Simulations pédagogiques en traitement du signal avec JavaScript

Vincent MAZET

ICube, Université de Strasbourg – CNRS, 67412 Illkirch, France
vincent.mazet@unistra.fr

Résumé – Les concepts du traitement du signal ou des communications numériques ont parfois besoin d'être expliqués à l'aide de simulations ou d'animations interactives. Nous montrons que les langages HTML5 et JavaScript sont efficaces pour réaliser ce type de support pédagogique : ils apportent une solution gratuite, libre, simple d'utilisation, accessible depuis n'importe quel appareil, moderne et très maintenue. À ce titre, nous avons développé plusieurs simulations et trois d'entre elles sont présentées dans cet article. Nous pensons que l'utilisation et le développement de simulations pédagogiques est de nature à intéresser les participants du Gretsi.

Abstract – Interactive simulations and animations are very useful for clarifying some concepts in signal processing or digital communications. We show in this article that HTML5 and JavaScript are very efficient for producing this kind of educational support: they allow to develop programs that are free, with no cost, user-friendly, available on any device, up-to-date and very well maintained. Therefore we have developed several simulations, three of them being presented in this paper. We think that the use and developing of pedagogical simulations should be of great interest to the people involved in the Gretsi community.

1 Introduction

L'apprentissage du traitement du signal ou des communications numériques demande aux étudiants d'appréhender des concepts qui peuvent être difficiles à enseigner dans le schéma classique des exposés magistraux, travaux dirigés ou pratiques. Il est parfois nécessaire d'illustrer ces concepts à l'aide de schémas interactifs, ce qui a amené ces dernières années au développement de simulations pour ces enseignements.

Une simulation a l'avantage de présenter de manière concise un concept, en permettant à l'utilisateur de modifier certains paramètres pour en comprendre l'influence. Une simulation permet également d'effectuer des animations. Enfin, une simulation peut être ludique, ce qui est un avantage certain pour l'enseignement de concepts difficiles.

Une simulation peut être présentée et commentée par un enseignant durant un enseignement en présentiel, par exemple en utilisant un logiciel de calcul scientifique comme Matlab. Il faut alors développer une interface graphique pour rendre la simulation attractive et claire. Cependant, le logiciel ne permet pas toujours d'être très interactif. Par exemple, il n'est pas possible en Matlab d'interagir directement sur une courbe : on est obligé d'utiliser des contrôles tels que des boutons, ce qui réduit l'ergonomie. Par ailleurs, le logiciel doit être installé et peut être payant, limitant alors son utilisation par un étudiant.

C'est pour ces raisons que les simulations accessibles depuis un navigateur web sont intéressantes : elles sont gratuites, accessibles généralement sans téléchargement et permettent de réaliser des interfaces interactives. Nous montrons dans cet article que l'utilisation des technologies récentes du web (HTML5 et JavaScript) remplissent ces conditions et ouvrent la possibilité de créer des simulations pédagogiques. Cette solution ne nécessite aucune installation, est gratuite, libre, répandue, utilisable sur tout type d'appareil (y compris tablettes et téléphones), maintenue et très documentée. Par ailleurs, des librairies se développent actuellement pour faire du calcul scientifique.

La section 2 de cet article résume les différentes technologies existantes pour réaliser des simulations pédagogiques dans le domaine du traitement du signal. La section 3 présente le principe des langages HTML5 et JavaScript, et leur intérêt pour la réalisation de simulations. Nous décrivons en section 4 le fonctionnement de notre approche, notamment par l'intermédiaire de quelques exemples. Enfin, la section 5 conclut l'article.

2 Solutions existantes

Une première possibilité pour développer des simulations scientifiques est bien sûr d'utiliser les outils spécialisés. Ainsi, Matlab (MathWorks) [9] offre un grand nombre de possibilités, en commençant par les démonstrations disponibles dans les *toolboxes Signal Processing* [14] et *Image Processing* [13]. D'autres exemples peuvent être obtenus en téléchargeant les exemples proposés par les utilisateurs, et disponibles notamment sur le *File Exchange* [4]. De la même manière, Mathematica (Wolfram) [7] permet également d'utiliser et de concevoir des démonstrations [18]. Sans avoir à installer Mathematica, on peut aussi utiliser le lecteur gratuit Wolfram CDF Player [11] pour visualiser les documents interactifs écrits avec Mathematica. Tous ces logiciels profitent de la puissance de leur langage pour faire presque tout. Cependant, ils souffrent de deux inconvénients majeurs. Le premier est qu'il est nécessaire d'installer un logiciel, parfois payant (le prix n'est souvent pas compatible

avec les ressources financières estudiantines). Le deuxième inconvénient est que les possibilités d'interactivité sont assez limitées car il n'est pas toujours possible d'interagir directement sur une courbe sans utiliser des contrôles (boutons...).

Notons également que des simulations ont été développées en VBA pour Excel (Microsoft) [1]. Cependant, les possibilités sont très limitées, tant au niveau des possibilités de simulation que de l'interactivité.

Le langage Java a alors constitué une très bonne alternative, d'autant plus qu'elle a permis l'apparition de solutions libres et gratuites. Ainsi, GeoGebra est une application dédiée à la réalisation de simulations interactives scientifiques [5]. Il existe maintenant une version en ligne dont l'utilisation ne nécessite pas d'installation. GeoGebra permet de manipuler des graphes mais plus difficilement des contrôles. Par ailleurs, Java peut être utiliser pour réaliser des applets [19] depuis un navigateur web. On peut ainsi citer les nombreux applets développés par Rugh et Crutchfield [2, 3, 15] ou Spanias [16] pour illustrer des concepts de traitement du signal et de théorie du contrôle. Le problème de cette solution est que la maintenance des lecteurs (*plug-ins*) Java des navigateurs est désormais abandonnée en raison de problèmes de sécurité [10].

Pour toutes les raisons précédentes, nous avons développé des simulations interactives écrites à l'aide des langages HTML5 et JavaScript, décrits dans la section 3 :

— ces langages sont libres (non propriétaires) : chacun peut développer librement et améliorer des codes existants ;
— ces langages sont gratuits, pour le développeur comme pour l'utilisateur ;
— ces langages ayant été créés pour le web, aucune manipulation n'est nécessaire pour les faire fonctionner ;
— ces langages étant très répandus, ils sont correctement maintenus et extrêmement bien documentés par une importante communauté d'utilisateurs ;
— enfin, ces langages fonctionnent sur n'importe quel appareil (ordinateur, tablette ou téléphone), ce qui est d'autant plus intéressant pour des étudiants.

3 HTML5 et JavaScript

3.1 HTML5 et l'élément canvas

HTML (*HyperText Markup Language*) est un langage conçu pour écrire des pages web. HTML5 en est la dernière version et date de fin 2014. C'est un langage à balise, c'est-à-dire que chaque élément d'une page web est décrit à l'aide de mot-clés. Entre autres, la balise `<canvas>` a été introduite dans HTML5 : elle définit des zones graphiques (pour afficher des formes géométriques et des images), interactives (l'utilisateur peut interagir avec la souris ou le doigt sur les éléments qu'elle contient), et dynamiques (elles peuvent être modifiées même après l'affichage de la page web). Des exemples de possibilités offertes par cette balise sont présentées dans le site www.chromeexperiments.com/.

3.2 JavaScript

JavaScript est un langage interprété et orienté objet écrit en 1995. Malgré son nom, JavaScript est totalement indépendant de Java. Il est principalement utilisé pour générer de l'interactivité dans les pages web ; son usage tend toutefois à se diversifier depuis quelques années et les sites web les plus importants l'utilisent.

Le code est interprété par le navigateur web du client indépendamment du système d'exploitation et de l'appareil, et ne nécessite pas d'installation. Comme il est devenu très utilisé, les interpréteurs sont devenus très rapides (comme par exemple Chrome V8 [17]). Par ailleurs, c'est en JavaScript qu'on peut modifier le contenu de l'élément `<canvas>` d'une page web. Enfin, il est possible d'afficher des graphes en trois dimensions (par exemple pour représenter une exponentielle complexe [1]). Ces raisons nous ont fait préféré JavaScript à d'autres langages de programmation web (tel PHP) pour le développement de simulations sur le web.

Or, du point de vue d'un enseignant en traitement du signal, JavaScript est très bas-niveau. Par exemple, il n'existe pas de fonction pour afficher un signal dans un graphe. C'est la raison pour laquelle nous avons écrit le script spetsi.js (décrit section 4.1), qui rend la programmation de simulations en traitement du signal plus aisée. Nous avons ainsi codé les fonctions de base pour représenter un signal, en essayant d'être aussi proche que possible de la syntaxe de Matlab, qui est très utilisé dans la communauté.

Enfin, il serait difficile d'enseigner le traitement du signal sans présenter des équations mathématiques. Or, HTML ne permet pas d'afficher correctement des symboles et équations mathématiques. Aussi, la libraire MathJax [8] a été développée en 2009 pour écrire des mathématiques avec du code LaTeX interprété par un script JavaScript au chargement de la page web.

4 Description de la solution proposée

Le script spetsi.js, qui met à disposition des fonctions pour faciliter le développement de simulations, est décrit section 4.1. Nous présentons ensuite trois exemples de simulations : la représentation d'une sinusoïde (section 4.2), l'illustration du produit de convolution (section 4.3) et l'algorithme de Huffman pour le codage source (section 4.4). Ces simulations ainsi que neuf autres sont accessibles à cette adresse :

http://miv.u-strasbg.fr/mazet/animations/

Les codes sont également téléchargeables à cette même adresse.

4.1 spetsi.js

Le script spetsi.js regroupe plusieurs fonctions JavaScript pour manipuler les objets de l'interface ou effectuer des calculs.

1. cf. miv.u-strasbg.fr/mazet/animations/expo.html.

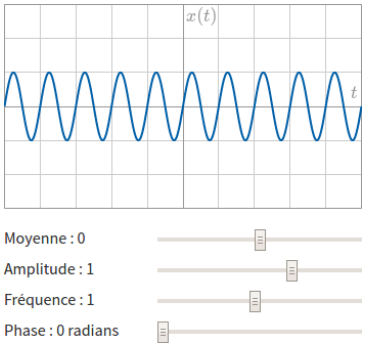

FIGURE 1 – Représentation d'un signal sinusoïdal.

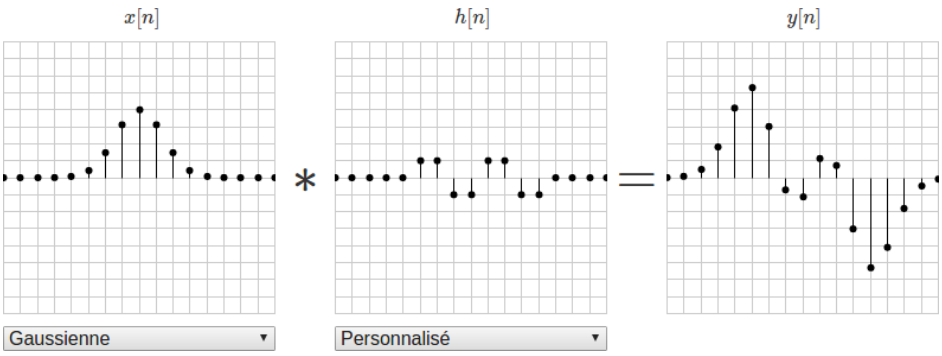

FIGURE 2 – Convolution de signaux en temps discret.

Des exemples d'utilisation sont disponibles :

http://miv.u-strasbg.fr/mazet/animations/sandbox.html

Il est ainsi possible d'intégrer dans l'interface de la simulation des étiquettes de texte (`Label`), des curseurs de défilement (`Slider`), des listes de choix (`Select`), des boutons (`Button`), ou des zones de texte (`Edit`). Par exemple, la commande

```
Button(div, txt, x, y, w, h, tip, f);
```

crée un bouton sur l'élément `<canvas>` (qui est identifié par `div`) aux coordonnées (`x`,`y`), de dimension `w`×`h`, sur lequel est écrit `txt`, avec l'info-bulle `tip`, et enfin qui exécute la fonction `f` lors d'un clic.

Les graphes sont générés à l'aide de la fonction `Graph` qui prend comme arguments les coordonnées et les dimensions (en pixels) du graphe dans l'élément `<canvas>`. Il possède également des méthodes d'affichage pour :

1. afficher un cadre (`box`), une grille (`grid`) ou des axes (`axes`) ;
2. définir les valeurs limites des coordonnées (`xlim`, `ylim`) ;
3. afficher un signal à temps continu (`plot`) ou à temps discret (`stem` ou `dots`) ;
4. afficher une impulsion de Dirac (`dirac`) ;
5. effacer le graphe (`clear`).

La syntaxe de ces fonctions est inspirée de Matlab. Le graphe possède enfin des événements pour exécuter une action lors du déplacement du pointeur ou d'un clic [2].

Notons que pour effacer une courbe, la solution pour obtenir le meilleur rendu est d'effacer tout le graphe avec `clear`. Mais cela aura pour conséquence d'effacer également les éventuels cadre, grille et axes qu'il faudra ensuite dessiner à nouveau. Une solution moins coûteuse consiste à créer deux graphes : un graphe d'arrière-plan contenant le cadre, la grille et les axes (qui ne sera jamais effacé), et un cadre d'avant-plan qui ne contient que les courbes.

2. Ou toute action équivalente sur un écran tactile.

Des fonctions mathématiques sont également disponibles, comme `conv` pour calculer le produit de convolution (la syntaxe est identique à la fonction Matlab) ou `range` pour définir une liste de progression arithmétique (reproduisant la syntaxe `a:b:c` de Matlab).

4.2 Représentation d'un signal sinusoïdal

Cette simulation affiche tout simplement un sinus dont on peut modifier la moyenne, l'amplitude, la fréquence et la phase (figure 1). Des curseurs de défilement permettent à l'utilisateur de modifier ces quatre paramètres pour voir leur influence directement sur le signal. En particulier, cette simulation est utile pour illustrer la différence entre déphasage et décalage.

Le code de cette simulation contient deux fonctions :

— `init` initialise la position des contrôles et des graphes, puis exécute la fonction `draw` ;

— `draw` récupère les valeurs des curseurs de défilement, met à jour les textes affichant les valeurs des paramètres, calcule les valeurs du signal et enfin met à jour le graphe d'avant-plan.

Lors d'un clic sur un curseur de défilement, la fonction `draw` est également exécutée.

La sinusoïde est représentée de -5 à 5 avec un pas de $0{,}02$. Ses amplitudes sont calculées avec une boucle :

```
for(n = -5; n < 5; n=n+.02)
{
  t.push( n );
  y.push( m + a*Math.sin(2*Math.PI*f*n+p) );
}
```

et l'affichage du signal est effectué ainsi :

```
gf.clear();
gf.plot(t,y);
```

où `gf` est le graphe d'avant-plan.

4.3 Convolution de signaux en temps discret

Avec cette simulation (figure 2), l'utilisateur a la possibilité de tracer deux signaux x et h avec la souris, et d'observer le

produit de convolution $y = x * h$ qui est affiché en temps réel sur un troisième graphe. Il est également possible de choisir des signaux parmi une liste (signal nul, impulsion de Kronecker, porte, gaussienne, exponentielle décroissante, rampe). Cette simulation permet d'appréhender l'effet d'une modification d'un ou plusieurs échantillons sur une convolution.

Ici encore, une première fonction permet d'initialiser l'interface. Un événement est affecté au graphe `gx` de x permettant d'exécuter la fonction `movex` lors du déplacement du pointeur de la souris sur le graphe :

```
gx.mouseDrag(movex);
```

La fonction `movex` récupère alors les coordonnées du pointeur pour redéfinir le signal x, puis exécute la fonction `draw` pour calculer le produit de convolution et afficher les trois signaux. La même chose est fait pour le graphe de h. D'autres fonctions permettent de définir complètement les signaux x et h si l'utilisateur choisi l'un des signaux prédéfinis.

4.4 Code de Huffman

Le code de Huffman [6] est un codage source obtenu à partir des probabilités d'occurrence des symboles. La simulation (figure 3) propose à l'utilisateur de choisir les probabilités des symboles de la source (le nombre de symboles n'est pas limité), puis affiche l'arbre et les codes binaires des symboles.

Cette simulation s'organise en trois étapes.

1. En premier lieu, les probabilités rentrées dans une zone de texte sont récupérées et enregistrées dans une liste.
2. Ensuite l'arbre de Huffman et la définition du code sont calculés, de la même manière que dans [12, p. 342]. Pour cela, il a été nécessaire de trier les symboles et leurs codes en fonction des probabilités contenues dans un second tableau. Chaque code est en fait une liste dont les éléments 0 et 1 sont ajoutés au fur et à mesure des fusions des branches de l'arbre.
3. Enfin, l'interface est mise à jour en écrivant les symboles et les codes associés, puis en représentant l'arbre.

5 Conclusion

Les technologies modernes du web que sont HTML5 et JavaScript permettent de réaliser des simulations et animations pédagogiques pour enseigner le traitement du signal et les communications numériques. Ces simulations sont accessibles librement, rapidement et sans aucune manipulation pour les étudiants, à condition de disposer d'un navigateur web récent. Nous avons pour cela développé un code (spetsi.js) pour aider à la conception de ce type de simulations. Les codes sont librement accessibles et nous encourageons l'amélioration de ces simulations et le développement de nouvelles !

Remerciements

Plusieurs des simulations disponibles ont été développées dans le cadre de projets étudiants : je remercie donc vivement F. Schorr, M. Abouhaouari et S. Sandiramourty. Un grand merci également à C. Meillier pour la relecture.

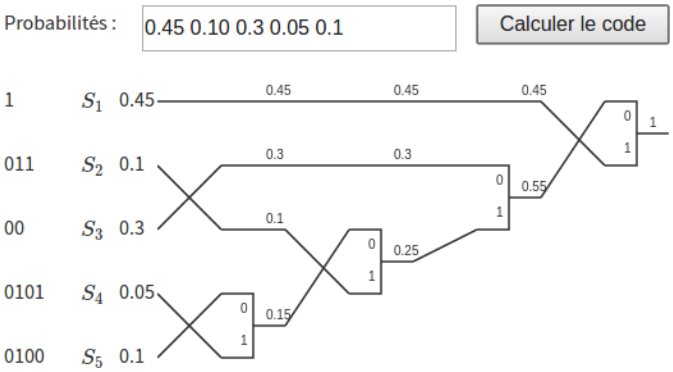

FIGURE 3 – Code de Huffman.

Références

[1] N. Aliane. Spreadsheet-based interactive modules for control education. In *Computer Applications in Engineering Education*, 2009.

[2] S.G. Crutchfield and W.J. Rugh. Interactive exercises and demonstrations on the web for basic signals, systems and control. In *Proceedings of the 36th IEEE Conference on Decision and Control*, 1997.

[3] S.G. Crutchfield and W.J. Rugh. Interactive learning for signals, systems, and control. *IEEE Control Systems*, 18(4) :88–91, 1998.

[4] File Exchange. fr.mathworks.com/matlabcentral/ fileexchange/.

[5] GeoGebra. www.geogebra.org/.

[6] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9) :1098–1101, 1952.

[7] Mathematica. www.wolfram.com/mathematica/.

[8] MathJax. www.mathjax.org/.

[9] MathWorks. fr.mathworks.com/.

[10] Oracle. *Moving to a Plugin-Free Web*. blogs.oracle.com/ javaplatform-group/entry/moving_to_a_plugin_free.

[11] Wolfram CDF Player. www.wolfram.com/cdf-player/.

[12] J.G. Proakis and M. Salehi. *Digital Communications*. McGraw-Hill, 2008.

[13] Matlab Toolbox Image Processing. fr.mathworks.com/products/ image.html.

[14] Matlab Toolbox Signal Processing. fr.mathworks.com/products/ signal.html.

[15] W.J. Rugh. *Signals, Systems, and Control Demonstrations*. pages.jh.edu/∼signals/.

[16] A. Spanias. *JAVA Digital Signal Processing*, 2008. www.eas.asu.edu/ midle/jdsp/jdsp_exercises.html.

[17] Chrome V8. developers.google.com/v8/.

[18] M.A. Wickert. Using software-based animations of signal processing mathematics to enhance learning. In *Digital Signal Processing Workshop, IEEE Signal Processing Education Workshop*, 2009.

[19] Wikipedia. *Java applet*. en.wikipedia.org/wiki/ Java_applet.