TREIZIÈME COLLOQUE GRETSI - JUAN-LES-PINS DU 16 AU 20 SEPTEMBRE 1991

MUSTIG, UN OUTIL PARTICULIEREMENT ADAPTE POUR L'ENSEIGNEMENT DU TRAITEMENT DU SIGNAL

G. Lejeune, F. Glangeaud et J. Liénard

CEPHAG - ENSIEG

BP 46, 38402 St-Martin d'Hères - FRANCE

Résumé. MUSTIG, langage graphique et interactif pour le traitement des signaux multidimensionnels, est considéré comme un environnement d'apprentissage pour l'enseignement du Traitement du Signal, et comparé à d'autres environnements. Pour que MUSTIG devienne un logiciel d'EIAO, il faudra lui ajouter un guidage intégré, avec toutes les difficultés que cela comporte. Dans sa forme actuelle cependant, il permet de construire des documents interactifs ou "Hypermedia" grâce au concept d'"Image". On donne quelques exemples de l'utilisation des documents interactifs pour l'enseignement.

Abstract. MUSTIG, graphical and interactive language for multidimensional signal processing, is considered as an environment module for teaching Signal Processing, and is compared to other environments. For MUSTIG to become an EIAO software, it must be augmented by an integrated guiding module, a very difficult issue for such an unstructured discovery learning. However, in its present stage, it allows to build interactive documents or "Hypermedia" through the "Image" concept. One gives some examples of use of such interactive documents for teaching.

1. INTRODUCTION

Historiquement, l'EAO (Enseignement Assisté par Ordinateur) s'est développé autour d'interfaces de type textuel (langue écrite, symboles), et la recherche en EIAO (Enseignement d'abord Intelligent, maintenant Interactif, Assisté par Ordinateur) s'est concentrée sur le développement des modèles de dialogue tutoriel.

Par ailleurs, avec le développement des interfaces Hommes-Machines du style Macintosh (clavier-souris et WYSIWIG), des outils logiciels "conviviaux" ont vu le jour ; l'idée essentielle derrière ces environnements logiciels s'appuie sur la notion de métaphore : celle d'un bureau pour le bureau électronique du Macintosh ou de Windows, celle d'un cahier de brouillon, d'une feuille de dessin ou d'une page de document pour les applications multiples qui ont suivi. La plupart de ces applications (traitements de texte, tableurs, ...) sont des logiciels professionnels, dont l'objectif est la production : ce qui est apprécié en eux par les utilisateurs, c'est leur caractère d'efficacité. Ces deux caractéristiques (convivialité et efficacité) font que souvent "les formateurs se tournent vers ces logiciels qui a priori ne sont pas construits à des fins pédagogiques, mais qui, en apportant des outils caractéristiques d'un domaine, sans imposer de démarche particulière au cours de leur mise en œuvre, permettent d'en moduler l'utilisation en fonction du public auquel ils s'adressent" (Baulac, 1989). Le langage MUSTIG joue un tel rôle en traitement du signal, grâce à la convivialité de son interface (programmation graphique), son interactivité (calculs incrémentaux), sa puissance (hiérarchie illimitée) et ses possibilités de mise en page des résultats (les "Images").

Enfin on assiste, dans la ligne des développements précédemment évoqués, à la naissance de véritables environnements d'apprentissage ou de micromondes, dont l'archétype est le fameux LOGO, et une réalisation récente mettant en œuvre intégralement la manipulation directe est le logiciel Cabri-géomètre pour l'apprentissage de la géométrie. Le problème essentiel dans ces environnements est le problème du guidage (Elsom-Cook 1990). C'est le deuxième axe des recherches actuelles en EIAO.

On verra que MUSTIG est un excellent environnement d'apprentissage pour le traitement du signal, et on le montrera à partir des documents interactifs (Slaney 1990) qu'il produit. On verra par contre le grand chemin à parcourir pour qu'il devienne un environnement d'apprentissage guidé au sens d'Elsom-Cook 1990, car il est établi que l'expérimentation seule ne suffit pas à modifier le système de connaissances de l'utilisateur.

2. LES ENVIRONNEMENTS D'APPRENTISSAGE

2.1 L'interface Homme-Machine

La conception de l'interface est de la plus haute importance pour les environnements d'apprentissage (O'Malley, 1990) : en effet, l'interface, intermédiaire entre le domaine de connaissances et l'utilisateur, représente des "objets", des concepts et des relations, permet de manipuler une partie des objets, est une mémoire auxiliaire de l'utilisateur (sous forme de textes d'annotation par exemple), et enfin est le lieu de génération de dialogues.

L'ambition de tout environnement d'apprentissage est de réduire la distance entre l'élève et le domaine de connaissances (Hutchins et al, 1985), ce qui implique de minimiser deux distances : d'une part la distance entre le domaine de connaissances et les représentations à l'interface (transparence du domaine), et d'autre part la distance entre l'interface et l'utilisateur (la formation à l'utilisation du système réduisant cette distance, formation qui sera d'autant plus facile que l'interface sera ressenti comme "transparent" par l'utilisateur). D'après Burton (1987), l'interface peut présenter à l'utilisateur différents niveaux d'abstraction (le plus concret convenant aux débutants, le plus abstrait aux spécialistes du domaine), et pour chaque niveau d'abstraction, l'interface doit être fidèle (fidélité visuelle, mécanique, ou conceptuelle selon les niveaux

d'abstraction). Enfin, le système peut imposer une plus ou moins grande quantité de "structure" à l'activité de l'utilisateur.

Pour situer MUSTIG dans cette problématique, il est utile de le comparer à d'autres environnements d'apprentissages.

2.2 Les différents types d'environnements

L'exemple souvent cité de Steamer (Woolf, 1987) correspond par exemple à une simulation à relativement faible niveau d'abstraction alors que FAULT (Johnson,1987) a un niveau d'abstraction élevé.

Steamer offre une représentation et un contrôle graphique de la simulation d'une centrale thermique. Le système est figé : l'élève ne peut que modifier des conditions initiales ou aux limites, mais ne peut ni ajouter ni supprimer des composants au système : le niveau de structure imposée est donc grand. L'élève peut commettre des erreurs dans le pilotage de la centrale, par exemple en imposant un débit en dehors des plages de fonctionnement normal, mais la prise en compte de ses actions par le système expert et le tutoriel sous-jacent est facilitée par le caractère très structuré des actions possibles.

Le système FAULT est basé sur une représentation fonctionnelle de l'équipement, les connections entre les différents modules, appartenant à un graphe de dépendances, ne voulant représenter aucun mécanisme physique. Afin de procéder à un diagnostic de panne, on présente à l'élève un tel graphe représentant par exemple un moteur de voiture. Un tel exemple représente en fait **un** tutoriel particulier (appelons-le "Moteur") construit à l'aide du générateur de tutoriel FAULT. "Moteur" représente la réalité à un grand niveau d'abstraction, mais la structure imposée est grande. FAULT, en tant que générateur, a une structure imposée très faible, mais on ne le met pas entre les mains de l'élève !

L'environnement d'apprentissage Cabri-géomètre (Laborde JM et C, 1991) représente des objets très concrets, les figures de la géométrie plane. L'élève peut expérimenter à sa guise et créer n'importe quelle figure issue de son imagination. On est ici en présence d'un environnement d'apprentissage particulier qu'on appelle micromonde. Le micromonde contraint l'ensemble des actions possibles de telle sorte qu'il est impossible de "faire une erreur" : le niveau de structure imposée est donc plus grand que pour les langages de programmation, mais bien moindre que dans Steamer ou les tutoriels classiques.

MUSTIG, en tant que langage de programmation, se situe plus comme un générateur de simulateur, se rapprochant donc davantage du générateur de tutoriel FAULT en ce qui concerne les caractéristiques d'abstraction et de structure (voir aussi le système IMTS, "Intelligent Maintenance Training System", de Towne, 1987). On pourrait imaginer la réalisation du simulateur de Steamer en MUSTIG : on partirait des équations discrétisées du modèle physique sous-jacent à Steamer, on les représenterait sous forme d'un graphe de dépendances et on construirait à partir de ce graphe un ou plusieurs "documents interactifs" (cf Partie 3), permettant la visualisation et le contrôle des paramètres de la simulation. De même, et encore plus simplement, le simulateur "Moteur" défini précédemment à partir de FAULT pourrait être construit avec MUSTIG (on n'inclue bien évidemment pas la partie tutoriel) !

Par contre, les objets MUSTIG sont d'une autre nature : on peut dire que, quand on construit interactivement le graphe de dépendance d'un programme MUSTIG, on "visualise des concepts" sur l'écran ; l'interface utilise alors la notion de métaphore : l'utilisateur est en face d'un "cahier de brouillon" sur lequel il dispose des opérateurs sous forme de "boîtes" reliées entre elles. On peut rajouter d'autres boîtes sur le graphe, en détruire certaines, en changer le contenu, les ouvrir et les refermer, ... Grâce au concept d'"image", boîte d'un type particulier sur l'écran, résurgence d'une partie du graphe, on pourra construire les "documents interactifs".

Le système de programmation ayant été conçu au départ pour résoudre des problèmes de traitement du signal, c'est sur ces problèmes que MUSTIG sera le plus efficace : la distance entre le domaine de connaissances et les objets manipulés à l'écran sera minimisée, ainsi que la distance entre l'utilisateur (le "traiteur de signal") et l'interface, puisque la représentation MUSTIG correspond souvent aux représentations du domaine dans l'esprit de l'utilisateur (cf la représentation du filtre en partie 4). La plus grande difficulté de cette approche de MUSTIG de type "micromonde" pour la réalisation d'un environnement d'apprentissage est son caractère très peu structuré. Le compilateur de MUSTIG détecte bien sûr les erreurs de syntaxe et les incohérences de programmation : mais dans l'état actuel des choses, il ne peut en aucun cas reconnaître les intentions de l'utilisateur !

2.3 La nécessité du guidage

"De nombreuses expériences ont montré la nécessité du guidage, essentiellement parce que pour explorer intelligemment il faut avoir des buts précis. Au bout d'un certain temps, l'élève demande souvent : "Qu'est-ce qu'il faut faire maintenant... "" (Baker, 1991).

Soit alors le guidage a été intégré dans l'environnement d'apprentissage lui-même, et là réside la plus grande difficulté de réalisation, soit on situe le guidage dans la situation didactique en sollicitant une intervention du professeur sous la forme d'une liste d'objectifs, d'une supervision ... C'est cette deuxième solution qui est retenue par Glangeaud (1991) pour introduire ses étudiants à la pratique de l'analyse spectrale à partir de documents interactifs MUSTIG.

2.4 Problèmes liés au guidage intégré dans les environnements d'apprentissage peu structurés

Avant de pouvoir guider, le système va devoir reconnaître ce que veut faire l'élève : avoir quelque notion du sens que l'utilisateur donne à son interaction avec le système ; cela n'est pas seulement vrai pour les environnements d'apprentissage, mais prend une importance particulière quand il s'agit de proposer des environnements d'apprentissage adaptatifs, avec guidage intégré dans le système.

Il est difficile d'imaginer une méthode de diagnostique passif, basé sur la reconnaissance des intentions de l'élève à partir de sa seule interaction avec le système. Une approche plus réaliste est celle dans laquelle des buts sont fixés à l'élève, et ces buts doivent être réalisés selon un certain planning. Le guidage procédera alors à travers des dialogues tutoriels, la génération de ces dialogues posant aussi certains problèmes (Baker,1991) :

- quand faut-il générer une intervention de quidage ? (sans véritablement interrompre la tâche de l'apprenant, ni le démotiver)
- de quoi faut-il parler ? (reprise d'objets des dialogues précédents et selection d'aspects de la tâche actuelle de l'apprenant)
- comment faut-il donner le guidage ? (questionnement, explication, démonstration, ...)
- où faut-il générer l'énoncé ? (dans une autre fenêtre, comme message qui bloque la tâche de l'apprenant, en forme d'exemple, ...)

3. LE DOCUMENT INTERACTIF

3.1 Le langage de programmation graphique MUSTIG

Le langage MUSTIG (Lienard, 1987, Lejeune et Lienard, 1989) permet la construction graphique d'un graphe de dépendances, interconnectant entre eux des opérateurs puisés dans une bibliothèque. Ce graphe peut être "calculé" à tout instant, un compilateur intégré déterminant alors de quoi dépendent les sorties dont on demande le calcul, et parcourant le graphe de la fin au début pour remonter aux entrées. Si la syntaxe est correcte, les calculs nécessaires sont effectués et en général des résultats sont affichés et des courbes tracées. S'il y a une erreur de syntaxe, l'endroit erronné sur le graphe est sélectionné, et le message d'erreur correspondant apparait sur l'écran.

Pour chaque opérateur, l'utilisateur peut demander une aide : une nouvelle fenêtre s'ouvre alors, comportant la description de l'opérateur sélectionné.

L'utilisateur peut fabriquer ses propres opérateurs (sous forme de "macros") : des "objets" ou fonctions complètement nouveaux peuvent ainsi être construits à partir des blocs de base, et documentés au moment de leur construction.

3.2 La construction de documents interactifs

D'après Slaney (1990), un document interactif inclue du texte et un modèle pour calculateur ("computer model") de sorte qu'il est plus facile pour les lecteurs de comprendre le matériel, car ils peuvent lui poser des questions. Les notions d'hypermedia et de "Literate programming" (qu'on pourrait peut-être traduire par "programmation transparente") sont aussi évoquées à leur propos. Les exemples donnés par Slaney ont été construits à l'aide du logiciel Mathematica.

Le document interactif est en effet une des approches possibles à l'introduction dans un simulateur (tel que Mathematica et MUSTIG) d'une composante tutorielle **dans le cas où on a fixé un but à l'utilisateur**. Slaney regrette cependant le manque de transparence d'un tel document dans la mesure où les notations ne sont pas universelles : MUSTIG déplace le problème en supprimant la notation textuelle et en la remplaçant par une programmation graphique, qui doit être connue de l'utilisateur. Mais MUSTIG permet aussi, grâce à son concept d'"image", de présenter des équations dans le langage de tous les jours, dans la mesure où les coefficients d'équations figurant dans le graphe MUSTIG peuvent être "imagés" au milieu d'un texte explicatif quelconque. Ces images sont des résurgences au milieu du texte de certaines parties du graphe MUSTIG, et sont **actives** : une modification de l'image est reportée automatiquement dans le graphe. La construction de telles "images" est très facile grâce aux possibilités de mise en page du logiciel. On construit de la même manière des "Faces avant" d'instruments virtuels destinés à piloter le traitement.

3.3 Exemples de documents interactifs et leur utilisation en classe

Un des buts qui peut être proposé à l'utilisateur (cf Slaney) est de comprendre une petite partie du domaine de connaissances, représentée sous la forme d'un document interactif. La figure 1 montre le document interactif expliquant la méthode du "Périodogramme moyenné" en analyse spectrale. Tous les paramètres entourés représentent des valeurs que l'élève peut modifier : si des résultats (valeurs, courbes, ...) du document interactif dépendaient de ce paramètre modifié, ils sont effacés et l'élève peut demander qu'ils soient recalculés. Le texte explicatif doit être suffisant pour que la signification des contrôles soit bien comprise : en cas d'incertitude, l'élève peut toutefois aller consulter le graphe sous-jacent : par exemple, il peut "ouvrir" une des trois boîtes "Géné", "Apod./t" ou "Analyseur par blocs/t". Il a alors accès au graphe (dont le premier niveau de hiérarchie est représenté sur la figure 2).

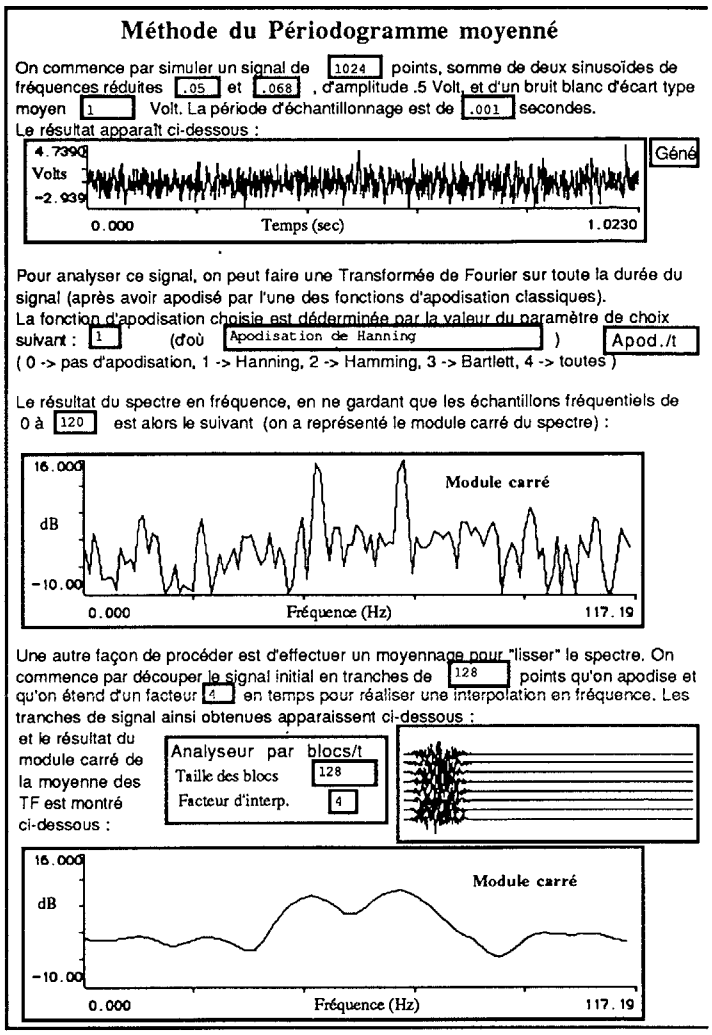

Figure 1

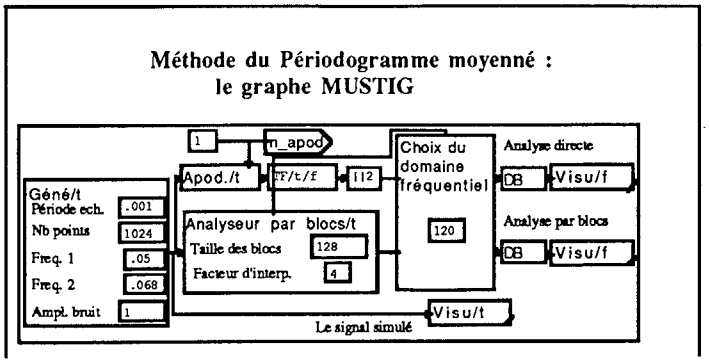

Figure 2

Un autre exemple d'utilisation possible des documents interactifs est la résolution d'un exercice de traitement du signal : prenons par exemple le problème 5.1 page 255 du livre de classe de Ludeman (1986). Le langage MUSTIG est très bien adapté au **dessin** des filtres digitaux (qui constitue en fait le programme graphique MUSTIG). L'élève vérifiera que les 4 réalisations de filtres demandées conduisent à la même réponse impulsionnelle, dans la limite des arrondis de calcul (figure 3) :

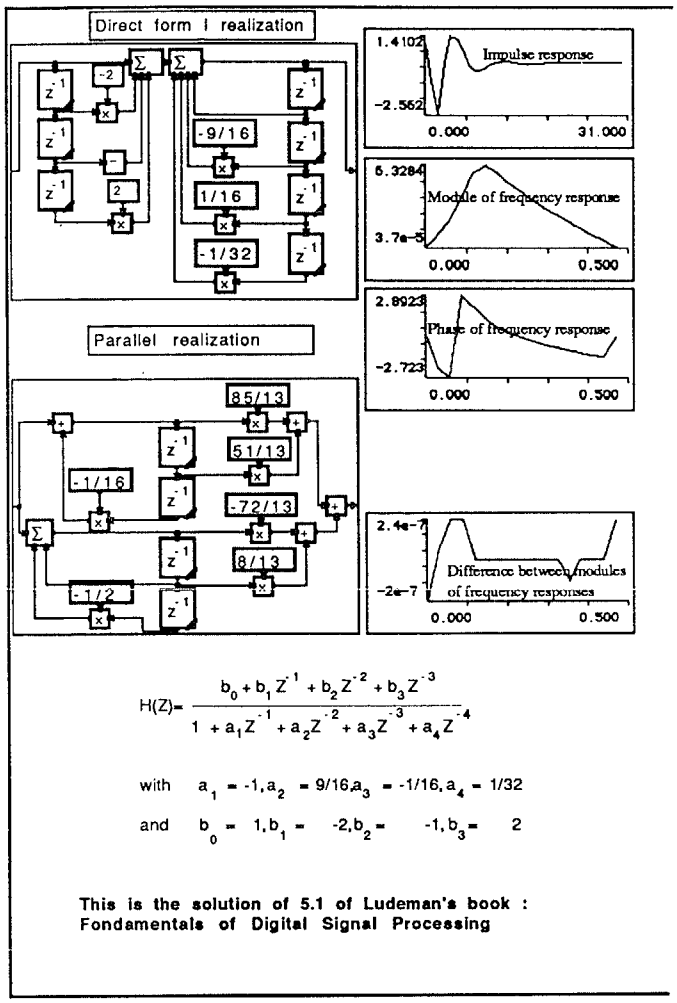

Figure 3

A la suite de 4 années d'enseignement (Cours, Bureaux d'études, Formation permanente, Examens) du traitement du signal sur MUSTIG, les différents types d'utilisation des documents interactifs de MUSTIG pour l'enseignement sont maintenant passés en revue :

- la rédaction d'un "polycopié" (Glangeaud, 1991), distribué sur support papier et formé d'une suite d'Images, reliées ou non.
- la distribution, en plus du "polycopié" papier, des fichiers MUSTIG correspondant, ce qui permet à l'élève d'expérimenter sur les degrés de liberté, et même de modifier éventuellement le schéma.
- l'utilisation en Travaux Dirigés ou en cours de tels types de fichiers, avec explications complémentaires de la part du professeur.
- la réalisation d'un examen, écrit ou oral, qui consistera pour l'élève à réaliser un but fixé ("Soit un premier fichier M1 : construire M2 de manière à ce que le signal résultat ..." ou bien "Voici un fichier de données : analyser ces données et donner votre réponse sous forme d'une Image de MUSTIG").

4. PERSPECTIVES ET CONCLUSION

On a vu que l'utilisation des documents interactifs MUSTIG facilite déjà grandement la tâche de l'enseignant en Traitement du Signal. Il reste que, pour réaliser un enseignement individualisé et idéalement adapté au moment où l'élève en a besoin, il faut un guidage automatique incorporé directement au cœur de la tâche de l'apprenant, et nous avons vu dans la partie 2 les difficultés théoriques et pratiques que cela pose.

Le projet ASI (Analyse Spectrale Intelligente) du CEPHAG consiste à faire interagir le "simulateur" MUSTIG (Adnet, 1991) avec un système expert construit à partir d'un ensemble de règles en analyse spectrale. La réalisation de ce projet sera déjà un premier pas vers l' environnement d'apprentissage guidé, en ce sens qu'il résoudra le problème de la spécification d'un graphe MUSTIG à partir d'une tâche à résoudre.

Mais, si on suit Woolf (1987), il faut "clarifier et implémenter les composantes d'une philosophie de l'apprentissage tôt dans le processus de développement du tutoriel intelligent. .... nous ne pouvions construire le système expert, et puis plus tard le tutoriel."

Il faut donc étudier dès maintenant le problème inverse du précédent, à savoir la reconnaissance par le système des intentions de l'élève à partir de ses actions sur l'interface, et donc de disposer d'un certain modèle de l'apprenant. Cela devra se faire sans trop réduire la généralité du langage MUSTIG, c'est-à-dire sans imposer une structure trop contraignante à l'élève.

Enfin, pour l'évaluation et la validation du produit en cours de réalisation, nous ne pourrons pas nous satisfaire d'appréciations globales du type : "Les élèvent aiment ...". Pour arriver à des résultats quantitatifs, il faut que les mécanismes de validation aient été pensés au départ et que des outils correspondants soient intégrés au produit.

Il n'existe pas encore de véritable EIAO expérimenté dans le réel, à l'école et à l'université, mais il est sûr que les développements théoriques en EIAO influencent les expériences concrètes. Cela justifie chaque petit pas dans la direction que nous nous sommes fixée.

5. REFERENCES

Adnet C., Présentation du logiciel SAS, Rapport CEPHAG,1991.

Baker M., Communication privée, 1991.

Baulac Y.,Cabri-géomètre, présentation fonctionnelle d'un outil logiciel d'aide à l'enseignement de la géométrie élémentaire, Document interne LSD2 IMAG (1989).

Burton R.R,The Environment Module of Intelligent Tutoring Systems, Intelligent Systems Lab., Xerox Palo Alto Research Center (1987).

Elsom-Cook M., Guided discovery tutoring, London : Paul Chapman (1990).

Glangeaud F., La pratique de l'analyse spectrale assistée par MUSTIG, Rapport CEPHAG n° 2/91 (1991).

Hutchins E.L., J.D. Hollan et D.A. Norman, Direct manipulation interfaces Human-Computer Interaction,1,311-338. Hillsdale, NJ : Lawrence Erlbaum Associates (1985).

Johnson W.B., Development and evaluation of simulation-oriented computer-based instruction for diagnostic training. In W.B. Rouse (Ed.), Advances in man-machine systems research (Vol 3). Greenwich,CT : JAI Press (1987).

Laborde C et Laborde J.M., Micromondes intelligents et environnement d'apprentissage, In BELLISSANT C. (Eds) Actes des XIII° journées francophones de l'informatique, Grenoble : IMAG-CNRS (à paraître 1991).

Lienard J., Langage conversationnel pour le traitement des multi-signaux, GRETSI, Juin 1987.

Lienard J. et G. Lejeune, Exemples d'utilisation du logiciel de traitement interactif MUSTIG, GRETSI, Juin 1989.

Ludeman L.C., Fundamentals of Digital Signal Processing, Harper & Row, 1986.

O'Malley Cl., Interface Issues for Guided Discovery Learning Environmants (1990).

Slaney M., Interactive Signal Processing Documents, IEEE ASSP Magazine, Avril 1990.

Woolf B.P., Representing complex knowledge in an intelligent machine tutor, Computational Intelligence, 3(1),1987.