Master Mathématiques et Applications, 1-ière année, Aix-Marseille Université

Modélisation en Traitement du Signal, TP 1

Année 2015-16

---

1 Préliminaires

Les signaux analogiques sont des fonctions d'une variable continue ($t \in \mathbb{R}$, $t \in \mathbb{R}^+$ ou $t \in [0,1]$ pour les signaux unidimensionnels). Représenter de tels signaux sur un ordinateur est impossible pratiquement. Comment faire ? la seule solution est de tricher quelque peu, en les représentant par des signaux numériques (des vecteurs de taille finie) finement échantillonnés. On utilisera donc comme espace des signaux analogiques l'espace $\mathbb{C}^L$, où L est un (grand) entier positif, à la place de $L^2([0,1])$.

Par exemple, la séquence d'instructions MATLAB/OCTAVE

```
>> L = 2^10;
>> ttt = linspace(0,1,L);
>> nu = 64;
>> x = cos(2*pi*nu*ttt);
```

génère 1024 échantillons d'une sinusoïde de fréquence $\nu = 64$, sous forme d'un *vecteur ligne*, qu'il faut transposer pour en faire un *vecteur colonne*.

On représentera les signaux sous forme de vecteur colonne, et les bases par des matrices dont les colonnes sont les vecteurs de base. Se référer à la fiche de préparation du TP pour la version matricielle des calculs.

2 Projection orthogonale

Nous avons vu en cours que la projection orthogonale de $L^2([0,1])$ sur un sous-espace engendré par des translatées de N fonctions $\Phi = \{\phi_n,\ n = 0, \dots N-1\}$ s'écrit sous la forme

$$\Pi_F(x) = \sum_{n=0}^{N-1} \alpha_n \phi_n \ , \quad \text{avec} \quad \alpha_n = \langle x, \tilde{\phi}_n \rangle \ ,$$

où $\widetilde{\Phi} = \{\tilde{\phi}_0, \dots \tilde{\phi}_{N-1}\}$ est la *famille duale* (ou base biorthogonale) de la famille des ϕ_n, définie par

$$\tilde{\phi}_n = \sum_{m=0}^{N-1} \overline{h}_{nm} \phi_m \ ,$$

et les h_{nm} sont les coefficients de la matrice $H = G^{-1}$, G étant la *matrice de Gram* de la famille Φ donnée par $g_{nm} = \langle \phi_m, \phi_n \rangle$.

Exercice 2.1 *En remplaçant $L^2([0,1])$ par $\mathbb{C}^L$ (voir ci-dessus), écrire une fonction* `baseduale.m` *prenant en entrée une matrice $\Phi \in \mathcal{M}_{L,N}$ dont les colonnes sont N vecteurs de $\mathbb{C}^L$ (les vecteurs ϕ_n) et retournant la matrice $\widetilde{\Phi}$ de la famille duale. La fonction vérifiera au passage l'inversibilité de la matrice de Gram, et retournera un message d'erreur si la matrice n'est pas inversible.*

Exercice 2.2 *Dans le même cadre, écrire une fonction* `orthproj.m` *prenant en entrée la matrice Φ et un vecteur $x \in \mathbb{C}^L$, et retournant son projeté orthogonal $y \in \mathbb{C}^L$ sur le sous-espace de $\mathbb{C}^L$ engendré par les colonnes de Φ.*

3 Approximations constante et affine par morceaux

Les exercices ci-dessous utilisent des fonctions MATLAB/OCTAVE disponibles sur le site web du cours (archive `GenerateursDeBases.zip`). Des exemples de signaux, ainsi qu'une fonction permettant de les charger dans MATLAB/OCTAVE se trouvent aussi sur le site web (archive `signals.tgz`).

Pour chacune des bases (constante, affine, trogonométrique) on testera la fonction correspondante, on évaluera le conditionnement de la matrice de Gram et on tracera les éléments de la base biorthogonale.

3.1 Approximation constante par morceaux

Ici N est un entier positif, diviseur de L, et on s'intéresse au sous-espace $\mathcal{E}_0 = \mathcal{E}_0^{(N)}$ de $\mathbb{C}^L$ des vecteurs qui sont constants sur des intervalles entiers de longueur L/N. La fonction `BaseSplineConst.m` permet de générer la matrice Φ de la base de $\mathcal{E}_0$.

Exercice 3.1 *Ecrire une fonction* `ConstApprox.m` *prenant en entrée un signal $x \in \mathbb{C}^L$ et un entier positif N, et retournant son projeté orthogonal $y = p_{\mathcal{E}_0}(x)$ sur le sous-espace $\mathcal{E}_0$ correspondant, ainsi qu'une mesure de l'erreur d'approximation (par exemple la norme de la différence entre le signal original et le signal projeté, normalisée par la norme du signal original). La fonction tracera x et son projeté y (optionnellement).*

Exercice 3.2 *Tester la fonction ainsi obtenue sur des signaux synthétiques, et sur des signaux réels (sons) à télécharger sur le site web du cours. Les signaux synthétiques peuvent être générés suivant votre imagination. Des signaux réels peuvent être lus grâce à la fonction* `LitSignal.m`*. Les signaux de parole sont au format* `.wav`*, et peuvent être lus grâce à l'instruction* `wavread`*.*

Remarque 3.1 *Les tests sur signaux réels nécessitent quelques précautions. Les signaux réels sont souvent assez longs ; si des opérations matricielles sont effectuées, la longueur des signaux conditionne la taille des matrices,... et cela peut poser problème quand les matrices sont trop grosses. Il est donc préférable de travailler dans un premier temps sur des signaux relativement courts.*

3.2 Approximation affine par morceaux

Après les approximations par fonctions constantes par morceaux, on considère maintenant un sous-espace $\mathcal{E}_1 = \mathcal{E}_1^{(N)}$ de fonctions affines par morceaux. La procédure est essentiellement la même que plus haut. La fonction `BaseSplineAff.m` permet de générer la matrice Φ de la base de $\mathcal{E}_1$.

Exercice 3.3 *Ecrire une fonction* `AffApprox.m` *prenant en entrée un signal $x \in \mathbb{C}^L$ et un entier positif N, et retournant son projeté orthogonal $y = p_{\mathcal{E}_1}(x)$ sur le sous-espace $\mathcal{E}_1$ correspondant, ainsi qu'une mesure de l'erreur d'approximation (par exemple la norme de la différence entre le signal original et le signal projeté, normalisée par la norme du signal original). La fonction tracera x et son projeté y (optionnellement).*

Exercice 3.4 *Tester la fonction ainsi obtenue sur des signaux synthétiques, et sur des signaux réels (sons) à télécharger sur le site web du cours.*

4 Devoir et compte-rendu

Finaliser les exercices de cette fiche. Les devoirs sont à rendre (par binôme) sous forme d'une archive contenant

- Les fichier MATLAB/OCTAVE des fonctions développées ; celles ci devront être suffisamment commentées pour qu'un utilisateur puisse les utiliser sans effort.
- Un bref compte-rendu, comprenant pour chaque exercice
  - une description de l'approche suivie pour résoudre le problème posé
  - une description de la fonction programmée (syntaxe, pseudo-code,...) si nécessaire
  - des illustrations (sauvegardes de figures au format JPEG (extention `.jpg`) par exemple) si vous le jugez utile.
- En cas de difficultés pour inclure des figures dans le texte, on pourra plus simplement ajouter à l'archive les fichiers `.jpg`... et préciser dans le compte-rendu quelle figure on doit regarder.

Les devoirs sont à rendre sous forme d'une archive contenant tous les fichiers demandés, à envoyer par mail à `bruno.torresani@univ-amu.fr`