IUT de Nîmes GEII
2017/18

OL3 - Traitement numérique du signal
TP 1 (1ère partie)

Représentations d'un signal dans les domaines temporel et fréquentiel
1 – Représentation d'un signal dans le domaine temporel

Le but des séances de TP en **OL3** est de vous permettre de maîtriser les notions fondamentales du traitement du signal (échantillonnage, filtrage, transformée de Fourier,...) en les simulant à l'aide du logiciel de calcul libre **Scilab**. Ainsi, vous allez approfondir votre maîtrise du logiciel Scilab, avec lequel vous avez travaillé en 1ère année dans le cadre du module **OL2**, par exemple.

Rappels :

Après avoir lancé l'application Scilab, cliquez sur "**SciNotes**" dans le menu "**Applications**". Ceci ouvre une fenêtre d'éditeur de texte, dans laquelle, vous allez écrire vos programmes. Vous pourrez ensuite sauvegarder ces scripts dans des fichiers ".sce" afin de les réutiliser ou de les modifier ultérieurement. Vous allez donc devoir réutiliser toutes les notions vues lors des TP d'OL2, à savoir les variables, les boucles, les structures conditionnelles, les graphiques, etc... S'ajoute à cette liste, une grande bibliothèque de fonctions mathématiques très utiles dans le traitement des nombres complexes, matrices, équations différentielles, polynômes, statistiques, etc, que nous découvrirons au fur et à mesure.

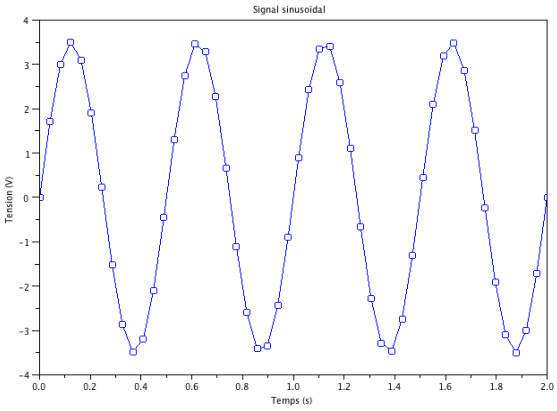
Pour des rappels sur la syntaxe de Scilab, vous pouvez consulter les énoncés des TP effectués en OL2, ainsi que le mémo, qui sont disponibles sur http://geii.iut-nimes.fr/content/cours-en-ligne,

REMARQUE : N'oubliez pas de faire des sauvegardes régulières de vos scripts, par exemple sur une clé USB personnelle, etc.

1.1 - Signaux alternatifs

Scilab est un logiciel de calcul numérique, qui manipule uniquement des nombres. Ce qui veut dire que ce logiciel ne sait pas tracer la fonction "sin(x)" littéralement. Il faut dans un premier temps stocker dans un tableau 1D toutes les valeurs numériques des abscisses x (ce qui implique de les ranger dans un vecteur). Ensuite, on calcule pour chacune de ces valeurs, l'ordonnée du point correspondant (ici, $y_i=\sin(x_i)$, avec i l'indice du tableau), et on les stocke dans un deuxième vecteur. Enfin, on demande à Scilab de tracer sur un graphique l'ensemble des points dont les couples abscisse/ordonnée sont rangés dans les 2 tableaux $\boldsymbol{x}$ et $\boldsymbol{y}$. La fonction est donc tracée point par point.

Tapez le programme suivant et exécutez-le :




```
clear;
clf;
nbpts=50;
duree=2;
t=linspace(0,duree,nbpts);
E=3.5;
f=2;
Vs1=E*sin(2*%pi*f*t);
plot(t,Vs1,'-ob');
xtitle("Signal sinusoïdal","Temps (s)","Tension (V)");
```

Pour chacune des 4 variables : ***t, E, f*** et ***Vs1***, précisez ce que représente la variable, combien de valeurs elle contient et quelles sont ces valeurs. On souligne que dans la console Scilab, on peut afficher le contenu d'une variable en écrivant simplement son nom.
Augmentez la valeur de **nbpts** à 200, que voit-on ? Baissez la valeur à 20, puis 10. Conclusion.
Remettez nbpts=50, et changez la durée à 5 ou 10. Conclusion.
Quelle précaution faut-il prendre pour visualiser correctement un signal sur le graphique ?

Dans la ligne de commande "**plot**", supprimez la variable ***t***. Que représente alors l'axe des abscisses ? Remplacez l'expression '-ob', par '-b' puis par 'ob'. Vous pouvez changer la couleur de la courbe avec 'r' pour 'red', 'g' pour 'green', 'b' pour blue, etc...

Dans l'exemple ci-dessus, le vecteur temps a été créé à partir d'une valeur initiale, de la durée totale et du nombre de points. Une autre façon de générer un vecteur identique est de donner la valeur initiale, la valeur finale, et le pas (l'incrément). Quelle relation lie le pas temporel à la durée et au nombre de points ? Attention, pour ***N*** points, il n'y a que ***N-1*** pas !!
Remplacez la ligne correspondante et vérifiez. La syntaxe est :

```
t = valeur_initiale : pas : valeur_finale
```

Créez un nouveau signal ***Vs2***, qui représente un cosinus d'amplitude 2 et de fréquence 3 Hz. Affichez-le avec le premier signal, mais avec une couleur différente. Ajoutez un 3ème signal sinusoïdal identique au premier signal ***Vs1***, mais retardé de π/4.

Exercices :

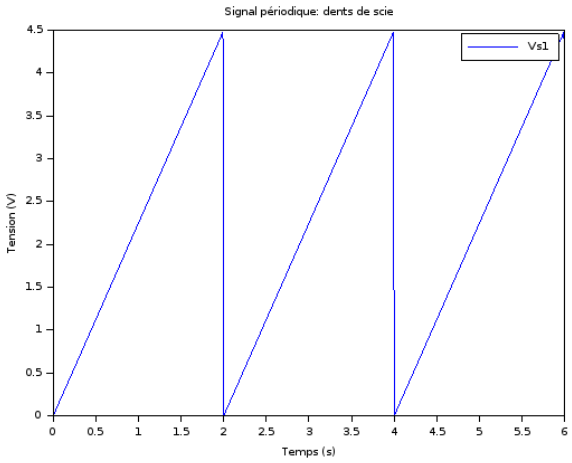
Tracer les signaux suivant en regroupant les graphiques sur une seule et même figure en utilisant la fonction **subplo**t().

- Le signal ***Us1*** d'expression $2\cos(2\pi f t)$ avec f=50 Hz.
- Le signal ***Us2*** égal à la valeur absolue de ***Us1.***
- Le signal ***Us3*** égal au double de ***Us1*** si les valeurs de ***Us1*** sont positives, et égal à zéro sinon.
- Le signal ***Us4*** égal à 1.5, si les valeurs de ***Us1*** sont positives ou nulles, et égal à -1.5, sinon.
- Le signal ***Us5*** égal a $\frac{2}{\pi}\cos^{-1}\left(\frac{Us1}{2}\right)-1$

Remarque :
Les signaux définis ci-dessus étaient sinusoïdaux ou pouvaient être définis par rapport à un signal sinusoïdal. De manière plus générale, la représentation graphique d'un signal périodique non-sinusoïdal sur plusieurs périodes peut être obtenue en suivant d'autres approches.

Par exemple, le programme ci-dessous permet de tracer un signal en dents de scie.

```
clear;
clf;
pas=0.01;
T=2;
E=4.5;
t=0:pas:3*T;
Vs1=E*modulo(t,T)/(T);
plot(t,Vs1,'b-');
xtitle("Signal périodique: dents de scie","Temps
(s)","Tension (V)");
legend("Vs1")
```



Recopiez le programme. Essayez de bien comprendre le rôle de chaque variable et la façon dont le signal ***Vs1*** est défini sous Scilab. Augmentez d'un facteur 10 la valeur du pas. Qu'observez vous ?

1.2 – Signaux périodiques usuels

A l'aide de Scilab, on se propose de tracer les représentations graphiques des signaux étudiés en **TD1 du module de mathématiques MA3**, exercices 1 à 4 (voir l'énoncé de ce TD sur http://geii.iut-nimes.fr/content/cours-en-ligne).

Consignes :
- Pour le signal ¨rectangle¨ prenez un rapport cyclique égal à 30 % et une période de 2s.
- Pour le sinus redressé prenez *a=2*.
- L'ensemble des tracés seront regroupés sur une seule et même figure en utilisant la fonction **subplo**t().
- Soigner la présentation : titres, axes, légende, etc...

1.3 – Signaux quelconques non périodiques

Tous les signaux électriques ne sont pas périodiques dans le temps, on peut donc être amené à simuler des signaux de forme quelconques. En vous aidant des paragraphes 1.1 et 1.2, tracez les signaux suivants :

- décharge d'un condensateur préalablement chargé (valeurs libres)
- impulsion de 2 V de forme rectangulaire et de largeur 10 µs
- un échelon de 1 V retardé de 0.5 s
- une rampe de tension de 10 V/s

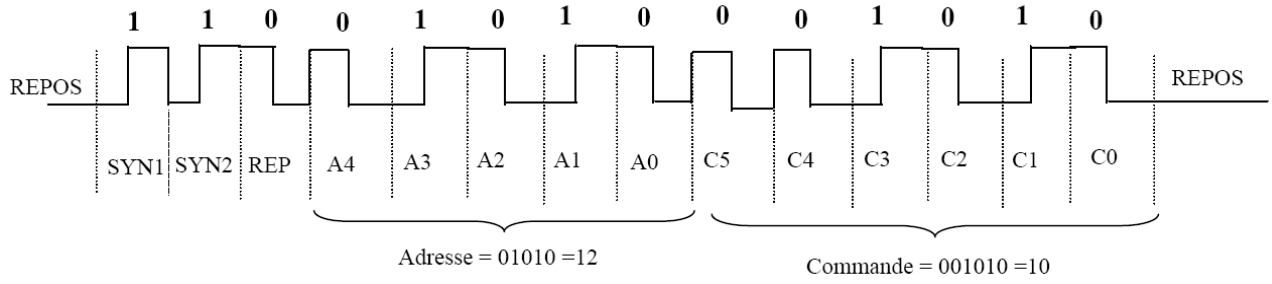
Regroupez ces tracés sur une seule et même figure en utilisant la fonction **subplo**t().

1.4 – Application : trame RC5

La plupart des appareils électroniques actuels (téléviseur, chaîne hifi, box, etc) sont équipés d'une télécommande. Lorsqu'on appuie sur un bouton de la télécommande, une trame est générée et est envoyée par une LED (signal dans l'infrarouge, non visible !). Du côté du recepteur, une photodiode capte le signal et le décode. Le protocole le plus utilisé pour la communication infrarouge est le code RC5, développé initialement par la société Phillips.

Une trame RC5 est composée d'un « mot » de 14 bits défini comme suit :
– 2 bits de départ servant à la synchronisation du récepteur
– 1 bit de répétition qui change d'état chaque fois qu'une nouvelle touche est actionnée (si une touche est maintenue enfoncée, ce bit reste inchangé)
– 5 bits d'adresse, qui permettent de coder 2^5=32 adresses différentes. Chaque appareil ayant sa propre adresse, ceci permet de modifier le volume de la télé sans changer le volume de la chaine hifi, par exemple ! Quelques codes d'adresses sont donnés en annexe.
– 6 bits de commande, soit 2^6=64 instructions différentes programmables, associées à chacune des touches de la télécommande (volume, programme, télétexte,...). Voir en annexe.

Exemple de trame de code RC5 :



Les bits du code RC5 sont codés en biphasé (codage Manchester), c'est-à-dire qu'un bit est constitué de la succession de 2 demi-bits de niveaux opposés:
– un niveau bas suivi d'un niveau haut (c'est-à-dire un front montant) = bit 1
– un niveau haut suivi d'un niveau bas (c'est-à-dire un front descendant) = bit 0

La durée d'un bit est de 1,778 ms, donc chaque demi-bit dure 889 µs. La durée totale d'une trame RC5 vaut environ 14x1,778 = 24,89 ms. Les trames se succèdent avec un temps de 88,9 ms équivalent à une durée de 50 bits.

On souhaite réaliser sous Scilab un générateur de trame RC5. Pour cela :

1- Ecrire une fonction qui génère un vecteur temps proportionnel au nombre de bits. On prendra un intervalle de temps égal à 0.1 ms.

2- Ecrire une fonction qui génère un bit 0 ou 1. On arrondit la durée d'un demi-bit à 0.9 ms.

3- Ecrire une fonction qui génère les 5 bits d'adresse à partir de l'adresse exprimée en décimal. On

utilisera la fonction bitget(nb,i) qui renvoie la valeur du bit à la position i du nombre décimal nb.

4- Ecrire une fonction qui génère les 6 bits de commande à partir de la commande exprimée en décimal.

5- Ecrire une fonction qui génère la trame complète RC5 (14 bits), avec le bit de répétition, l'adresse et la commande. Tester la fonction.

Annexes

Code RC5 des adresses de quelques appareils

Adresse	Appareil
00000=00	TV1
00001=01	TV2
00010=02	Télétexte
00101=05	Magnétoscope (VCR1)
01000=08	Récepteur satellite (SAT1)
01100=12	Lecteur DVD
10000=16	Préampli audio
10001=17	Tuner
10010=18	Magnétophone
10100=20	Lecteur CD audio

Code RC5 de quelques instructions en mode TV1

Instruction	Signification
000001=01	Chaîne 1
000010=02	Chaîne 2
000011=03	Chaîne 3
000100=04	Chaîne 4
000101=05	Chaîne 5
000110=06	Chaîne 6
000111=07	Chaîne 7
001000=08	Chaîne 8
001001=09	Chaîne 9
001100=12	Standby
001101=13	Mute
010000=16	Volume +
010001=17	Volume –