Traitement Numerique du Signal

Luc Deneire

1 HW 1 : Espaces vectoriels L2 : application aux communications numériques

L'objectif principal de ce devoir est de manipuler des signaux de communication numériques en utilisant des modulations dites vectorielles. Vous allez donc générer des signaux "numériques", les tracer dans le domaine temporel et en tant que vecteurs (tracer les constellations). Vous les ferez passer dans un cnal à brui tbland additif (Gaussien), retracerez les signaux, et ensuite vous les décoderez.

1.1 Forme du devoir

Le devoir sera rendu sur Moodle sous la forme d'un notebook Jupyter.

1.2 La source

Comme source, vous utiliserez, pour l'affichage, votre nom et prénom. Vous transformerez ce texte en binaire, et ce flot binaire sera votre source.

1.3 Modulation vectorielle

Soient 4 types de modulation vectorielle basés sur (t étant continu, $\imath = \sqrt{-1}$).

$$c(t,\phi) = \sqrt{2}\cos(2\pi t + \phi), t \in [0,1], \qquad (= 0 \text{ sinon})$$

$$s(t,\phi) = \sqrt{2}\sin(2\pi t + \phi), t \in [0,1]0, \qquad (= 0 \text{ sinon})$$

.

1. Type de modulation 0 : les signaux $v_i(t) = e^{\imath i\frac{\pi}{4}}c(t,0), i = 0,\cdots 7$.
2. Type de modulation 1 : les signaux définis au TD 2, exercice 2.2.
3. Type de modulation 2 : les signaux $v_i(t) = (2*(i-3)-1)*c(t,0), i = 0,\cdots 7$
4. Type de modulation 3 : $v_i(t) = (c(t,0)+\imath s(t,0))*(1+2*(i//4)), i = 0,\cdots 7$, où $//$ représente la division entière.

Vous choisirez la modulation vectorielle comme étant le produit des codes ascii des premières lettre de votre prénom et de votre nom, modulo 4.

(exemple, pour Luc Deneire, le choix se porte sur la Modulation 0 ((ord('L')*ord('D'))modulo 4)

1.4 Génération des signaux

Vous découperez le flot binaire en autant de mots de “x” bits nécessaires (par exemple, ’L’ est représenté par 1001100, si on travaille sur 2 bits, on aurait les 3 premiers mots : ’10’, ’01’, ’10’).

Pour le flot binaire généré par votre nom, tracez :

- les signaux temporels des trois premiers mots (en séparant la partie réelle de la partie imaginaire)
- les constellations obtenues (voir ce qui a été fait en TD2)

1.5 Energie moyenne

Calculez théoriquement, et sous python, l’énergie moyenne utilisée par mot.

1.6 Addition du bruit blanc

Sur la partie réelle et sur la partie imaginaire, additionnez un bruit blanc, dans un premier temps, en utilisant une énergie par unité de temps 100 fois plus faible que l’énergie moyenne d’un mot, et ensuite en utilisant une énergie 2 fois plus faible que l’énergie moyenne d’un mot.

Pour rappel, vous pouvez utiliser np.random.normal(0,1,taille).

1.7 Tracé

Pour les deux rapports Energie du signal / Energie du bruit, effectuez les même tracés que précédemment.

1.8 Récepteur

Au récepteur, effectuez le produit scalaire entre le signal reçu et votre base. Sur base de ce produit scalaire, vous devez pouvoir décider quel mot a été envoyé (la procédure peut différer pour les différentes modulations, voir à l’oral en cours/TD).

1.9 Taux d’erreur de mots et de bits

Pour chacun des cas, estimez le taux d’erreur de mots (nombre de mots erronés divisé par le nombre de mots envoyés), ainsi que le taux d’erreur de bits.

Pour avoir un taux d’erreur qui soit significatif, il faut au minimum 100 erreurs. Faites en sorte d’avoir au moins 100 erreurs, en vous limitant à un nombre maximum de mots de $1e^6$. Pour obtenir une source plus “longue”, vous pouvez simplement envoyer le même texte, (mais en regénérant le bruit à chaque fois), un grand nombre de fois.

2 Intégration

Pour le produit scalaire, vous aurez besoin de faire une intégrale de fonctions. Un exemple est donné ci-dessous :

```
import numpy as np
import scipy.integrate


t=np.arange(0,1.0000001,0.01) # 1.00000001 car la ''fin'' est exclue
de arange

#définition des fonctions cc et ss

cc = lambda t: np.sqrt(2)*np.cos(2.0*np.pi*t)
ss = lambda t: np.sqrt(2)*np.sin(2.0*np.pi*t)


prod = scipy.integrate.simps(cc(t)*cc(t),t)

print(prod)

>>> 1.0
```