Chambre de Commerce et d'Industrie de Paris ———— ESIEE/ISBS	Unités : SE4 et B2ST12 TP filtrage adaptatif	

Remis par M. J.-F. BERCHER

ÉNONCÉ

Le but de ce TP est d'étudier et mettre en œuvre un algorithme LMS.

0.1 Programmation

Programmez un algorithme LMS. La syntaxe d'appel sera la suivante :

```
function [w, e] = lms(d ,w, u, mu);
% function [w, e] = lms(d ,w, u, mu);
% !! la recurrence sur le temps etant menee par appel a ce script.
%
% Variables d'entree :
% d  : séquence désirée
% w  : reponse impulsionnelle recherchee (a mettre a jour)
% u  : entree
% mu : pas d'adaptation
% Variables de sortie :
% w  : reponse impulsionnelle
% e  : erreur d'identification
```

Le "programme" devrait tenir en une à deux lignes de code.

0.2 Identification

Pour tester cet algorithme, on débute par un problème d'identification, où il s'agit d'identifier un filtre htest, à partir des observations de son entrée $x(n)$ et de sa sortie $y(n)$.

- Explicitez la mise en forme de problème sous forme Wienerienne.
- Vous utiliserez comme signal test la sortie d'un filtre excité par un bruit blanc, selon, par exemple

  ```
  x=randn(1,N);
  htest=10*[1 0.7 0.7 0.7 0.3 0 ];
  y=filter(htest, [1 ],x) + 0.01*randn(1,N);
  ```

  Mais vous avez le droit de choisir autre chose.
- Pour évaluer le comportement de l'algorithme, vous pourrez tracer une estimée de la variance de l'erreur, ainsi que l'erreur quadratique entre le filtre identifié et le filtre exact.
- Examinez la vitesse de convergence en fonction du pas d'adaptation : évaluez qualitativement le comportement de l'algorithme, pour des pas grands ou petits, et les capacités d'adaptation en introduisant une non-stationnarité lente dans le signal, par exemple selon

  ```
  u=randn(N,1);
  htest=10*[1 0.7 0.7 0.7 0.3 0 ]';
  for t=L:N
  y(t)=htest'.*(1+cos(2*pi*t/(N)))*u(t:-1:t-L+1);
  end;
  ```

0.3 Soustraction de bruit

– ***Soustraction de bruit***. Le signal que l'on cherche à estimer est s, à partir du mélange $x = s + b$, et de la référence bruit u. Vous cherchez à retrouver le signal s. Vous appliquerez donc une structure de soustraction de bruit, où le filtre sera identifié de manière adaptative à l'aide d'un algorithme LMS. Vous disposez pour vos expérimentations, de signaux tests contenus dans les fichiers `sb1.mat` et `sb2.mat`. Le premier fichier contient deux références bruit, l'une stationnaire, l'autre non stationnaire, que vous considérerez successivement. Vous prendrez des valeurs de pas comprises entre 0.01 et 1. Pour le second signal test, contenu dans le fichier `sb2.mat`, on a une non-stationnarité beaucoup plus importante et un rapport signal-à-bruit très défavorable. Il pourra être utile de traiter le problème en « deux passes », afin d'obtenir une première estimée de la réponse impulsionnelle du filtre, qui sera utilisée lors de la seconde « passe ».

– ***Soustraction de bruit et signal de parole*** On dispose d'un signal de parole perturbé par un cri d'insecte (il s'agit d'une decticelle bariolée). On dispose aussi du signal de l'insecte seul, pris par le deuxième microphone tout près de celui-ci.

```
%signaux disponibles
load parole_bruitee
load decticelle
```

Testez vos algoritmes sur ces signaux. Vous pourrez vous amuser à les écouter à l'aide de la fonction soundsc.

Syntaxe d'appel :

```
function [s,h]=sousb(ref,signal,h_ini,mu);
% function [s,h]=sousb(ref,signal,h_ini,mu);
% Algorithme de soustraction de bruit :
% Entrées :
%  ref : reference bruit seul
% signal : voie signal (signal composite s +b, ou b est corrélé avec ref
% h_ini : réponse impulsionnelle initiale
%  mu : pas d'adaptation
% Sortie :
% s : signal identifié par soustraction de bruit
% h : réponse impulsionnelle identifiée
%
```