PROJET : Mise en Place d'une chaîne de transmission sous l'environnement SCICOS

MODULE – TRC6

I) Présentation de la problématique :

SCILAB/SCICOS est un environnement de programmation libre et gratuit dont l'interface se rapproche de MATLAB/SIMULINK.

SCICOS permet de réaliser une programmation par objet facilitant ainsi la mise en place d'un module de communication. Néanmoins la toolbox communication n'est pas implantée par défaut, il faut donc la charger dans SCICOS pour utiliser les palettes configurées dans le cadre de notre projet.

Il est à noter que les séquences de TPs sont libres dans le sens ou vous pouvez installer SCICOS et les librairies associées chez vous.

Néanmoins, la note prendre en compte l'avancé de votre travail et le guide technique portant sur l'utilisation de votre chaîne de transmission.

Découpage des séances de TPs

Le développement de la chaîne de transmission est défini sur une période de 18 heures de TP (6 séances). Les trois premières heures seront réservées à la prise en main du logiciel. Au cours des 5 autres séances vous travaillerez en autonomie mais en essayant tant que possible de respecter l'avancé du projet défini ci-dessous.

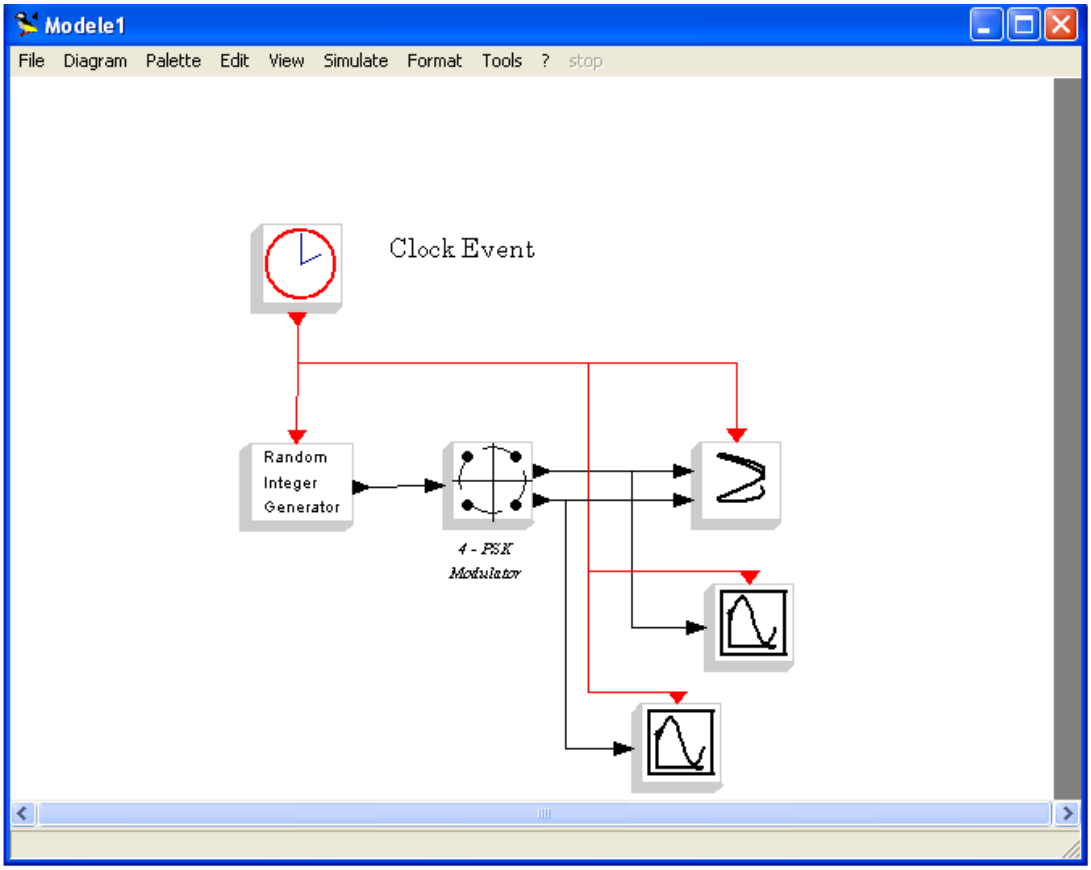
II) Initiation à Scilab/Scicos.

1) Ouvrez l'application scilab.
2) Créer un répertoire C:/DataScilab
3) Récupérez sous Y/ …/ launay/TRC6/Programme SCILAB/modnum.exe
4) Installez le (executable). Si vos droits sont insuffisants, copier le répertoire Modnum et copiez le dans le répertoire indiqué dans le 2.
5) Onglet Fichier -> Exec : loader (sous le répertoire ModNum4)
6) Dans la fenêtre de travail, tapez la commande → scicos()
7) Surfez sur le site www.scilab.org puis sur www.scicos.org.

Réalisation pratique

APPLICATION 1 :

Afin de se familiariser avec le logiciel, nous allons réaliser un premier schéma de conception se composant d'une source aléatoire, d'un modulateur QPSK et de fenêtre de visualisation temporelle.



Le schéma se compose d'une horloge d'événement qui active chaque bloc et des blocs suivants :

- Générateur de nombre aléatoire
- D'un modulation M-PSK, avec M=4
- De fenêtre de visualisation temporelle

On trouvera ces éléments dans les palettes Mod Num Source, Mod Num Communication et sink.

Les entrées/sorties sont représentées par un trait noir, obtenu en double cliquant sur la sortie du bloc concerné. Les traits rouges représentent les évenements.

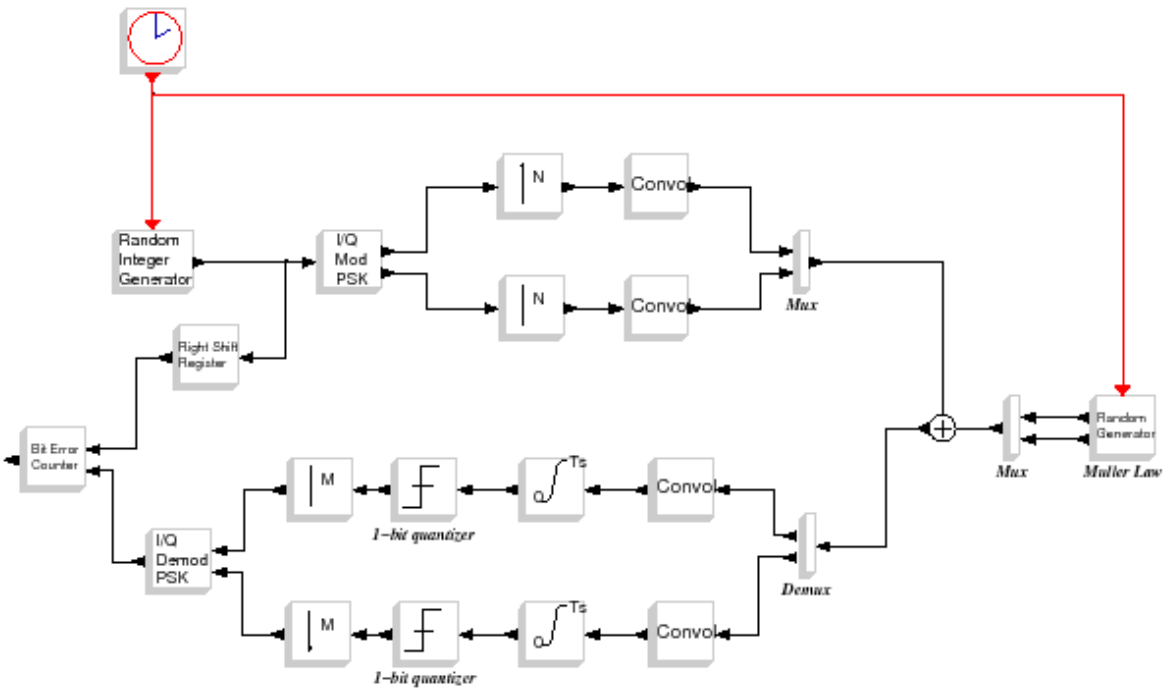
Pour lancer la simulation :
- Onglet simulate -> Compile
- Onglet simulate > Run

Vous devriez observer la sortie temporelle I et Q, et la constellation sur 4 points.
Dans le cas ou la constellation n'est que sur 2 points, ouvrez la fenêtre de commande de la source (Random Integer Generator) en double cliquant dessus et dans l'éditeur de commande *Type*, entrez la valeur 1.

APPLICATION 2 :

Nous allons apprendre dans cette application à exécuter scicos sous scilab et récupérez les résultats de la simulation sous Scilab.

Construisez l'application suivante.



NOUS ALLONS PROCEDER PAR PARTIE, d'abord l'émetteur progressivement et on validera lorsqu'on observera la séquence filtrée en émission. Puis après l'émetteur et filtre nyquist de reception, on observera la séquence filtrée.
On met ensuite en place le récepteur et au final le bruit.
Attention, le plus ne marche pas, on récupérera au final l'exemple qui sera sauvegardé.

La simulation portera sur 100 symboles (Nu), chaque symbole étant codé sur 2 bits (Nbit=2). La source devra donc générer un vecteur ligne de Nu symboles (ones(Nu,1), chaque symbole étant codé sur 2 bits (Nbits=2) soit un nombre de bit par entier =2*(ones(100,1)).
Chaque entier défini prend pour valeur 0 ou 1, on choisira donc un type =1 pour chaque symbole soit Type=1*ones(100,1) ;

La modulation choisir est une modulation QPSK, soit à 4 états possibles pour chaque symboles. Donc le nombre d'état total à générer est (2^Nbit)*ones(Nu,1).

Afin de filtrer le signal et améliorer la réponse du filtre, on va surechantillonner d'un facteur Nech=15 (On choisira un sur-échantillonnage 'frame-based' par insertion de zéros).

La convolution réalisé est un filtrage sur Nu bits pour I et Nu bits pour Q. la fonction de transfert du filtre est défini en temporel par :
4*r/pi*(cos((1+r)*pi*t_Ts) + (sin((1-r)*pi*t_Ts)./(4*r*t_Ts)))./(1-(4*r*t_Ts).^2); '
ou r=0,35 le facteur de roll off, et t_Ts la période d'estimation du filtre.
Il s'agit d'un demi filtre de nyquist : Vector of impulse reponse = pulse*gain_em.
Un filtre de Nyquist est non causal, on va le définir sur 127 points (Nb_coeff)
Ainsi, t_Ts=1/fe*(-nb_coef/2:nb_coef/2-1) représente le vecteur temps.

Si vous lancer le simulateur, vous ne visualiserez aucun résultat. Rajouter un oscilloscope et une application permettant de visualiser le diagramme de l'œil ;

Toutes ces variables sont enregistrées sous scilab avant de lancer la simulation sous scicos manuellement ou en exécutant un programme de paramètre :

```
scs_m.props.context=[
'Te=0.1;'
'Nu=16;'
'Nech=15;'
'N=Nu*Nech;'
'Nbit=2;'
'nb_coef=127;  '
'r=0.35; '
'fe=Nech; '
'gain_em=1; '
'gain_rec=1/Nech; '
'pulse=zeros(1,1); '
't_Ts=1/fe*(-nb_coef/2:nb_coef/2-1); '
'pulse = 4*r/%pi*(cos((1+r)*%pi*t_Ts) + (sin((1-r)*%pi*t_Ts)./(4*r*t_Ts)))./(1-
(4*r*t_Ts).^2); '
```

'Nb_vec=2^11; '
'nb_event=3;'
'Tfin=Te*Nb_vec'
'sigma=0.6;'
];
Ce programme sera nommé qpsk_teb_sim_ctxt

Random generator (muller law) se trouve dans la palette Modnul Source ;

L'application est cette fois ci lancée à partir de scilab (onglet fichier -> Exec). Récupérez le fichier sous Y:/.. TRC6/Programme SCILAB/qpsk_teb_sim.

Développement du projet

Vous allez réaliser une chaîne de transmission QPSK, vous allez procéder par étape afin de valider chaque maillon de la chaîne.

Etape 1 :

- Concevoir une source binaire aléatoire à 2 sorties I et Q (modulation_QPSK)
- Sur-échantillonner votre signal par un facteur M
- Sous Echantillonner votre signal par un facteur M ce qui revient à démoduler

Observer les résultats :

- Oscilloscope
- Analyseur de Spectre
- Diagramme de l'œil
- Taux d'Erreurs Binaires

Etape 2 :

- Réaliser un filtre de Nyquist
- Réaliser un demi filtre de Nyquist en émission et en réception
- Générer un canal de bruit additif Gaussien
- Rajouter des multi-trajets dans votre canal.

Etape 3 :

- Réaliser un démodulateur par la méthode du minimum de vraisemblance (cf. le récepteur conçu dans l'application 2 de la première séance de TP)
- Ajouter une technique de multiplexage par code au niveau de l'emetteur (Code de Walsh et code de Gold)
- Réaliser le récepteur

Etape 4 :

- Concevoir l'algorithme de Viterbi en émission et en réception
- Améliorer le récepteur par un récepteur Rake

-