Séance V – Décryptage

Motivations et objectifs

Cette dernière séance se veut un peu plus ludique. Au cours de ce travail, vous aller utiliser vos connaissances en traitement du signal pour déchiffrer un message caché dans un signal aléatoire. Plusieurs stratégies sont utilisables suivant que vous supposez être ou non destinataire du message. Dans la première partie, vous supposerez être destinataire, et donc vous en possédez les clefs. Dans la seconde partie, vous essayerez de décrypter le message sans en connaître les clefs. Le premier message est un proverbe chinois, le second est extrait d'un très beau poème français.

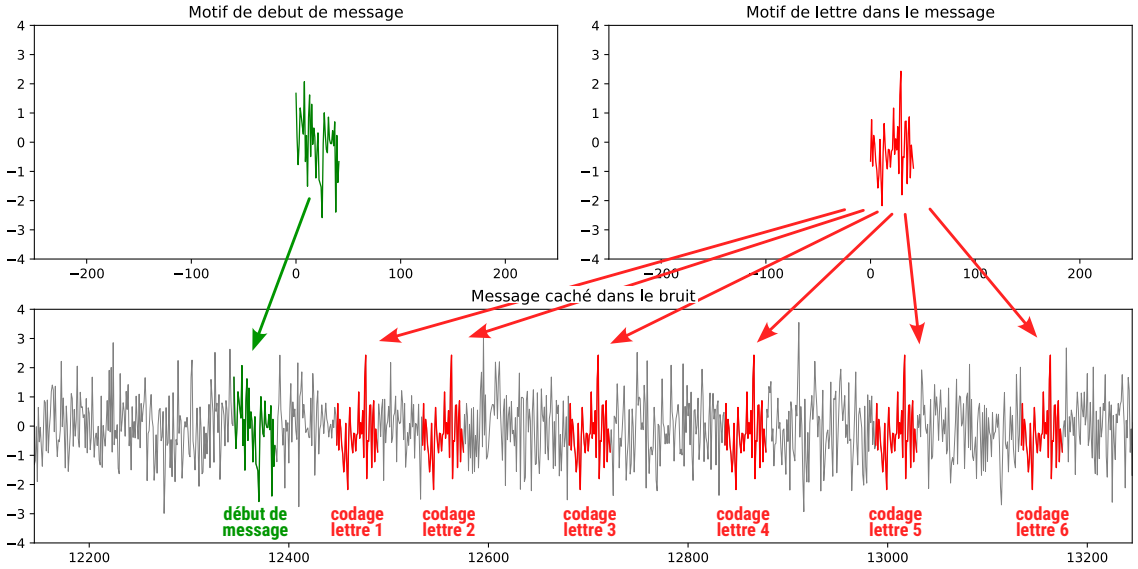
V.1 Présentation générale

V.1.1 Description du codage

Le principe du codage est très simple. On dispose d'une longue séquence de nombres aléatoires de distribution gaussienne et de deux séquences plus courtes ayant la même distribution. On appellera ces séquences courtes "motifs". Chacun de ces deux motifs est inséré dans la séquence plus longue à des positions qui codent le message à transmettre, comme on le voit sur l'illustration ci-dessous.

Pour créer un message, on insère d'abord le premier motif (en vert), une fois et une seule : sa position représente le début du message,

Ensuite, le message lui-même est encodé uniquement grâce au deuxième motif (en rouge). Ce message est composé d'une liste de lettres codées en ASCII (c'est à dire qu'à chaque lettre correspond un chiffre codé sur 8 bits et vice-versa). Pour coder le message, on insère de façon **presque** aléatoire le motif dans le segment le plus long. C'est la position où se trouve le motif rouge par rapport au début du message qui code une des lettres du message (comme représenté dans le dessin ci-dessous) en utilisant un nombre p de référence.



Par exemple la lettre `'A'` a comme code ASCII 65. Si je veux coder un A, je positionnerai le motif à une distance k du motif de départ, telle que le reste de la division entière de k par p donne 65. Puis je passe à la lettre suivante.

V.1.2 Principe du décodage

Pour décoder le message, il faut donc repérer les positions du deuxième motif, relativement au premier. Pour ce faire, la première stratégie nécessite la connaissance des deux motifs et du nombre p. Dans ce cas, il suffit de calculer la fonction de corrélation du premier motif par rapport au segment le plus long pour trouver la position de départ du message : on repère alors un maximum de corrélation, cette position sera le point de départ du message. Ensuite, on réalise une deuxième correlation entre le deuxième motif et le segment le plus long. On repère alors tous les maxima de la corrélation, ce sont ceux-ci qui codent le message. On calcule l'écart entre les positions de ces maxima et la position du début du message, puis, à partir des valeurs obtenues, on calcule la division entière de chacune de ces position avec le nombre p et on obtient le message codé.

Cette stratégie est appelée "stratégie renseignée", c'est à dire celle que vous employez si vous êtes le destinataire du message, puisque vous connaissez les deux motifs et la valeur de p.

Si vous n'en êtes pas destinataire et que vous ne connaissez pas la manière dont le code est formé, vous avez peu de chance de le découvrir car le message a une statistique pseudo-aléatoire. On suppose ici bien sur que vous connaissez la méthode de codage, mais que vous ne connaissez pas le segment codé, il vous faut essayer itérativement des tronçons de message et regarder si vous trouvez des répétitions de séquence en cernant au mieux la longueur de la séquence répétée. Si la longueur est mal identifiée, il se peut que les maxima détectés soient décalés, auquel cas vous ne pouvez pas retrouver le message. Il vous faut ensuite essayer différentes valeurs de p et choisir celui qui vous donnera le message le plus cohérent. Dans ce TP nous supposerons que p est connu ($p = 128$).

V.1.3 Fonction de corrélation

Pour pouvoir réaliser ce TP vous aurez besoin d'une fonction qui calcule la corrélation de deux signaux **de même taille** :

```
scipy.signal.correlate()
```

Remarques :

— Faites attention au fait que le produit de corrélation n'est pas commutatif ...

— Si les signaux ne font pas la même taille, il faut ajouter des zeros au plus court des deux.

V.2 Décodage

V.2.1 Données

Vous disposez de trois séquences, téléchargeables ci-dessous. La première contient le motif de début de message, La deuxième le motif qui code les lettres, et le dernier le signal avec le message :

https://dl.eea-fds.umontpellier.fr/PythonM1/mat/4/motif1.txt
https://dl.eea-fds.umontpellier.fr/PythonM1/mat/4/motif2.txt
https://dl.eea-fds.umontpellier.fr/PythonM1/mat/4/signalAvecMessage.txt

Après avoir téléchargé un fichier texte, vous pouvez le charger avec Python :

```
X = np.loadtxt(nom_du_fichier)
```

La valeur de la congruence est $p = 128$.

Problème V.1 : Décodage renseigné

A. Ecrivez une fonction dont l'appel sera :

```
message = DecodageMessage(signal_avec_message,pattern, cle)
```

[codeTP/codeTP-V-2-3.py]

où `cle` est la valeur de p.

Remarque 1 : Représentez vos signaux, y compris le résultat de la corrélation !

Remarque 2 : Le reste de la division entière de `a` par `b` s'écrit : `a%b`. Vérifier bien que vous avez des entiers. Pour transformer un caractère `mon_caractere (='A' par exemple)` en son code ASCII `code_ASCII` on écrit :

```
code_ASCII = ord(mon_caractere).
```

De même on écrit dans l'autre sens :

```
mon_caractere = chr(code_ASCII).
```

Attention ! ! ! le codage des positions sous Python commence à 0. Il faut en tenir compte pour décoder.

Remarque 3 : Vous pouvez utiliser comme seuil de détection des pics de corrélation, soit la valeur maximale de l'auto-corrélation de votre signal pour décalage faible (1 ou 2) en multipliant cette valeur par 1.2 par mesure de sécurité, soit prendre la moitié de sa variance centrée. ■

Problème V.2 : Décodage non renseigné

Ce cas est bien sur plus compliqué.

Pour le tester, vous disposez de la seule séquence `signal_avec_message` dans le fichier, vous ne disposez plus du "motif" ou "pattern". Vous ne connaissez pas la clé non plus.

Pour le décoder, vous devez d'abord l'analyser avec votre fonction de corrélation en essayant systématiquement de corréler le signal avec tous ses segments ayant une longueur comprise entre 60 et 140 (normalement il faudrait en tester plus bien sur). Vous pouvez bien sûr vous limiter à une portion réduite de la séquence `signal_avec_message` en supposant qu'il y a beaucoup de répétition. Le segment ayant donné la corrélation la plus forte est alors choisi comme pattern. Vous effectuez alors le décodage, en testant plusieurs valeurs de cle (p). Le segment `signal_avec_message` étant très long, essayez d'optimiser votre recherche. ■