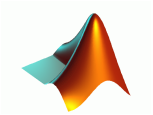


TRAVAUX PRATIQUES DE GÉNIE INFORMATIQUE

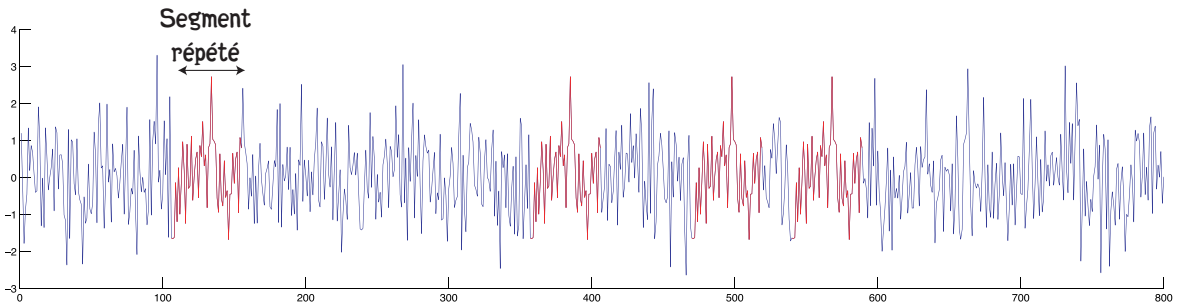
1 • But de la séance.

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 poême français.

2 • Matériel.

2.1 • Description du codage.

Le principe du codage est très simple. On dispose d'une longue séquence de nombres aléatoires de distribution gaussienne et d'une séquence plus courte ayant la même distribution. La séquence plus courte est insérée dans la séquence plus longue à des positions qui codent le message à transmettre. Ce message est composé d'une liste de lettres codés 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 segment plus court dans le segment le plus long. Chaque position où se trouve le segment le plus court 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 la séquence la plus courte à une position k telle que le reste de la division entière de k par p donne 65. Puis je passe à la lettre suivante.

2.2 • Décodage.

Pour décoder le message, il faut repérer les positions du segment le plus court. Pour ce faire, la première stratégie nécessite la connaissance du segment aléatoire le plus court et du nombre p. Dans ce cas, il suffit de calculer la fonction de corrélation du segment le plus court par rapport au segment le plus long. On repère alors les maxima de la corrélation, ce sont ceux-ci qui codent le message, on calcule la division entière

de chacune de ces position avec le nombre p et on obtient le message codé.
Cette stratégie est la stratégie renseignée, c'est à dire celle que vous employez si vous êtes le destinataire du message. 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).

2.3 • Fonction de corrélation.

Pour pouvoir réaliser ce TP vous aurez besoin d'une fonction qui calcule la corrélation de deux signaux **de même taille**. On vous fournit une fonction qui realise ce calcul. Elle est codée en C pour que son execution soit plus rapide. Vous devez cependant la compiler.
La procédure à suivre est la suivante :
1- recuperez le fichier Correlation.c sur le site,
2- placez le dans le dossier où vous réalisez vos expérimentations de travaux pratiques,
3- **compilez-le** en ecrivant dans la zone de texte :

```
>> mex Correlation.c
```

Appelez votre enseignant si vous avez des erreurs qui apparaissent.

3 • DÉCODAGE RENSEIGNÉ

3.1 • Données.

Vous disposez de deux séquences dans le fichier `signal_message.mat` :

```
>> load signal_message ;
```

qui sont `signal_avec_message` qui est la séquence longue et `pattern` qui est la séquence courte. La valeur de la congruence p est p=128.
Vous devez produire une fonction dont l'appel est :

```
>> [message] = DecodageMessage( signal_avec_message, pattern, cle ) ;
```

où `cle` est la valeur de p.

3.2 • Aide à la programmation.

Le reste de la division entière de `a` par `b` s'écrit : `mod(a, b)`. Vérifier bien que vous avez des entiers. Pour transformer un caractère `mon_caractere` en un entier `mon_entier` on écrit : `mon_entier = int(mon_caractere)` ; De même on écrit dans l'autre sens `mon_caractere = char(mon_entier)` ;
Attention !!! le codage des positions sous Matlab commence à 1. Il faut en tenir compte pour décoder ... sinon tous les caractères décodés seront décalés de 1.
Pour calculer la corrélation, le mieux serait d'utiliser la fonction que vous avez déjà

utilisé. Sinon, vous pouvez aussi utiliser
`mean(signal_avec_message(t:(t+length(pattern)-1)).*pattern)`
ou `t` est l'indice ou vous positionnez votre séquence courte par rapport à la longue.

3.3 • Seuil de détection.

Vous pouvez utiliser comme seuil de détection, 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.

4 • 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 `signal_crypte.mat` :

```
>> load signal_crypte ;
```

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 sûr). 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.