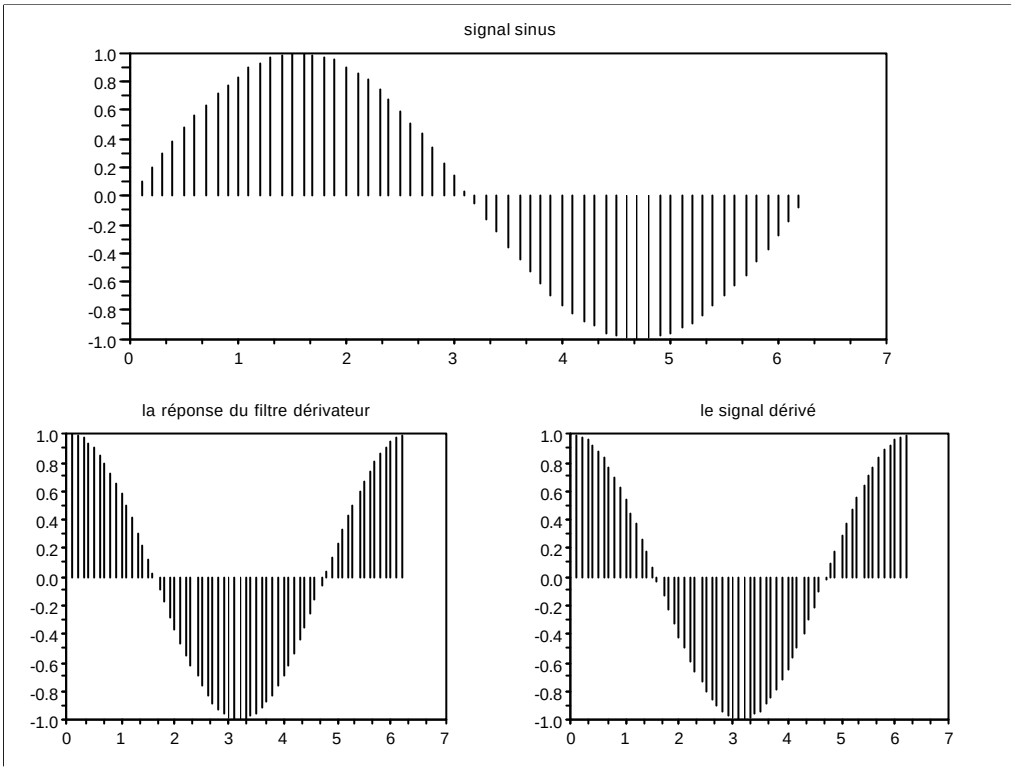
Ce qui nous donne

```
function y=efd(a,b,Te,f,fp,nom)
t=a:Te:b;
n=length(t);
x=f(t); //signal d'entrée
y=fder(x); // réponse du filtre
d=fp(t);//dérivée théorique échantillonnée
//suppression du premier terme
t=t(2:$);x=x(2:$);y=y(2:$);d=d(2:$);
y=(1/Te)*y
clf
//tracé du signal de départ
xsetech([0,0,1,1/2]);
plot2d3(t,x)
xtitle("signal "+nom);
// tracé de la réponse du filtre
xsetech([0,1/2,1/2,1/2]);plot2d3(t,y);
xtitle("la réponse du filtre dérivateur") ;
// tracé de la dérivée échantillonnéé
xsetech([1/2,1/2,1/2,1/2]);plot2d3(t,d);
xtitle("le signal dérivé") ;
endfunction
```

On remarquera que cette fonction renvoie le signal dérivé divisé par la période d'échantillonnage.

Nous obtenons alors,

- par **efd(0,2*%pi,0.1,sin,cos,'sinus');**



- par **efd(0,2*%pi,0.1,exp,exp,'exponentiel');**