Electrónica e Telecomunicações

universidade de aveiro

theoria poiesis praxis

AVEIRO • JAN • 95 • VOL.1 • Nº3

Revista do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro

Electrónica e Telecomunicações

Revista do Departamento de
Electrónica e Telecomunicações
da Universidade de Aveiro

Editores:

Francisco Vaz
José Luis Oliveira

Comissão Editorial:

Alexandre Mota
Ana Maria Tomé
Aníbal de Oliveira Duarte
António Ferrari de Almeida
António Nunes da Cruz
António Sousa Pereira
Atílio Gameiro
Dinis Magalhães Santos
Fernando Ramos
Joaquim Arnaldo Martins
João Pedro Estima de Oliveira
José Alberto Fonseca
José Alberto Rafael
José Carlos Neves
José Carlos Pedro
José Ferreira da Rocha
José Rocha Pereira
Nelson Pacheco da Rocha
Maria Beatriz de Sousa Santos
Paulo Jorge Ferreira

Morada e Secretariado:

Departamento de Electrónica e
Telecomunicações
Universidade de Aveiro
Campo Universitário
3800 Aveiro
Portugal

Artes Gráficas:
Sérgio Cabaço

Impressão:
Gráfica Aveirense

Tiragem: 600 exemplares

Editorial

Este terceiro número da revista Electrónica e Telecomunicações surge cerca de um ano após o número inicial. Foi assim atingido a periodicidade semestral que de início nos tinhamos proposto, e constitui, sem dúvida, um motivo de alegria para os responsáveis pela sua edição. No entanto é ainda grande o conjunto de objectivos que pretendemos alcançar. Entre eles gostariamos hoje de salientar o de aumentar a difusão da revista, sobretudo junto dos antigos alunos do Departamento de Electrónica e Telecomunicações. Para tal contamos com a colaboração da Associação dos Antigos Alunos da Universidade de Aveiro para a realização de uma campanha de divulgação e angariação de assinaturas. Através da revista gostariamos de criar mais um elo de ligação às centenas de profissionais que por aqui passaram e contribuir, na medida do possível, para a sua actualização científica e técnica.

Queremos deixar também expresso o nosso agradecimento à JNICT - Junta Nacional de Investigação Científica e Tecnológica pelo apoio financeiro prestado na edição dos números anteriores.

CENTRO DE ESTUDOS DE TELECOMUNICAÇÕES

45 anos de Investigação e Desenvolvimento
20 anos de colaboração com a Universidade de Aveiro

Sonhar em Aveiro as Telecomunicações do Futuro

Design: CET/MK

CENTRO DE ESTUDOS DE TELECOMUNICAÇÕES

PORTUGAL TELECOM
DIRECÇÃO CENTRAL DE INVESTIGAÇÃO E DESENVOLVIMENTO

Rua Engº José Ferreira Pinto Basto - 3810 AVEIRO - Telefone: 034-381831 - Fax: 034-24723 - E-mail: cet@cet.pt

Índice

Interface de Utilizador para Sistemas de Gestão de Imagem Médica

Gonçalo Paiva Dias, José Alberto Rafael, Beatriz Sousa Santos

***Resumo*- Neste artigo é feito um estudo da aplicação à área dos sistemas de gestão de imagem médica dos princípios gerais dos interfaces de utilizador. É apresentado o projecto de um interface para estes sistemas, e uma simulação usando *Microsoft® Visual Basic™*.**

***Abstract*- This paper is about using user interface principles in the development of Picture Archiving and Communication Sistems. A PACS user interface project is described, and a prototype developed using Microsoft® Visual Basic™ is presented.**

I. PRINCÍPIOS GERAIS DOS INTERFACES DE UTILIZADOR

Mayhew, Foley e outros [1], [2] apontam alguns princípios gerais a seguir no desenvolvimento de interfaces de utilizador, em que se baseia uma grande parte das escolhas feitas neste trabalho. Estes princípios são bastante gerais, pelo que devem ser interpretados no contexto da sua aplicação a casos concretos.

A. Conhecer o utilizador

As características dos utilizadores são preponderantes no projecto do interface de utilizador de qualquer sistema informático. O projecto do interface deve ser feito tendo em conta o funcionamento do sistema cognitivo humano e os conhecimentos e comportamentos específicos dos utilizadores a que se destina.

Existem algumas características dos utilizadores que influenciam determinantemente as escolhas que é necessário fazer quando se desenha qualquer interface. As principais são:

- atitude;
- motivação;
- facilidade de uso do teclado;
- experiência no uso da aplicação;
- experiência no desempenho das funções;
- uso frequente de outros sistemas;
- conhecimentos gerais de computadores.

Um bom interface deve prever a possibilidade de poder haver disparidade quanto a estas características entre os vários utilizadores. É particularmente importante que o interface se adapte a utilizadores com maior ou menor facilidade no uso do teclado e maior ou menor experiência no uso da aplicação e no desempenho das funções.

B. Conhecer a tarefa

A funcionalidade oferecida por um interface de utilizador deve ser baseada numa ideia clara e completa da organização a que se destina e do trabalho e tarefas que os utilizadores desempenham. Algumas características dessas tarefas são importantes na definição do interface:

- frequência com que a tarefa é executada;
- existência de treino prévio dos utilizadores;
- obrigatoriedade do uso da aplicação;
- outras ferramentas utilizadas no desempenho da tarefa;
- importância da tarefa;
- estruturação da tarefa.

Os utilizadores estão normalmente interessados em aceder às funcionalidades disponíveis em função da tarefa que pretendem desempenhar, e não em procurar as funcionalidades que lhes permitem executar uma determinada tarefa. A possibilidade de transitar rapidamente entre diferentes tarefas constitui também um importante factor para o sucesso de um interface.

C. Familiaridade e simplicidade

A necessidade de memorização por parte do utilizador deve ser, sempre que possível, substituída pela capacidade do utilizador reconhecer objectos ou formas familiares. Este princípio é tanto mais importante quanto menor for o treino ou experiência do utilizador no uso do sistema.

Um interface não deve disponibilizar simultaneamente demasiada funcionalidade. Isto resulta normalmente num interface demasiado complexo, em que uma grande parte da funcionalidade raramente é utilizada. Com efeito, com um interface demasiado complexo, os utilizadores nunca se lembram de como usar uma grande parte das operações disponíveis. Quanto mais simples for o interface mais fácil é também aprender a utilizá-lo.

D. Feedback

O interface deve responder sempre de imediato a uma acção do utilizador. Se não o fizer, o utilizador terá tendência para pensar que a sua acção não foi recebida ou percebida pelo computador. Quando o processamento que resulte da acção for demorado, o interface deverá informar o utilizador desse facto. Para tempos de resposta

muito elevados é inclusivamente aconselhável que o interface dê indicação do estado de progressão do processamento.

E. Consistência

Num sistema existem normalmente operações similares que podem ser executadas em diferentes alturas ou contextos. Um sistema é consistente se a essas operações similares corresponderem procedimentos similares ou mesmo idênticos. A importância deste princípio advém do facto de as pessoas raciocinarem normalmente por analogia e, desse modo, poderem prever a forma de resolver um problema, baseando-se na sua experiência anterior.

F. Minimização de erros

Num determinado contexto de utilização, o interface só deve permitir que o utilizador escolha ou aceda às operações que fazem sentido nesse contexto. A não observação deste princípio pode resultar na escolha de uma operação ilegal por parte do utilizador e numa escusada mensagem de erro. Sempre que o utilizador escolha operações que possam ter consequências drásticas, nomeadamente aquelas que possam levar à perda de informação, o interface deverá pedir uma confirmação indicando essas consequências.

G. Robustez e recuperação de erros

Os erros humanos na utilização de um interface são comuns e inevitáveis. Qualquer interface de utilizador deve estar preparado para lidar com essas situações. Sempre que possível o interface deverá ser capaz de repor a situação anterior a uma acção errónea por parte do utilizador. As situações de bloqueamento ou *crash* do sistema deverão ser minimizadas o mais possível. O utilizador deve ter confiança em que o sistema é suficientemente robusto para lidar com qualquer tipo de entrada, incluindo erros.

II. SISTEMAS DE GESTÃO DE IMAGEM MÉDICA

A. Definição

Não é fácil encontrar uma definição globalmente aceite para um Sistema de Gestão de Imagem Médica (SGIM) e para as funcionalidades que deve oferecer [3]. No âmbito deste trabalho, consideramos que um SGIM é um sistema digital que permite, de forma integrada, adquirir, armazenar e visualizar imagens e relatórios de exames imagiológicos, e consultar informação demográfica de pacientes e informação relativa a marcações de exames. Esta definição é bastante abrangente, englobando funcionalidades que são atribuídas por diversos autores aos sistemas de PACS (*Picture Archiving and Comunication System*), IMACS (*Image Management and Comunication System*), DINS (*Digital Image Network System*) e IDS (*Integrated Diagnostic System*) [3]-[9].

B. Interacção com outros sistemas de informação hospitalar

Frequentemente existem nos hospitais outros sistemas de informação hospitalar que mantêm informação utilizada pelo SGIM. A informação demográfica de pacientes e a informação relativa a marcações de exames são exemplos de informação residente normalmente em sistemas HIS (*Hospital Information Systems*) e/ou RIS (*Radiology Information Systems*). Sempre que seja possível, o SGIM deverá utilizar a informação mantida por esses sistemas. Evita-se assim a duplicação da informação e, consequentemente, o surgimento de inconsistências. Quando a ligação ao HIS e/ou RIS não for possível, ou quando estes não existirem no hospital, deverá ser o próprio SGIM a manter a informação demográfica de pacientes e a informação relativa a marcações de exames [3], [10], [11].

C. Identificação das tarefas

Estamos interessados em identificar quais as tarefas que um utilizador desempenha ao utilizar um SGIM. O conhecimento dessas tarefas é muito importante para que possamos proceder ao projecto do interface de utilizador. Tendo em conta a definição de SGIM utilizada, as tarefas identificáveis são:

- consulta de marcações de exames;
- realização de um exame imagiológico (aquisição de imagens);
- edição de relatório clínico relativo a um exame;
- consulta de exame imagiológico e respectivo relatório;
- consulta de dados demográficos de paciente.

Vimos que podemos ter necessidade de atribuir ao SGIM funções normalmente desempenhadas por outros sistemas de informação hospitalar. Neste caso, são identificáveis duas novas tarefas que um utilizador pode desempenhar:

- marcação de exames;
- recolha e alteração de dados demográficos de pacientes.

Embora estas tarefas não sejam sempre acessíveis a partir de um SGIM, elas são sempre consideradas neste estudo, por ser essa a situação mais geral.

A literatura utilizada não faz uma análise sistemática das características das tarefas identificadas. No entanto, a experiência de contacto com hospitais permite-nos afirmar com alguma segurança que essas tarefas são importantes, são realizadas frequentemente e normalmente são muito estruturadas. É possível ainda afirmar, pelo menos no que diz respeito aos médicos, que não é utilizado muito tempo no treino dos utilizadores [3], [7], [16]. No que diz respeito ao uso de outras ferramentas para o desempenho das tarefas, constata-se

que, excepto para a aquisição e visualização de imagens, são utilizados normalmente formulários preenchidos manualmente.

No que diz respeito à aquisição e visualização de imagens, existe um conjunto de funcionalidades que são apontadas como bastante importantes num sistema deste género [4], [12]-[15]. As mais referidas são:

- visualização simultânea de imagens relativas a exames de diferentes modalidades;
- inserção de anotações sobre as imagens;
- ferramentas de modificação de contraste e melhoria de imagem;
- rotação e zoom de imagem;
- definição de zonas de interesse;
- medição de comprimentos e áreas na imagem.

D. Caracterização dos utilizadores

É possível identificar quatro classes distintas de utilizadores de um SGIM [13], [14]:

- pessoal administrativo;
- pessoal técnico;
- imagiologistas;
- médicos requisitantes de exames.

O pessoal administrativo desempenha as tarefas de marcação de exames e de recolha e alteração de dados demográficos de pacientes. O pessoal técnico pode ser responsável pela realização de exames de algumas modalidades. Os imagiologistas desempenham as tarefas de realização de exames, edição de relatórios, e de consulta de exames e relatórios. Finalmente os médicos requisitantes poderão proceder apenas à consulta de exames e relatórios. As tarefas de consulta de marcações de exames e consulta de dados demográficos de paciente poderão ser desempenhadas por qualquer classe de utilizadores, uma vez que são complementares em relação às restantes tarefas.

A literatura utilizada não caracteriza convenientemente os utilizadores de um SGIM. Constata-se que, pelo menos no que diz respeito aos médicos, os utilizadores não têm normalmente um grande apreço pelo teclado [3], [7], [17]. O contacto com hospitais permite-nos ainda afirmar que o pessoal hospitalar tem, na generalidade, uma grande experiência no desempenho das funções.

III. MODELO DO INTERFACE

A. Modelo conceptual

Um modelo conceptual é uma organização funcional genérica de um sistema, que pretende facilitar o desenvolvimento de um modelo mental por parte dos utilizadores. Um modelo mental é uma representação interna do entendimento e conceitualização que um utilizador tem de um sistema [1]. Alguns dos princípios gerais dos interfaces de utilizador apresentados têm precisamente como objectivo facilitar o desenvolvimento de um modelo mental por parte dos utilizadores: familiaridade e simplicidade, feedback, consistência.

No interface de utilizador projectado é utilizada uma analogia com um sistema familiar aos utilizadores para facilitar o desenvolvimento de um modelo mental. O sistema representa um arquivo de exames. Cada exame possui imagens, relatório, dados relativos ao exame, informação demográfica do paciente e informação sobre a marcação do exame. A procura de exames no arquivo pode ser feita recorrendo aos catálogos de pacientes, marcações e exames. Uma vez encontrado o exame pretendido, o utilizador pode guardar uma cópia do mesmo numa pasta do seu arquivo pessoal de exames.

B. Organização em janelas

Foi escolhido o paradigma de janelas como organização geral do interface. Esta organização permite sugerir de forma intuitiva o modelo mental escolhido, permitindo simultaneamente uma rápida transição entre tarefas. Às várias entidades que compõem o modelo mental (catálogos, exames, arquivo virtual, etc.) correspondem janelas diferentes. É utilizada uma janela denominada gestor de sessão que constitui o ponto de entrada no sistema. Esta janela permite identificar a classe do utilizador (administrativo, técnico, imagiologista, médico requisitante.), configurando o sistema para disponibilizar apenas as funcionalidades que permitem a execução das tarefas que correspondem a essa classe de utilizador. As janelas utilizadas são:

- gestor de sessão
- catálogo de marcações;
- catálogo de pacientes;
- catálogo de exames;
- realização de exames;
- visualização de exames;
- edição de relatórios
- arquivo virtual de utilizador.

C. Estilos de diálogo

As características das tarefas e dos utilizadores são preponderantes na escolha do tipo de diálogo a usar nas janelas do interface de utilizador. No interface projectado é usada uma combinação dos tipos de diálogo que mais se adaptam às características estudadas. Esta conjugação de vários tipos de diálogo é bastante comum e permite, se bem utilizada, construir interfaces melhor adaptados às características dos utilizadores [1].

O hábito de utilização de formulários, a grande experiência no desempenho das tarefas, o pouco treino dos utilizadores na utilização do sistema, a importância das tarefas, a boa estruturação das tarefas e o uso frequente do sistema levou-nos a usar o tipo de diálogo *fill-in forms* como base para o projecto do interface. A possibilidade de existência de utilizadores com pouca prática no uso do teclado fez com que utilizássemos menus para preencher os campos de um *form* que

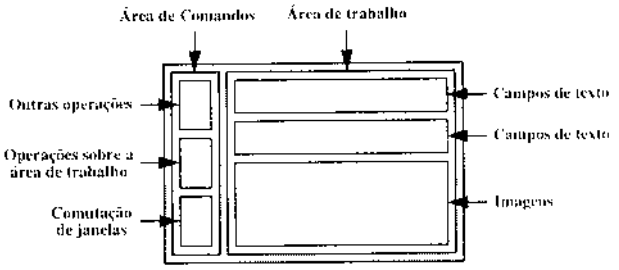


Fig. 1 - Modelo de janela

permitem escolhas alternativas. A experiência dos utilizadores no desempenho das tarefas, a importância das tarefas e a frequência com que são realizadas levou-nos a, sempre que possível, usar menus na forma de botões ou ícones. A manipulação directa foi o estilo de diálogo utilizado para executar operações sobre a imagem e para gerir o arquivo do utilizador.

D. Modelo de Janela

Foi desenvolvido um modelo genérico de janela a que obedecem as janelas do interface.

Cada janela (Fig. 1) divide-se nas áreas de trabalho e de comandos. A área de trabalho é constituída por imagens e/ou por campos de texto, agrupados em zonas que associam a informação de forma lógica. A área de comandos destina-se à colocação dos botões ou ferramentas que permitem ao utilizador escolher opções. Esta área divide-se em três zonas lógicas:

- a zona de comutação de janelas agrupa os botões que permitem iconificar a janela, abandonar a janela ou abrir outra janela;
- a zona de operações sobre a área de trabalho agrupa os botões ou ferramentas que permitem ao utilizador actuar sobre os dados presentes na área de trabalho;
- a zona de outras operações agrupa as ferramentas ou botões que permitem escolher opções específicas relacionadas com a funcionalidade oferecida pela janela.

IV. DESCRIÇÃO DO INTERFACE

O interface proposto foi simulado usando o *Microsoft® Visual Basic™*. Vamos descrever o interface analisando a forma como, nessa simulação, se processa o desempenho das tarefas identificadas para um SGIM.

A. Marcação de exames e consulta de marcações.

A marcação de exames e a consulta de marcações são feitas recorrendo ao catálogo de marcações (Fig. 2). Esta janela pode ser invocada directamente a partir do gestor de sessão, ou então a partir do catálogo de pacientes ou do catálogo de exames.

O utilizador pode procurar um paciente usando o catálogo de pacientes e, a partir deste, invocar o catálogo

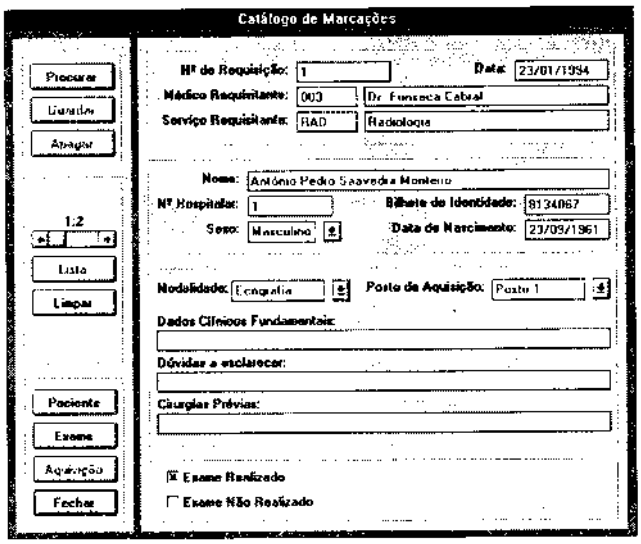


Fig. 2 - Catálogo de marcações

de marcações, que será visualizado com uma lista das marcações existentes para esse paciente. O utilizador pode também procurar um exame, usando o catálogo de exames, e invocar o catálogo de marcações carregado com a marcação que lhe deu origem. Se for invocado directamente a partir do gestor de sessão, o catálogo de marcações é lançado sem nenhuma marcação carregada, permitindo que o utilizador procure marcações pelas suas características, ou crie novas marcações.

A área de trabalho das janelas de catálogo é constituída por campos de texto editáveis, que correspondem aos campos da base de dados que suporta o catálogo. Os campos de texto são agrupados em zonas em função da informação que apresentam ao utilizador. O utilizador usa os campos de texto para introduzir informação que serve de base à pesquisa de dados e à criação de novos registos ou, alternativamente, para ver um conjunto de registos carregados na janela. A zona de outras operações da área de comandos contém um botão que permite pesquisar a base de dados em função da informação presente nos campos de texto. Os registos resultantes da pesquisa são carregados na janela e podem ser vistos na área de trabalho. A zona de outras operações inclui ainda botões que permitem apagar e alterar registos da base de dados. A zona de operações sobre a área de trabalho contém ferramentas, que permitem navegar nos registos carregados ou descarregar os registos. A navegação nos registos carregados na janela pode ser feita usando uma *scroll bar* ou acedendo a uma outra janela que contém a lista dos registos.

B. Consulta, recolha e alteração de dados demográficos de pacientes

As tarefas de consulta, recolha e alteração de dados demográficos de pacientes são executadas pelo catálogo de pacientes (Fig. 3). Esta janela pode ser invocada a partir do catálogo de marcações, do catálogo de exames ou do gestor de sessão, sendo o funcionamento nos vários

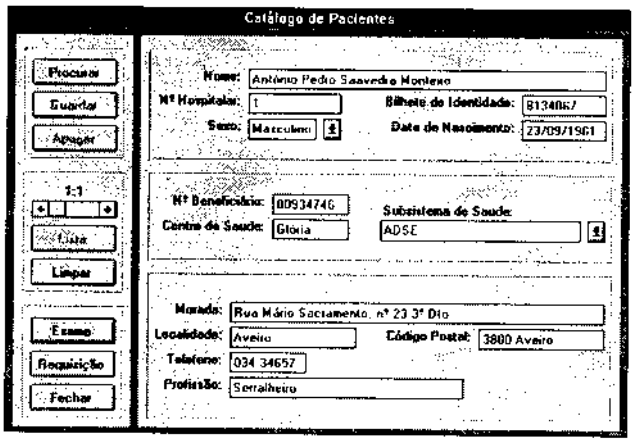


Fig. 3 - Catálogo de pacientes

casos idêntico ao da marcação de exames e consulta de marcações.

C. Consulta de exame imagiológico e consulta e edição de relatório

O acesso a um exame residente no arquivo de exames é sempre precedido de uma selecção feita usando o catálogo de exames (Fig. 4). A selecção de um exame feita nesta janela pode ser precedida de uma procura feita usando o catálogo de pacientes ou o catálogo de marcações. A visualização do exame é feita usando a janela de visualização de exames (Fig. 5). Nesta janela a área de trabalho contém uma zona de imagens e uma zona de campos de texto. A zona de imagens permite a visualização de imagens, enquanto na zona de campos de texto é mostrada a informação fundamental relacionada com o exame. A zona de operações sobre a área de trabalho contém ferramentas que permitem navegar nas imagens e fazer operações sobre as imagens. A zona de outras operações contém botões que permitem mostrar outras janelas contendo toda a informação sobre o paciente e sobre a marcação do exame. Esta zona contém ainda botões para impressão e para acesso à janela de edição de relatórios, que permite editar ou visualizar o relatório correspondente ao exame.

D. Realização de exame imagiológico

O acesso à janela de realização de exames (aquisição de imagens) é feito a partir do catálogo de marcações. Só é possível aceder à realização de exames se, no catálogo de marcações, for seleccionada uma marcação cujo respectivo exame ainda não tenha sido realizado. Uma vez realizado o exame, é possível aceder à janela de visualização de exames e daí à de edição de relatórios. A edição do relatório clínico poderá assim ser feita de imediato, ou deixada para mais tarde.

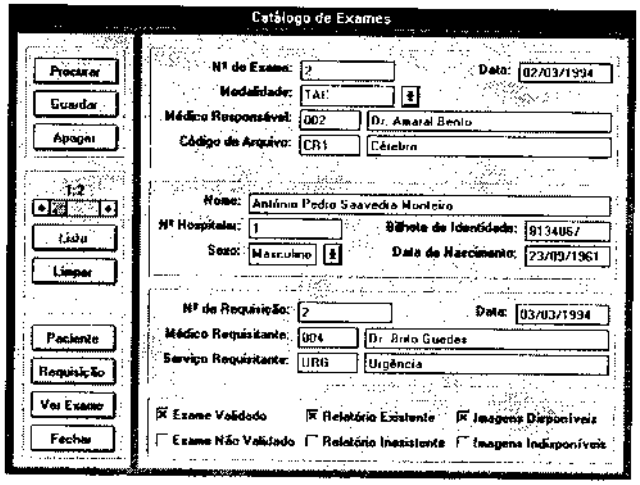


Fig. 4 - Catálogo de Exames

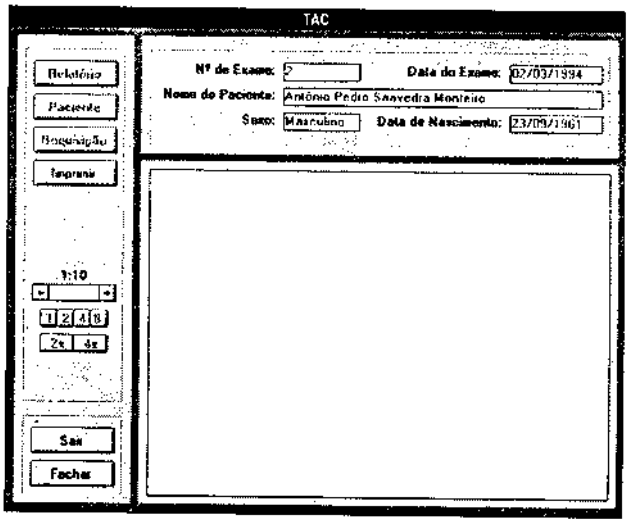


Fig. 5 - Visualização de exames

E. Arquivo virtual do utilizador

Quando um utilizador está a visualizar um exame, pode indicar que quer arquivar esse exame no arquivo virtual do utilizador. Este arquivo é organizado pelo próprio utilizador e usa o conceito de pasta. Trata-se de um arquivo virtual, porque os exames não são realmente copiados para esse arquivo. O sistema deverá manter entre sessões informação sobre os exames que o utilizador quer no arquivo pessoal e sobre a sua organização.

V. CONCLUSÕES

A literatura utilizada não analisa as tarefas e os utilizadores de um SGIM de forma sistemática. Esta dificuldade levou-nos, algumas vezes, a basear essa análise em factores empíricos. Estamos convencidos que, apesar disso, o interface projectado se adapta convenientemente aos utilizadores e tarefas em causa. Parece-nos que um estudo aprofundado dos utilizadores e

das tarefas associados a um SGIM poderia constituir um trabalho de grande relevância para esta área, principalmente no que diz respeito ao estudo das características que influenciam o desenvolvimento de interfaces de utilizador.

Na fase de projecto, preocupámo-nos em seguir um conjunto básico de princípios de construção de interfaces. Estamos convencidos que o interface projectado está, no essencial, em conformidade com esses princípios. Ao proceder-se à implementação do interface usando aplicações reais é essencial que esses princípios continuem a ser observados.

O interface projectado foi simulado usando *Microsoft® Visual Basic™*. Não foi possível apresentar essa simulação a utilizadores reais para avaliação de resultados. No futuro, seria necessário fazer uma avaliação da receptividade de utilizadores tipo ao interface proposto. O desenvolvimento de um interface de utilizador é, na maior parte das vezes, um processo iteractivo.

REFERÊNCIAS

[1] Deborah J. Mayhew. Principles and Guidelines in Software User Interface Design. PTR Prentice Hall, 1992.

[2] J. Foley, A. van Dam, S. Feiner, J. Hughes. Computer Graphics. Addison Wesley, 1990.

[3] Rui Ribeiro. Desenvolvimento de um protótipo de um sistema de gestão de imagens médicas e comunicações. Tese de Mestrado, Novembro 1994.

[4] L. Elliot, S. Mun, S. Honii, H. Benson. Digital Imaging Network System Evaluation Report. Georgetown University Medical Center, March 1990.

[5] R. Maroldi, G. Battaglia, G.L. Moscatelli and A. Chiesa. Integrated diagnostic imaging: digital PACS in medicine 1980-2000. In: J.P.J. de Valk (ed.), Integrated Diagnostic Imaging - Digital PACS in Medicine. Elsevier, 1992.

[6] R. Mattheus. Comunication services: a key issue in the PACS of the year 2000. In: J.P.J. de Valk (ed.), Integrated Diagnostic Imaging Digital PACS in Medicine. Elsevier, 1992.

[7] Rudi Van de Velde. Hospital Information Systems - The Next Generation. Springer-Verlag, 1992.

[8] J.P.J. de Valk. Preface. In: J.P.J. de Valk (ed.), Integrated Diagnostic Imaging - Digital PACS in Medicine. Elsevier, 1992.

[9] F.P Ottes, A.R. Bakker and J.M.L. Kouwenberg. Introdution, Definition and Historical Background of Picture Archiving and Communication Systems. In: Michel Osteaux (ed.), A Second Generation PACS Concept. Springer-Verlag, 1992.

[10] B.M. ter Haar Romeny and F.H. Barneveld Binkhuysen. PACS as an infrastructural essencial element in an advanced digital imaging department. In: J.P.J. de Valk (ed.), Integrated Diagnostic Imaging - Digital PACS in Medicine. Elsevier, 1992.

[11] G. Gell and M. Wiltgen. Digital PACS in medicine 1980-2000. In: J.P.J. de Valk (ed.), Integrated Diagnostic Imaging Digital PACS in Medicine. Elsevier, 1992.

[12] T. Wendler, K.J. Mönnich and J. Schmidt. Digital Image Workstations. In: Michel Osteaux (ed.), A Second Generation PACS Concept. Springer-Verlag, 1992.

[13] Rudi Van de Velde, et al, (eds). Concept for a HIPACS Architecture. In AIM Project - Foudations for a Hospital Integrated Picture Archiving and Communication System (HIPACS) - Report 4, October 1990.

[14] Rui Ribeiro e António Sousa Pereira. Funcionalidades para um Sistema de Gestão de Imagens Médicas. Relatório interno, Março de 1992.

[15] O. Ratib, Y. Ligier, D. Hochstrasser and J. Scherer. Hospital Integrated Picture Archiving and Communication System (HIPACS) at the University of Geneva. Medical Imaging V: PACS Design and Evaluation, R.G. Jost, Editor, Proc. SPIE, pp 300-340, 1991.

[16] M. Osteaux, et al.. Medical Requirements for Clinical Integration. In: M Osteaux et al. (eds) Hospital Integrated Picture Archiving and Communication Systems - A Second Generation PACS Concept. Springer-Verlag, 1992.

[17] F. H. Binkhuysen. Required functionality of PACS from clinical point of view. In: International Journal of Bio-medical Computing, Vol. 30, No. 3/4, May 1992.

Programação Orientada por Objectos aplicada à simulação de redes neuronais

Pedro Kulzer, António Branco, Ana Tomé, Armando Pinho

Resumo - **Neste artigo descrevemos uma representação orientada ao objecto, destinada à implementação de redes neuronais artificiais. Esta representação é constituída por vários objectos-base que possuem, cada um deles, um nível diferente de representação. A ideia essencial centra-se numa representação *bottom-up*, em que se implementam objectos cada vez mais globais que possuem dentro de si, objectos da posição hierárquica imediatamente inferior. Espera-se assim conseguir uma representação global incremental fácil de compreender, imaginar e programar. Descrevemos um modelo global de um sistema neuronal, focando as três características essenciais: topologia, actualização das saídas e treino.**

Abstract - **In this paper, we describe an object oriented representation that is useful for the programmable implementation of artificial neural networks. This representation is basically composed of several base objects which achieve, every one of them, a different level of representation. The main idea is based on a bottom-up representation, where we implement objects that get more and more global, containing other objects of an immediate lower hierarchical position. With all this, it is expected to achieve an incremental global representation which is easy to understand, imagine and program. We describe a global neural system model, focusing the three essential characteristics: topology, recall and train.**

I. INTRODUÇÃO*

Uma rede neuronal é constituída por neurónios ligados entre si, pressupondo-se assim estruturas de processamento local, distribuído e paralelo. No entanto, o princípio de funcionamento da maioria das estruturas neuronais é suportado pelo conceito de camada, definindo-se camada como um grupo de neurónios (com ou sem as mesmas características funcionais), que recebem sinais com a mesma proveniência e enviam sinais com destino comum. Assim, uma rede neuronal pode definir-se como sendo constituída por uma ou várias camadas. Podemos ainda considerar um nível superior nesta hierarquia, definindo um sistema neuronal como sendo constituído por várias redes. Deste modo, concluímos que um sistema neuronal pode ser descrito como sendo uma estrutura hierárquica, constituída por vários níveis de abstracção diferente (blocos). Aos diversos blocos do sistema neuronal são atribuídas características funcionais diferentes, quer a nível da topologia, cálculo das actuais saídas dos diferentes neurónios e treino.

Posto isto, e passando a falar em termos de programação orientada a objectos, podemos dizer que os sistemas neuronais, são sistemas estrutural e conceptualmente complexos, compostos por subsistemas interrelacionados, que por sua vez possuem os seus próprios subsistemas e assim por diante, até que seja atingido o nível mínimo de componentes elementares. Nestes sistemas, as interacções intracomponentes são geralmente mais fortes do que as interacções intercomponentes. Este facto tem o efeito de separar a dinâmica de alta frequência dos componentes (envolvendo interacções na estrutura interna dos respectivos objectos) da dinâmica de baixa frequência (envolvendo interacções entre aqueles objectos). Assim na programação orientada ao objecto, os dados estão ligados às rotinas de código que executam operações sobre eles, encapsulando código e dados no mesmo objecto.

Os sistemas hierarquizados, que são uma particularidade da programação por objectos, são usualmente compostos por apenas alguns subsistemas diferentes, com várias combinações e arranjos, o que corresponde à estrutura do sistema neuronal [6].

No fundo, para uma implementação correcta duma biblioteca neuronal com base em objectos, é imprescindível ter em conta as seguintes regras de programação orientada a objectos [7]:

i) Cada ideia diferente é uma classe diferente, isto é, devem-se implementar classes diferentes para representar comportamentos diferentes de entidades intrinsecamente diferentes (Ex: Ligação, Neurónio).

ii) Cada entidade separada e do mesmo tipo, é também um objecto separado e do mesmo tipo, isto é, cada entidade é uma realização individual da respectiva classe (Ex: Neurónio1, Neurónio2, Neurónio3 na camada I).

iii) Se duas classes possuem dados e código significativos em comum, faz-se uma classe base com esses dados e código, isto é, os dados e código diferente são colocados em classes descendentes desta classe-base (Ex: A classe-base é o SupervisedNeuron e as classes descendentes são as PerceptronNeuron e AdalineNeuron).

* Trabalho realizado no âmbito da disciplina de projecto.

iv) Se uma classe é um tipo mais específico de uma outra classe mais geral, então a primeira será herdada da segunda (Ex: Um neurónio que é treinado pela regra de Hebb é herdado de um outro que já possui código básico para neurónios treináveis).

v) Se um objecto possui outros objectos, então estes últimos serão membros do primeiro (Ex: Uma camada contém neurónios; um neurónio contém ligações).

As vantagens da programação por objectos são mais notórias quando se tentam implementar sistemas complexos em computador, visto que esta forma de programação se aproxima mais da estrutura real abstraída pela mente humana.

A seguir, vamos proceder ao detalhe deste tipo de programação, em que primeiro falaremos do modelo global de um sistema neuronal, em termos absolutamente gerais, sem nenhuma intenção inicial de descrição do respectivo treino, etc. Depois da descrição da estrutura base deste modelo de programação, procederemos ao detalhe mais específico de cada uma das características desse modelo.

II. MODELO LOBAL DE UM SISTEMA NEURONAL

Um modelo neuronal tem três características principais a serem consideradas [5]:

- Topologia (forma das ligações entre os neurónios).
- Actualização dos valores de saída (forma pela qual são calculadas as saídas dos neurónios).
- Treino (forma de alterar os pesos para se atingir um determinado fim).

Para se obter uma grande flexibilidade de implementação dum modelo neuronal, estes princípios devem atender aos seguintes requisitos básicos:

i) A topologia deve permitir qualquer estrutura de ligações (campos receptivos, realimentações, ligação ponto-a-ponto, etc.).

ii) A actualização dos valores de saída, além do processo de apenas uma propagação dos sinais de entrada para a saída, deve também permitir processos mais complexos (actualização da saída, recursivamente, até atingir um valor estável).

iii) Deve ser possível, além de se usar os treinos previamente implementados, redefinir novos métodos de treino que possam surgir (através da capacidade de *polimorfismo* [1] das classes de treino).

Uma rede neuronal é constituída por neurónios ligados entre si, usualmente ordenados em camadas. A representação desta rede é feita a partir da unidade mais simples: a ligação. Os neurónios possuem ligações, estão dispostos em camadas que os contêm, que por sua vez estão contidos em camadas numa rede neuronal. Por último, a rede pode fazer parte dum conjunto de redes, iguais ou não, contidas num objecto ainda mais global, que é o *sistema* neuronal. Assim obtém-se uma representação hierárquica do tipo *bottom-up*, onde cada objecto é contido no objecto imediatamente superior, tal como se pode ver na Fig.1. Nesta figura visualizam-se os diferentes níveis da hierarquia de um sistema neuronal complexo, sendo cada nível constituído por objectos que contêm uma lista ligada de objectos do nível imediatamente inferior, isto é, cada *Sistema* contém uma lista ligada de *redes*, cada *rede* contém *camadas*, cada *camada* contém *neurónios* e cada *neurónio* contém *ligações*.

Na Fig.2 pode-se observar um exemplo de um sistema neuronal muito simples, constituído por apenas uma rede, duas camadas, três neurónios e sete ligações, onde aparecem as componentes neuronais discretizadas. Na parte superior aparece a topologia da rede, enquanto que na parte inferior se detalhou a representação orientada aos respectivos objectos.

Note-se que existe sempre apenas um objecto do tipo *Sistema*, enquanto que existe um ou mais objectos dos restantes tipos para cada objecto-ascendente. Por exemplo, um neurónio tem de ter, pelo menos, uma ligação.

Ao implementar as várias classes de objectos, teremos encapsulados os dados e o respectivo código que os gere. O exemplo que se segue é específico para o caso do neurónio, que será invariavelmente a estrutura de dados mais complexa de um sistema neuronal. Este exemplo está escrito em C++.

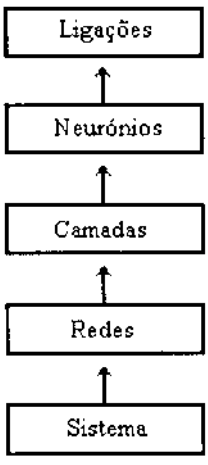


Fig. 1 - Representação hierárquica do tipo *bottom-up*.

```
class TCNeuron:TCStreamObject
{
public:
 TCNeuron();
 virtual ~TCNeuron();
 virtual TCLink  *CreateLink(TCNeuron *Source);
 virtual void    DestroyLink(TCNeuron *Source);
 virtual TWeight     GetBias();
 virtual TLearnRate  GetDecay();
 virtual TLearnRate  GetEpsilon();
 virtual TCLink     *GetLink(TLinkIndex Index);
 virtual TLinkIndex  GetLinkCount();
 virtual TLearnRate  GetMomentum();
 virtual TInOutValue GetOutput();
 virtual TBoolean    HasLink(TCNeuron *Source);
 virtual TBoolean    IsBiasUsed();
 virtual void        Recall;
 virtual void        Save(TCStream *Stream);
 virtual void   SetEpsilon(TLearnRate Epsilon);
 virtual void   SetMomentum(TLearnRate Moment);
 virtual void Train(TInOutValue DesiredOutput);
   . . .
protected:
 TWeight     Bias;
 TBoolean    BiasFrozen;
 TBoolean    BiasUsed;
 TBoolean    Conscious;
 TLeranRate  Decay;
 TLearnRate  Epsilon;
 TLearnRate  Momentum;
   . . .
};
```

O *constructor* TCNeuron() inicializa os dados quando o objecto é criado em memória e o *destructor* ~TCNeuron() destrói-os na altura da sua remoção da mesma.

A palavra-chave *public* permite o acesso aos métodos desta classe, enquanto que o *protected* impede acessos directos às variáveis mas permitindo esse acesso a objectos de classes descendentes desta. Esta última permite que novas classes possam estender as funções dos

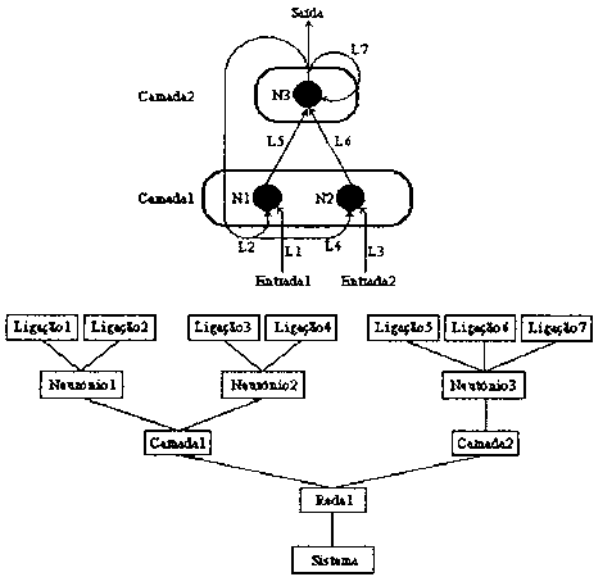


Fig. 2 - Exemplo dum sistema neuronal com apenas uma rede constituída por três neurónios e duas camadas.

métodos de treino e outros.

Apenas se mostraram algumas das muitas funções e campos de dados, que são incluídos neste tipo de estrutura [3]. Note-se que todas as outras estruturas base (ligação, camada, rede e sistema) também serão implementadas de forma semelhante. Cada uma destas classes, tal como esta TCNeuron, implementam funções destinadas às suas várias tarefas (responsabilidades). Para se poder realizar vários tipos de comportamento dos objectos associados a estas classes, temos duas hipóteses de implementação:

i) Existe uma única classe para cada nível hierárquico que possui todas as funções possíveis de uma só vez. Assim, ter-se-iam várias funções que implementariam diferentes comportamentos para uma certa tarefa.

ii) Usa-se a capacidade de polimorfismo através de métodos virtuais em classes derivadas, cada uma com o seu comportamento específico e descendendo de uma classe base abstracta (*abstract base class*).

A desvantagem da primeira hipótese, é que torna a estruturação muito deselegante e "amontoada" quando se deseja acrescentar novos comportamentos a uma *class* já existente, e cujo código, porventura, não é acessível ao programador. Enquanto isso, na segunda hipótese, o programador não necessita modificar esta *base class*, tendo apenas de criar classes descendentes daquela. Estas últimas implementariam as suas próprias funções que alteram o comportamento da classe base, funções essas que seriam automaticamente chamadas por esta classe-base através do processo de funções *virtuais* (*runtime late binding*). Uma boa implementação é tornar esta classe-base uma classe *abstracta*, isto é, os métodos de treino, actualização dos valores de saída, e outros que não realizem funções tão básicas como a escolha ou recolha directos de valores de variáveis, são tornados *virtuais puros* (sem código associado nessa classe).

Por exemplo, o programador dispõe de uma biblioteca neuronal, que possui *classes* pré-definidas (todas elas de alguma forma descendentes da *classe-base abstracta* TCNeuron). Estas classes conteriam o código necessário para a simulação de alguns tipos de neurónios mais utilizadas, tendo a possibilidade de criar novos objectos com comportamentos diferentes (*user-defined*). Todos os outros níveis hierárquicos respeitariam uma forma semelhante de implementação. Na Fig.3 apresenta-se um diagrama de caminhos de descendência de possíveis classes pré-definidas.

Para ilustrar esta forma muito conveniente e simples de redefinir um método, alterando assim o comportamento da classe ascendente, apresentamos de seguida um pequeno exemplo de neurónios em que o comportamento relativo ao treino é alterado.

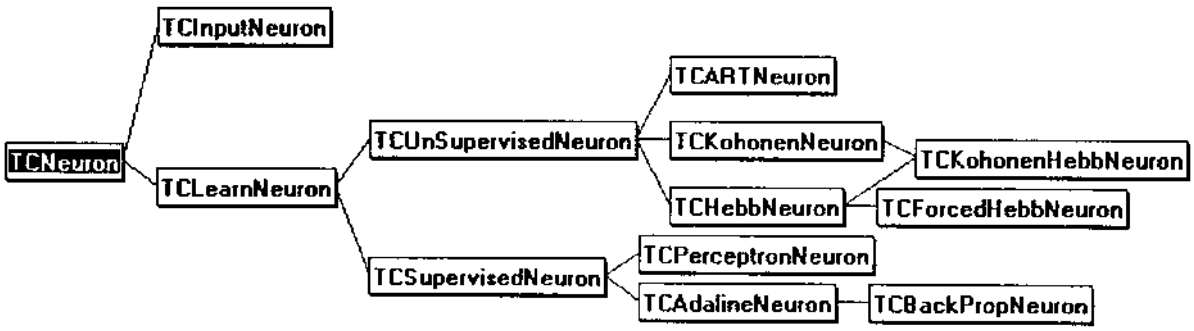


Fig. 3 - Ilustração em forma de diagrama de caminhos de descendência das diferentes classes de neurónios, implementada numa biblioteca neuronal standard. Note-se que se pode conjugar dois tipos de neurónios diferentes numa só classe, usando hereditariedade múltipla, tal como vê no TCKohonenHebbNeuron.

```
class TCHebbNeuron:TCUnSupervisedNeuron
{
. . .
virtual void NormalizeWeights();
virtual void Train();
. . .
};


class TCMyHebbNeuron:TCHebbNeuron
{
. . .
virtual void NormalizeWeights();
. . .
};
```

Neste exemplo, temos uma classe que implementa as funções dum neurónio que aprende segundo a *regra de Hebb*. O método *Train* é chamado sempre que se queira treinar o neurónio, através da classe base TCNeuron. É este método *train* que chama outro método, *NormalizeWeights*, que ocorre ao nível da classe TCHebbNeuron. Neste primeiro objecto TCHebbNeuron, o método *NormalizeWeights* desempenha alguma função *standard* tal como normalizar todos os pesos do respectivo neurónio. A segunda classe foi uma classe definida pelo programador, que vai permitir alterar o comportamento *standard* da primeira classe, redefinindo o método *NormalizeWeights*, que vai desempenhar algo diferente (por exemplo a normalização de todos os pesos excepto alguns). Desta vez, o *Train* vai chamar o *NormalizeWeights* localizado na nova classe TCMyHebbNeuron, por ser declarado virtual. Estamos aqui a utilizar a capacidade de *polimorfismo* dos objectos, já que uma *class* pode manifestar comportamentos diferentes, através do *overriding* de métodos seus em classes derivadas.

Conclui-se daqui que é aconselhável implementar esta biblioteca neuronal base, com todos os métodos ligados ao treino, etc. com a palavra-chave virtual, de forma a permitir a máxima flexibilidade e simplicidade na alteração dos comportamentos dos objectos dessa biblioteca. Em princípio, a classe base TCNeuron, de onde derivam todas as outras, será uma *abstract base-class* que apenas define os cabeçalhos de todos os métodos virtuais possíveis, mas sem nenhum código associado. Este código será implementado nas *classes* derivadas desta, que farão o *override* destes métodos conforme a necessidade. Por exemplo, a *class* TCInputNeuron apenas pode fazer pouco mais do que o *override* do método de actualização dos valores de saída, enquanto que classes mais sofisticadas poderão fazê-lo a todos.

O mesmo tipo de tratamento que se fez anteriormente, pode ser realizado também para as classes TCLink, TCLayer ,TCNetwork e TCNeuralSystem.

Note-se que este modelo global possui dois tipos de hierarquização:

i) Hierarquização a nível da representação de um sistema neuronal, através da individualização das suas diversas componentes abstractas: ligação, neurónio, camada, rede, sistema. Esta tem a ver com a regra v) de programação orientada ao objecto, anteriormente mencionada.

ii) Hierarquização a nível de cada componente anterior, através da criação de classes com comportamentos diferentes mas tarefas (treino, etc.) rigorosamente iguais: neurónios de entrada, supervisionados, não-supervisionados, de Hebb, etc. Esta tem a ver com a regra iv) de programação orientada ao objecto, anteriormente mencionada.

III. TOPOLOGIA

A estrutura topológica é criada através de comandos que estabelecem um certo padrão de ligações (rede completamente ligada, campos receptivos, etc.), ou através de uma linguagem descritiva (Ficheiros *script*), sendo o primeiro mais fácil de utilizar, mas perdendo bastante flexibilidade.

A estrutura anterior, altamente hierarquizada, possui vantagens concretas na representação topológica, sem levantar grandes encargos adicionais à programação, tais como as que se seguem.

i) Cada ligação é uma entidade perfeitamente independente de tudo o resto, o que permite interligar dois neurónios, independentemente da sua localização ou tipo.

ii) Cada neurónio é uma entidade isolada, permitindo além de estruturas baseadas em camadas convencionais, qualquer outro tipo de distribuição (por exemplo um aglomerado de neurónios).

iii) A estrutura em camadas, permite que se criem padrões de ligações entre estas, facilitando a implementação de estruturas em sistemas complexos. (Varrimentos de campos receptivos, interligações recursivas na mesma camada, etc)

iv) A vantagem de ter o objecto *Rede*, como entidade isolada permite criar módulos que são os componentes básicos de uma estrutura neuronal super-complexa. (Permite criar os módulos do córtex visual; a Retina, as Hiper colunas, o classificador final).

v) O sistema tem como finalidade ter uma entidade que suporta as redes.

O modelo global apresentado inicialmente, em que se faz uma distribuição dos vários objectos de um sistema neuronal com base numa hierarquia de listas ligadas, permite que haja também uma distribuição do código, que faz a actualização dos valores de saída. Por outras palavras, há uma distribuição desta responsabilidade por entre os diversos objectos, o que contribui para:

i) Redução a um mínimo de código necessário em cada objecto.

ii) Tornar o código extremamente simples, porque cada objecto apenas se responsabiliza pelo nível hierárquico imediatamente inferior.

iii) Para se obter um código sem erros, com uma manutenção muito mais simplificada.

IV. ACTUALIZAÇÃO DOS VALORES DE SAÍDA

As características funcionais de base, a nível da actualização dos valores de saída dos neurónios, são:

i) Para calcular a saída do neurónio, não interessa a proveniência da ligação, visto que são todas tratadas da mesma forma.

ii) Os neurónios calculam localmente o valor da saída, conforme as respectivas funções de entrada e transferência e tempo de estabilização em caso de recursividade, através de uma mensagem de uma entidade superior.

iii) A camada limita-se a emitir mensagens de actualização dos valores de saída para os neurónios desta.

iv) A rede estabelece a ordem pela qual as camadas de neurónios são actualizadas.

v) O sistema permite que blocos separados tenham valores de saída que podem ser combinados, mas calculados em alturas diferentes.

V. TREINO

As características funcionais de base, a nível do treino dos neurónios, são:

i) O valor das ligações é alterado durante o treino, independentemente umas das outras, permitindo implementar regras que criem e destruam ligações.

ii) Cada neurónio possui uma regra de treino que lhe confere um comportamento próprio, completamente autónomo de todo o resto; o próprio neurónio é que possui toda a responsabilidade de ir utilizar os parâmetros necessários à sua regra de treino (valores de entrada, propagação de erros, etc.) .

iii) A camada envia mensagens a cada neurónio para se treinar (por definição uma camada deve ter a mesma regra de treino para todos os neurónios).

iv) A rede envia mensagens às camadas, que podem ter regras de treino diferentes.

v) O sistema permite englobar vários módulos com regras de treino diferentes.

Uma ilustração simples do código necessário para um ciclo de treino a partir da execução do método TCNetwork::Train() do objecto de rede, para a regra de Hebb simples, é o seguinte:

```
TCNetwork::Train()
{
  for (TWord L=0; L<GetLayerCount(); L++)
    GetLayer(L)->Train();
};


TCLayer::Train()
{
  for (TWord N=0; N<GetNeuronCount(); N++)
    GetNeuron(N)->Train();
};
```

```
TCNeuron::Train()
{
  for (TWord L=0; L<GetLayerCount(); L++)
  {
    TCLink *Link=GetLink(L);
    Link->SetWeight(Link->GetWeight()
                    +(GetEpsilon()*Link-
>GetInput())
  };
  NormalizeWeights();
};
```

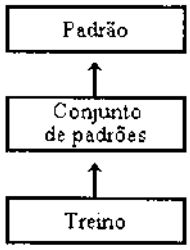


Fig. 4- Hierarquia de classes das estruturas de treino de um sistema neuronal.

O objecto *Sistema* está encarregue de gerar conjuntos de padrões de treino, teste, etc. intermédios (entre redes), conforme o necessário. Assim, cada rede é treinada individualmente, o que reduz significativamente o tempo de espera de resultados de redes intermédias, além de organizar melhor todo o complexo processo de treino de um sistema neuronal de tamanho significativo.

VI. PADRÕES DE TREINO E TESTE

Além da já referida hierarquização das componentes de construção das redes neuronais propriamente ditas, também se pode fazer uma hierarquização semelhante das componentes do respectivo treino. Estas serão subdivididas em Padrão, Conjunto de treino e Objecto de treino, tal como se mostra na Fig.4. Cada objecto do tipo padrão, conterá os dados referentes a um dado padrão. Cada conjunto de treino conterá uma lista ligada de padrões. Cada objecto de treino conterá uma lista ligada de conjuntos de treino, bem como dados e os respectivos métodos para a implementação de uma ou de mais estratégias de treino. Serão estes métodos que vão chamar os métodos de treino das respectivas redes, de acordo com aquelas estratégias.

De forma semelhante ao que acontece nas componentes do sistema neuronal, o Objecto de treino contém uma lista ligada de conjuntos de treino (treino, validação, teste e eventuais outros) e cada Conjunto de treino contém Padrões.

Como cada rede treinável tem de possuir uma estrutura de treino, o objecto Sistema possui ainda uma lista ligada de objectos Treino, associados às suas respectivas redes. Isto é visualizado na Fig.5 como exemplo de um sistema neuronal real.

Neste sistema neuronal temos quatro redes que formam um sistema artificial de visão, para reconhecimento de padrões. As redes da retina e do pós-processador não necessitam de treino, pelo que não têm nenhum objecto de treino associado. Apenas o classificador vai ser alvo de teste com um conjunto de padrões de teste. Como já foi dito anteriormente, estes conjuntos de treino intermédios são gerados pelo objecto do *Sistema* conforme a necessidade. Assim, neste caso específico da Fig. 5, o conjunto de teste será formado pelos padrões de saída do pós-processador, que vão servir para testar o classificador.

Resta dizer que tudo o que foi dito sobre *polimorfismo* e redefinição de métodos, também é aplicável a este conjunto de classes para treino. Pode-se, assim, redefinir o comportamento destas classes para conseguir alterar as estratégias de treino pré-definidas na biblioteca *standard* (tais como o critério de paragem, a sequência de apresentação de padrões, etc.).

Um exemplo de hierarquização de diferentes classes descendentes da classe-base abstracta TCPattern, é o que pode ver na Fig.6. Aqui, temos um objecto especializado para cada tipo de dados.

VII. CONCLUSÕES

A representação hierarquizada por classes de um sistema neuronal complexo apresentada possui todas as vantagens da programação orientada a objectos (polimorfismo, hereditariedade, encapsulamento, abstracção, reuso de código, modularidade) [1]. Estas características são amplamente usadas na vertente da topologia, actualização de saídas e treino da rede neuronal. Conseguiu-se, deste modo, atingir os requisitos a que um simulador de redes neuronais deve obedecer: a topologia deve permitir qualquer estrutura de ligações; a actualização dos valores de saída, além do processo de apenas permitir uma propagação dos sinais de entrada para a saída, deve também permitir processos mais complexos; deve ser possível, além de se usar os treinos previamente implementados, redefinir novos métodos de treino que possam surgir.

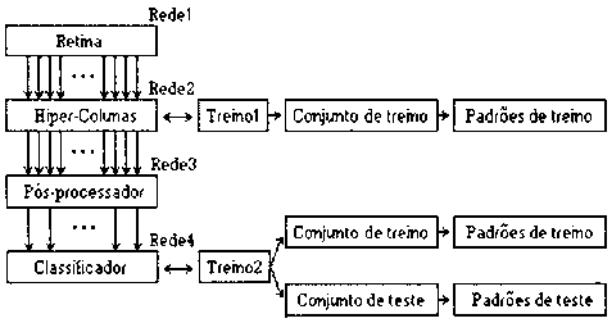


Fig. 5- Ilustração da relação estreita rede-treino.

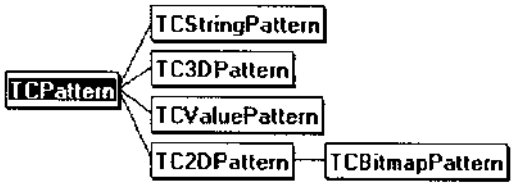


Fig. 6 - Ilustração esquemática dum exemplo de hierarquização das classes para os padrões de treino/teste.

Para finalizar, vamos resumir as principais consequências vantajosas oferecidas pela programação orientada a objectos, para este caso específico da implementação de sistemas neuronais complexos:

- **Modularidade**

Objectos bem concebidos são extremamente auto-contidos (não dependem de outros para funcionarem; são independentes), o que aumenta o nível de modularidade e de robustez, podendo estes ser reusados e estendidos na medida do necessário. Um objecto interage com o resto do programa duma forma muito restrita, isto é, o programa não deveria ter acesso aos dados dum objecto, a não ser através seus métodos. Além disso, os objectos devem-se apenas preocupar com os seus dados, sem ter de modificar variáveis globais exteriores.

Um objecto é como um circuito integrado (*chip*). Quando se quer construir uma aplicação, basta ligar alguns objectos diferentes de forma conveniente para obter um resultado altamente modular e estruturado. Isto leva a um conceito de controlo de avarias muito mais localizado e eficiente. Enquanto que em programas normais, o conserto dum erro de programação (*bug*) gerava mais dois ou três, na programação por objectos um erro não se propagará tão facilmente já que está confinado ao objecto em questão. Claro que isto só será verdade se os objectos comunicarem entre si duma forma estritamente necessária. Cada objecto até pode ser testado individualmente, sendo depois inserido no programa principal sem originar erros noutros objectos.

- **Segurança de utilização**

As mesmas características que tornam os objectos tão modulares, são responsáveis pelo seu elevado grau de confiança de utilização. Os objectos estão debilmente ligados uns aos outros, comunicando duma forma estritamente necessária. Portanto, o efeitos secundários de alterações a variáveis globais são minimizados, eliminando muitas oportunidades de aparecerem erros. Por causa dessa mínima comunicação, objectos que funcionam bem individualmente, têm elevadas probabilidades de funcionarem bem quando integrados no programa com outros objectos.

- **Reuso de código**

Durante a programação avançada dum projecto, poucas vezes se escreve uma rotina totalmente nova, limitando-se a fazer cópias modificadas para se adaptarem a cada situação (polimorfismo). Antes da programação por objectos, o reuso de código já feito era difícil e pouco modular. Mas com os objectos, este trabalho é grandemente facilitado. Para obter um comportamento duma rotina ligeiramente diferente para cada situação específica, basta criar um objecto descendente que implemente apenas uma actualização da parte do código que deverá ser diferente. Tudo o resto ficará igual.

- **Gravação e leitura através de *streams***

A hierarquização do sistema neuronal em objectos permite que se faça a transferência (leitura/escrita) de dados entre disco e memória, com base em *streams*, simplificando todo o código envolvido nesta operação.

Para permitir esta vantagem e simplicidade de se usarem *streams* para transferir um sistema neuronal entre disco e memória, na programação por objectos, cada classe envolvida tem de herdar a classe *TCStreamObject*. Conforme o compilador e a *runtime class library* usada, este nome pode variar.

Estas características são amplamente usadas em pacotes de ferramentas de programação que já incluem objectos-base, tais como a *ObjectWindows Library* [2] que contém objectos pré-fabricados para permitirem construir aplicações para o *Windows*. O que se pretende neste artigo é justamente dar os conceitos básicos de como se pode implementar um pacote de objectos-base (Ex: *ObjectNeural Library*), destinado à construção de sistemas neuronais de uma forma rápida, eficiente e segura, tal como no *ObjectWindows*.

Estes conceitos apresentados até aqui, foram resultado do estudo e construção de um simulador de redes neuronais, chamado de *NeuroCAD - Neural Network CAD-Simulator*, utilizado num projecto de reconhecimento automático de caracteres manuscritos[4], onde estavam envolvidas grandes quantidades de neurónios num sistema neuronal de elevada complexidade.

REFERÊNCIAS

[1] A. Porter, "C++ Programming for Windows", Osborne, McGraw Hill, 1993.

[2] Borland International, "ObjectWindows 2.0 programmer's guide", 1994.

[3] E. Schöneburg, N. Hansen, A. Gawelczyk, "Neuronale Netzwerke - Einführung, überblick und anwendungsmöglichkeiten", Markt & Technik Verlag AG, 1990.

[4] P. Kulzer, A. Branco, "Reconhecimento automático de caracteres", Publicação interna do DETUA, 1994.

[5] P.K. Simpson, "Foundations of Neural Networks", General Dynamics Electronics Division, .

[6] C.W. Lefebvre, J.C. Príncipe, "Object oriented artificial neural network implementation", Publicação interna.

[7] B. Stroustrup, "The C++ programming language", Addison-Wesley, 1985.

Simulação de um neurónio 100% analógico com processamento de corrente

Pedro Kulzer, António Branco, Dinis Santos

Resumo - **Projectou-se e simulou-se um neurónio 100% analógico, implementável em tecnologia CMOS de 1.2μm. O neurónio consiste em circuitos simples de processamento de corrente, desde as entradas das sinapses até à saída do axónio. Apenas a entrada do peso é em tensão, para atingir o máximo de rapidez de resposta. O neurónio tem um limiar de disparo adaptativo, bem como um circuito de aprendizagem não-supervisionada, autónoma e local.**

Abstract - **A 100% analog neuron, implementable in CMOS 1.2μm technology, was designed and simulated. The neuron consists of simple current processing circuits, right from the synapses' inputs to the axon's output. Only the weight input is in voltage form, to achieve the fastest possible response. The neuron has an adaptive firing threshold, as well as an autonomous, local, non-supervised learning circuit.**

I. INTRODUÇÃO*

Neste trabalho projectou-se um neurónio analógico em tecnologia CMOS de 1.2μm, com as suas diversas componentes: sinapses, limiar adaptativo, gerador de sigmóide e circuito de aprendizagem autónomo. Para construir uma rede completa, basta aglomerar o conjunto desejado destes neurónios analógicos e realizar as ligações entre eles.

II. MOTIVAÇÃO PARA ESTE TRABALHO

A motivação para este trabalho surgiu pela necessidade de se implementar uma rede neuronal treinável e suficientemente rápida para o processamento de sinais até 80 Mhz ou mesmo mais. Isto implica tempos de resposta da rede da ordem dos 10ns, o que apenas poderá ser conseguido através de neurónios completamente analógicos. A vantagem destes é que não há relógios nem acessos a memórias ou outros circuitos digitais "lentos". Apenas existe um tempo de resposta mínimo determinado pelo tempo de estabelecimento das saídas da rede analógica, e portanto, dos neurónios analógicos. O comportamento é semelhante ao de um circuito com amplificadores operacionais, dando origem a um tempo de estabelecimento global do sistema analógico.

Alguns argumentos que favorecem a opção por esta implementação 100% analógica são os que se seguem.

A. Tempo de resposta independente do nº de neurónios

As realizações digitais em computador exigem a execução das tarefas de cada neurónio, um de cada vez. A menos que se use multi-processamento, o tempo de resposta de cada camada é claramente proporcional ao número de neurónios em cada uma delas. Nas realizações analógicas em VLSI essa limitação não existe, já que se processa uma camada duma só vez, obtendo-se um puro multi-processamento.

B. Implementação fácil em VLSI

O facto de cada neurónio 100% analógico ser totalmente independente de sinais de controlo exteriores torna-o num módulo ideal para ser implementado numa estrutura VLSI com *layout* repetitivo. As implementações digitais ou mistas exigem a utilização de pistas que transportam sinais de controlo (*clocks*, *strobes*, *refresh*, etc.) a cada neurónio de forma individual, o que pode aumentar bastante a complexidade do projecto e exigir estudos de topologias de ligações que facilitem o trabalho.

O neurónio 100% analógico apenas necessita que lhe forneçam as linhas dos sinais de entrada, que também poderão ser apenas locais, facilitando assim o projecto do *layout*. Na Fig.1 mostra-se um exemplo de *layout*.

C. Tolerância a falhas

Justamente devido à natureza distribuída do *layout* em VLSI duma rede neuronal, essa rede será bastante resistente a falhas do próprio *chip* (poeiras e danos posteriores). Assim, perspectiva-se a possibilidade de aproveitamento quase total dos *chips* fabricados em cada *wafer* de silício. Só não se poderão aproveitar aqueles que ficaram demasiado danificados (partes periféricas da *wafer*, pistas

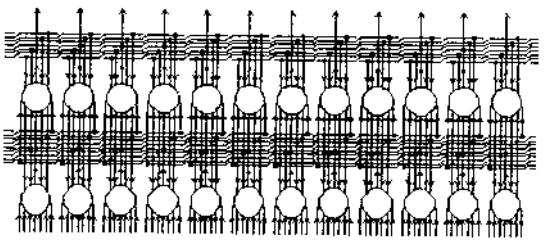


Fig. 1 - Exemplo de implementação de *layout* em que todas as ligações são locais (em forma de campo receptivo), estendendo-se as pistas de distribuição dos sinais pelo comprimento de apenas alguns neurónios. Desta forma, a área ocupada pelas pistas é mínima e o desenho facilitado.

* Trabalho realizado no âmbito da disciplina de VLSI analógico.

indispensáveis danificadas).

As redes existentes nos *chips* serão também resistentes a danos sofridos posteriormente durante a aplicação prática. Mesmo danificadas, estas redes continuarão a funcionar relativamente bem, dependendo da extensão dos danos (5 a 10%), podendo mesmo reorganizar-se e reaprender de forma a desempenhar melhor as suas funções com a funcionalidade que lhe resta. Isto é extremamente importante para aplicações espaciais (satélites, sondas, veículos lunares). Pelo contrário, num sistema computacional convencional, a ocorrência de uma falha numa única pista ou *bit* de memória compromete inevitavelmente o funcionamento.

D. Aprendizagem não-supervisionada incorporada

Uma rede que utilize uma aprendizagem não-supervisionada possui a potencialidade de distribuir essa aprendizagem por regras locais aos neurónios, o que também elimina a necessidade de pistas de sinais de supervisão da aprendizagem.

E. Adaptação às imprecisões dos neurónios

A presença da aprendizagem numa rede neuronal elimina à partida a necessidade de precisões impensáveis em VLSI. Qualquer não-linearidade, *offset* ou imprecisão de outra natureza, desde que mantida abaixo dum máximo crítico, será compensada pela aprendizagem efectuada em cada neurónio, à semelhança do que acontece num amplificador operacional realimentado: a saída é mais ou menos independente do valor exacto e da linearidade do ganho.

III. OBJECTIVOS

Propõe-se a implementação de um neurónio completamente analógico e autónomo, sem a necessidade de quaisquer circuitos externos. Este neurónio vai ter de obedecer aos seguintes requisitos:

A. Funcionamento dos transístores MOS na região de inversão forte

Para manter um baixo consumo optámos por uma implementação em tecnologia CMOS. Pretendemos um consumo por neurónio tão baixo quanto possível, sem que isso implique excessiva falta de precisão.

No entanto, a velocidade de processamento desejada não é tão baixa que permita um funcionamento na região de inversão fraca, pelo que se optou pelo funcionamento na região de inversão forte (acima da tensão de limiar, V_t).

B. Processamento de corrente

Nas implementações referidas na literatura, os sinais de entrada e de saída no neurónio são sinais de tensão, o que impõe limites estritos ao valor das capacidades de carga. As velocidades de processamentos são baixas, da ordem das centenas de ns (aproximadamente de 200ns no *chip* de Choi [2]). Como se pretendem tempos da ordem dos 10 ns, se possível com a tecnologia actual, é quase inevitável a opção pelo funcionamento dos circuitos no chamado modo de corrente Na Fig.2 apresenta-se um modelo de neurónio em modo de corrente.

Uma simplificação adicional consiste na eliminação de praticamente todos os circuitos de conversão corrente-tensão e tensão-corrente, excepto nas entradas.

C. Simplicidade de implementação

Como queremos obter um neurónio o mais simples possível mas funcional, de forma a conseguir níveis de integração em VLSI o mais elevados possível, teremos de manter o nível de complexidade por neurónio também o mais baixo possível. Isso será especialmente crítico nas sinapses, já que cada neurónio pode ter centenas ou até milhares de sinapses, ocupando estas a maior parte da área do *chip*.

Sempre que possível, utilizaram-se dimensões mínimas para os transístores. Isso terá vantagens no espaço ocupado, bem como na minimização das capacidades parasitas o que aumenta a velocidade, sem ter desvantagens resultantes da transcondutância menor destes transístores, já que isso vai ser pouco importante.

D. Velocidade elevada

Como estamos interessados em obter a máxima velocidade de resposta possível para estes neurónios, tivemos de ter algum cuidado no projecto e teste dos vários circuitos. A alínea anterior já implica, só por si, uma maior velocidade de resposta do que em circuitos mais complexos, devido ao menor número de transístores envolvidos. No entanto, é possível que tenhamos de sacrificar alguma simplicidade para obter maior rapidez, e vice-versa. As dimensões dos transístores poderão não ser mínimas para se conseguir esse objectivo.

IV. PROCESSAMENTO DE CORRENTE

Teremos um modelo exterior em que as entradas e as saídas são sinais de corrente. É óbvio que no seu interior também se vai manter a operação em modo de corrente, pelo menos no que respeita às partes do circuito em que se exige a máxima velocidade de resposta.

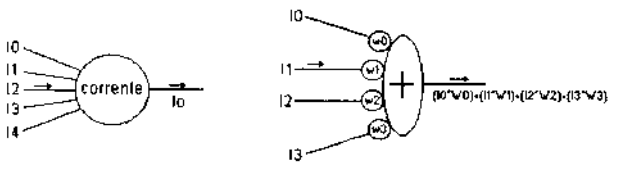


Fig. 2 - Modelo do neurónio a operar em modo de corrente. Os sinais de entrada são multiplicados pelos pesos e somados no corpo do neurónio. Se este tivesse uma função de saída linear, teríamos à saída uma soma pesada das entradas.

No multiplicador de *Gilbert* utilizado por Choi [2], são empregues as técnicas de *cascode* e fontes de corrente anteriores para melhorar a sua resposta, o que aqui já está implícito na própria arquitectura.

Os aspectos mais importantes a testar foram os tempos de resposta, a linearidade e as distorções.

V. SINAPSE

A sinapse é o lugar biológico onde se transmite um potencial excitatório ou inibitório ao corpo do neurónio, cuja intensidade depende directamente do factor eficiência da sinapse. Este processo pode ser visto como uma multiplicação aproximada, do sinal de entrada por um valor a que chamamos *peso*. Isto significa que o neurónio vai receber um potencial que será uma soma pesada dos sinais de entrada.

Cada sinapse vai ser constituída por um circuito multiplicador que vai receber um sinal de entrada e um sinal de peso, gerando uma saída que é o produto dos dois. Como não se conseguem multiplicar duas correntes, temos de utilizar um dos circuitos multiplicadores habituais. Podemos escolher entre o multiplicador de *Gilbert* e multiplicadores mais simples e com menos quadrantes de operação. O multiplicador tem que obedecer o melhor possível às seguintes especificações:

A. Simplicidade de implementação

Não queremos um circuito complicado e altamente preciso, bastando-nos um circuito que cumpra a sua função com o mínimo de precisão e transístores. Assim, no caso no modelo desejado em que apenas se querem saídas positivas, podemos utilizar um simples amplificador de transcondutância.

B. Entrada em corrente

O amplificador de transcondutância proposto possui uma entrada em tensão que terá de ser convertida numa entrada de corrente. Para isso, basta ligar essa entrada em corrente directamente ao par diferencial.

A função de transferência deste multiplicador (ignorando o efeito de corpo), será da forma:

$$i_{out} = g_{m_{bias}} \cdot \Delta v = \frac{i_{in}}{V_{GS} - V_t} \cdot \Delta v \qquad (1)$$

VGS é a tensão média nos transístores do par diferencial, e v é a diferença de potencial entre os dois condensadores.

C. Corrente mínima igual a zero

No que respeita à corrente mínima de entrada, esta será de 0A (ausência de actividade), o que pode causar problemas no funcionamento do par diferencial para correntes muito pequenas. De qualquer forma, esta é uma exigência que terá de ser cumprida pois as entradas podem variar entre zero e um valor máximo. Se houver problemas demasiados na zona de baixas correntes de entrada, teremos eventualmente de recorrer a multiplicadores cujo circuito esteja permanentemente polarizado a uma dada corrente mínima maior que zero.

D. Distribuição das velocidades pelas entradas

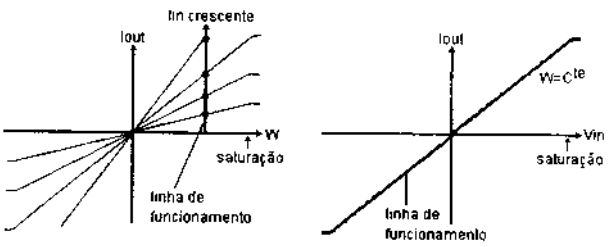
Já que este multiplicador tem de ser mesmo implementado pelo amplificador de transcondutância proposto, existe o problema da entrada diferencial em tensão. Esta será a única entrada em modo de tensão, o que reduzirá a velocidade de resposta de todo o circuito a essa entrada. Para minimizar este problema aparentemente sem solução, podemos ligar o sinal mais crítico à entrada mais rápida. Em princípio, vai interessar ter velocidade elevada para as entradas em corrente das sinapses.

As entradas em tensão referentes aos pesos não vão ser críticas pois estes variarão mais lentamente, além de atingirem estados aproximadamente estacionários durante os processos de aprendizagem da respectiva rede onde estarão integrados.

Uma vantagem resultante desta configuração das entradas, é que com um dado valor de peso abaixo dum limite máximo que provocaria saturação, o circuito virtualmente não satura enquanto se aumentar a entrada em corrente. Além de algumas não-linearidades, o circuito responde com uma corrente de saída aproximadamente proporcional à de entrada, aparentemente sem limitações de saturação. A única limitação é devida ao ponto em que as polarizações deixam de funcionar correctamente, devido a tensões dreno-fonte elevadas demais para a alimentação conseguir produzir. Isto é ilustrado em forma de gráficos no Grf.1.

E. Peso numa só capacidade

A nossa proposta é de armazenar cada peso numa só capacidade, sob a forma duma carga positiva. Isto simplifica grandemente o circuito de actualização dos pesos, apesar de degradar ligeiramente a resposta do par diferencial, devido ao ponto central já não estar à massa para sinais. O circuito resultante e as respectivas equações são mostrados na Fig.3.



Grf. 1 - À esquerda mostra-se a linha de funcionamento do multiplicador por nós proposto. Aí vê-se claramente que, para um dado peso abaixo do valor que provocaria saturação, obtém-se uma resposta linear para qualquer valor da corrente de entrada também abaixo de um valor máximo. Este valor máximo será apenas fixado pela tensão de alimentação. À direita temos a linha de funcionamento dum multiplicador em que o sinal de entrada é aplicado às entradas do par diferencial. Qualquer que seja o valor do peso, há um valor limite de tensão de entrada.

Na sinapse implementada por Barranco [1] também se utiliza um único condensador para o armazenamento do respectivo peso.

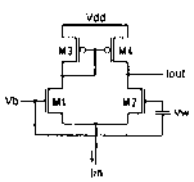
F. Ausência de refrescamento dos pesos

Como o objectivo da implementação destes neurónios analógicos era o de colocá-los a trabalhar a velocidades de aprendizagem da ordem dos 100Mhz, certamente que as fugas dos condensadores não terão qualquer importância significativa. As actualizações deles serão tão rápidas, que as fugas são constantemente compensadas. Além disso, não tem qualquer interesse prático conseguir manter os valores dos pesos durante longos períodos de *stand-by*, já que estes neurónios se destinam a serem integrados num sistema altamente adaptativo que vê o ambiente a mudar constantemente, pelo que terá de ser capaz de actualizar os pesos muito rapidamente. Esta última consideração elimina a necessidade de refrescamento. Mesmo que fosse necessário um refrescamento para processos mais lentos, poder-se-ia utilizar uma DAC neuronal como em Barranco [1].

G. Mínima dissipação de potência

A dissipação de potência é proporcional à corrente e tensão de alimentação da sinapse. Durante as experiências posteriores, vamos procurar valores razoáveis para estes dois parâmetros. Podemos desde já adiantar que a tensão *standard* de 3.3V utilizada nos processadores de baixo consumo, é perfeitamente adequada para a alimentação desta sinapse. Desta forma, garante-se uma parcela de 1.1V para a polarização de todos os transístores na zona de saturação (desde que a tensão de limiar, V_t, destes seja inferior àquele valor).

Com esta tensão de alimentação e uma corrente de entrada máxima de 5µA, o consumo seria de 16.5µW por sinapse. Se o sistema final contivesse 100 neurónios com 30 sinapses cada, o consumo total máximo (com todos os neurónios activos) seria de 50mW. Este consumo refere-se apenas às sinapses e teria de ser acrescido do consumo do circuito de geração de sigmóide, que também terá um consumo da mesma ordem de grandeza. Assim, o consumo de potência do sistema pode atingir os 100mW.



$$I_{out} = K \cdot I_{in} \cdot V_w$$

,em que

$$K = \frac{1}{V_{GS} - V_t}$$

Fig. 3 - No multiplicador proposto, o valor do peso é armazenado sob a forma de um condensador.

VI. DIMENSIONAMENTO E SIMULAÇÃO NO SPICE

Em todo o trabalho, o simulador de circuitos eléctricos PSPICE foi utilizado para analisar o comportamento, bem como para obter valores experimentalmente razoáveis para as diversas grandezas.

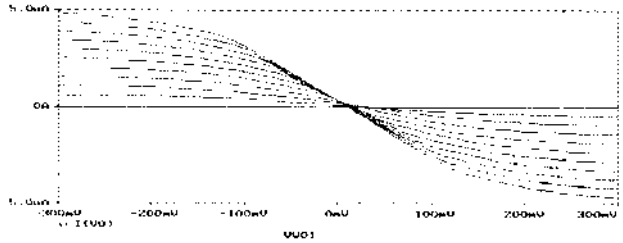
Quanto à corrente máxima de entrada, experimentámos inicialmente 5µA, o que poderá eventualmente ser baixo demais para atingir a velocidade de resposta desejada. Com esta tensão de alimentação, a tensão que se deverá colocar numa das entradas do par diferencial, será de 2V.

O valor máximo do peso corresponde à máxima diferença de potencial entre as entradas do par diferencial que ainda não provoca demasiada saturação. Este valor será certamente da ordem das centenas de millivolt, já que os transístores estarão polarizados no limite da sua zona de saturação (1.1V), pelo que qualquer perturbação poderá levá-los para fora dessa zona. Os resultados da simulação no Grf.2 mostram que esses limites são da ordem da centena de millivolt.

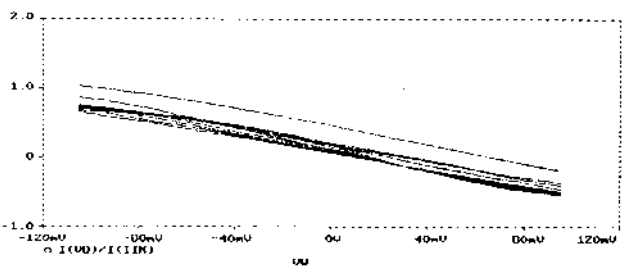
Podemos ainda observar a relação linear entre o peso e a saída do multiplicador, bem como o valor da constante K. Enquanto que a primeira curva devia ser uma linha recta com valores proporcionais apenas ao peso, já no segundo caso esperamos uma linha horizontal de valor igual àquela constante. Os resultados são mostrados nos gráficos Grf.3 e Grf.4.

Este valor experimental de K aproxima-se do valor teórico que se situará algures perto de:

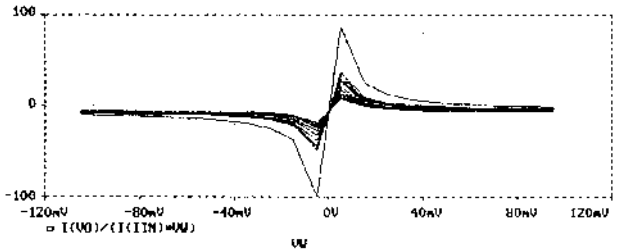
$$K_{teórico} = \frac{1}{1.07 - 0.77} = 3.3 \qquad (2)$$



Grf. 2 - Varrimento do valor do peso, obtido no PSPICE. Pode observar-se que a saída do multiplicador satura para pesos superiores a 100mV em valor absoluto. Assim, o valor máximo para os pesos será de 100mV.



Grf. 3 - Estas curvas representam o quociente da corrente de saída e de entrada, que se agrupam numa zona bem delimitada. A curva mais afastada das restantes é a referente a uma corrente de entrada de 0.01A. Para as correntes maiores já se verificou um menor erro.



Grf. 4 - Estas curvas representam o quociente da corrente de saída pela de entrada e pelo valor do peso. Desta vez já se obteve parte duma linha horizontal que tende para um valor de ordenada que só pode ser visto fazendo um *zoom*. A curva mais afastada das restantes é mais uma vez referente a uma corrente de entrada de 0.01A. Para as correntes maiores já se verificou um menor erro.

Podemos agora definir um *factor de rejeição da entrada*, para comparações futuras. Para peso nulo, designar-se-á aqui por *Zero Weight Input Rejection Ratio (ZWIRR)*:

$$ZWIRR = \frac{Iin}{Iout}\Big|_{Vw=0} \qquad (3)$$

$$= \frac{2\mu A}{360n} = 13.9(22.9dB)$$

O respectivo gráfico da variação da saída nestas condições, é mostrado no Grf.5.

Da mesma forma se podia introduzir o conceito de *Zero Input Weight Rejection Ratio*. No entanto, no caso presente esse valor seria infinito, uma vez que a corrente de saída é zero quando a corrente de entrada é zero.

Nas secções seguintes apresentam-se os resultados da simulação da resposta deste multiplicador isolado, para impulsos rápidos na entrada. Igualmente interessante é a observação da variação da velocidade d e resposta com os pesos, embora, como referido acima, este parâmetro seja menos crítico.

VII. CIRCUITO DE EXTRACÇÃO DA SAÍDA DO MULTIPLICADOR

Este circuito servirá ao mesmo tempo para permitir a junção de várias sinapses num ponto comum. Na Fig.4 podem-se ver os transístores M5 e M6 acrescentados para o efeito. Tal como já fizemos para o multiplicador isolado, vamos investigar os valores dos factores de amplificação obtidos por esta configuração com espelho de corrente na saída.

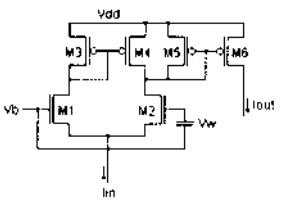
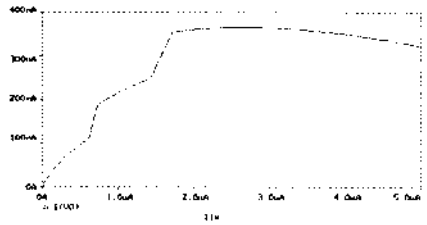


Fig. 4 - Os transístores M5 e M6 formam o circuito de extracção da corrente proveniente do multiplicador.



Grf. 5 - Corrente de saída com peso nulo. Esta corrente é baixa mas diferente de zero, o que pode ser útil, na medida em que um peso acidentalmente colocado a zero ainda pode aprender algo, pelo facto de ainda ter alguma acção.

Desta vez estes factores de amplificação já contém mais uma componente referente ao espelho de corrente de saída:

$$K = \frac{1}{V_{GS} - V_t} \cdot G_I \text{ , } c/ \quad G_I = \frac{W_{M6}}{L_{M6}} \cdot \frac{L_{M5}}{W_{M5}} = \frac{2.4}{6} \cdot \frac{24}{2.4} = 4 \qquad (4)$$

em que GI é o ganho em corrente fornecido pelo espelho de corrente da saída.

Um valor teórico para K seria:

$$K = \frac{1}{1.07 - 0.77} \cdot 4 = 13.3$$

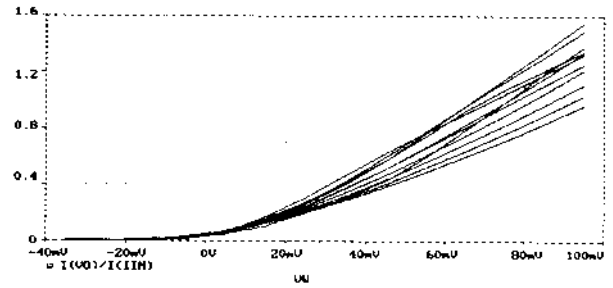
também considerando um peso nulo. Este valor é da ordem de grandeza dos valores experimentais. Neste caso, temos um *Zero Weight Input Rejection Ratio* de:

$$ZWIRR = \frac{5\mu}{180n} = 27.8(28.9dB)$$

Até este valor melhorou em relação ao anterior do multiplicador isolado (22.9dB). Isto será devido à distorção inicial da curva da corrente de saída do extractor, onde as correntes iniciais são mais baixas do que no multiplicador isolado original. No Grf.6 ainda se mostram as curvas correspondentes ao factor de amplificação da corrente de entrada para a de saída, para efeitos de apreciação da dispersão da mesma, relativamente ao multiplicador isolado.

VIII. JUNÇÃO DE SINAPSES

A junção de sinapses efectua-se muito simplesmente pela soma das suas correntes no ponto de entrada deste circuito extractor. Dessa soma resultará uma corrente que pode ser positiva ou negativa. Corrente positiva significa corrente no sentido de dentro para fora do extractor. Como é evidente, este tipo de extractor não funciona para correntes negativas (isto é, de fora para dentro), pelo que para essas correntes a saída é simplesmente zero. Na Fig.5 mostra-se a forma de interligação da várias sinapses. Cada sinapse introduz uma capacidade parasita resultante dos seus dois transístores de saída, o que reduz a velocidade de resposta global. Nos gráficos Grf.7 a Grf.11 mostram-se diversas situações de funcionamento que mais importa considerar.



Grf. 6 - Curvas referentes ao factor de amplificação que ainda contém o peso.

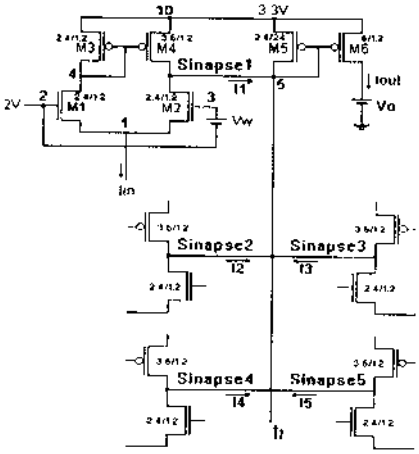
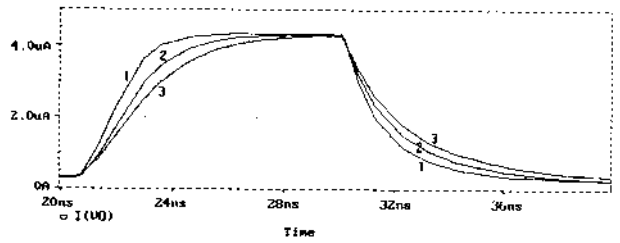
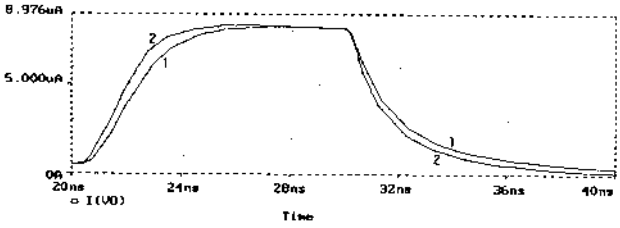


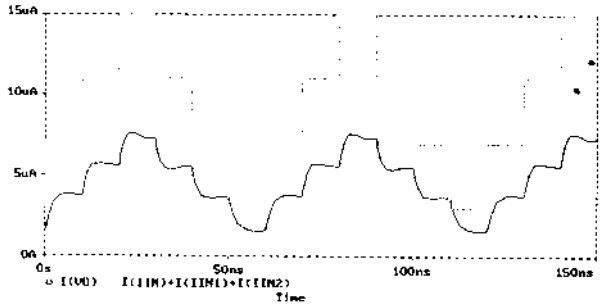
Fig. 5 - Aspecto final do circuito extractor de corrente com todas as sinapses associadas, penduradas no nó 5.



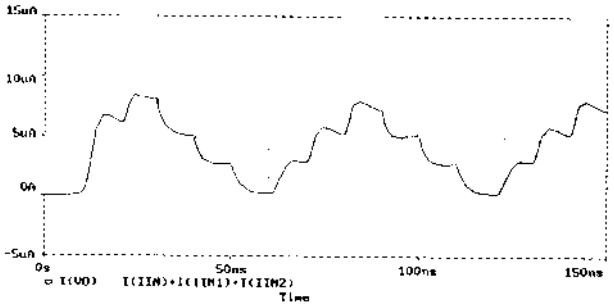
Grf. 7 - Respostas para uma só sinapse (1) e de duas e três sinapses - (2) e (3), sendo apenas uma sinapse excitada e as restantes não-excitadas. Pode-se concluir que, quantas mais sinapses se juntarem mais lenta fica a resposta à saída do extractor. Isto deve-se essencialmente às capacidades que cada sinapse acarreta para o nó 5 (Fig.5).



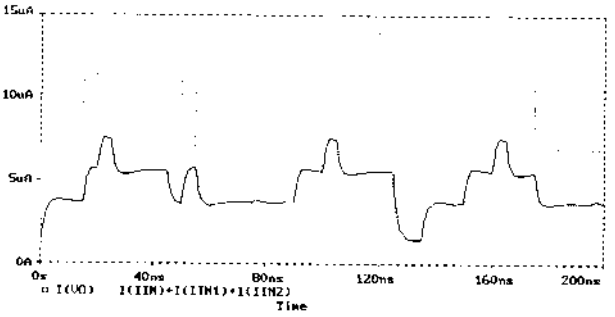
Grf. 8 - Respostas para apenas uma sinapse excitada (1) e para ambas excitadas (2) numa junção de duas sinapses. Quando ambas são excitadas, a velocidade de resposta é maior do que apenas uma sinapse excitada. Isso será devido à maior corrente disponível para carregar as capacidades parasitas às tensões para as quais o nó 5 (Fig.5) tem de variar.



Grf. 9 - Resposta para a excitação individual sobreposta de três sinapses unidas no nó 5 (Fig.5). Acima da curva da corrente de saída, está representada a curva da soma das excitações pesadas de entrada, para efeitos de comparação. Todos os pesos estavam a 0.1V, pelo que a soma directa das entradas (Iin+Iin1+Iin2) é semelhante à soma das correntes das sinapses (corrente de saída).



Grf. 10 - Resposta semelhante à anterior, mas com correntes de entrada a partirem do zero. Observe-se que a variação mais lenta da corrente de saída.



Grf. 11 - Resposta do circuito com três sinapses juntas. Todos os pesos são iguais a 0.1V e as correntes de entrada partem de 1µA. Mais uma vez, a corrente de saída segue as transições da curva da soma directa das entradas (Iin+Iin1+Iin2).

IX. GERADOR DE SIGMÓIDE

Na Fig.6 está representado o circuito do gerador de sigmóide. Este circuito segue o extractor de corrente das sinapses.

Para eliminar ou minimizar o *offset* na saída, podemos alterar as dimensões de M4, M3 ou M5. A modificação que resultou numa menor distorção na linearidade deste circuito, foi a feita no transístor M4 e resumiu-se a aumentar a relação W/L de forma conveniente: L=2.4µm e W=9.6µm. Para se conseguir o factor de ganho unitário desejado, teremos de fazer com que M2 desvie a corrente de M4 mais depressa, de forma a que M3 corte mais depressa. A modificação necessária para atingir este

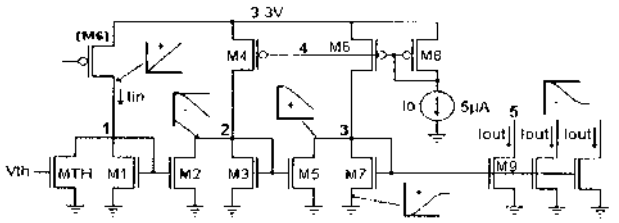


Fig. 6 - Esquema do circuito gerador de sigmóide. O transístor MTH impõe o limiar, já que só a partir duma corrente de entrada maior que a corrente em MTH, é que M1 iniciará a condução. M2 e M4 realizam a limitação superior de corrente, que terá de ser invertida por M5, M6 e M7 para se obter uma corrente a entrar em M9. No esquema estão representados os gráficos da corrente em cada linha, para a respectiva entrada de corrente a subir. Como se pode ver, as ramificações do axónio deste neurónio são feitas adicionando tantos transístores a seguir a M9 quantos se queira.

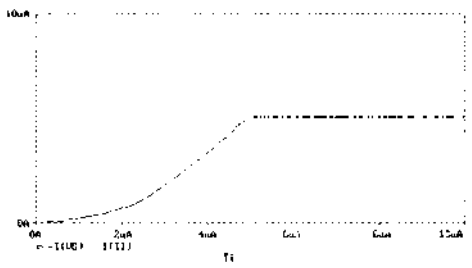
aquele objectivo foi um aumento da relação W/L de M2: L=2.4µm e W=13.2µm.

Por último, desejamos ainda corrigir a máxima corrente de saída para 5µA, tal como desejado inicialmente. Para isso, basta alterar as dimensões de M7 ou M9. Decidiu-se modificar apenas M9, que passa a ter as dimensões L=2.4µm W=6.0µm. Após estas alterações, pode-se observar o resultado final no Grf.12.

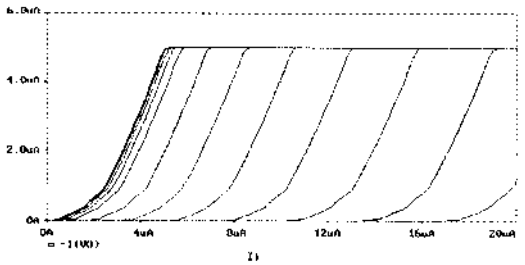
Para completar o circuito de activação do neurónio, resta apenas verificar o funcionamento do limiar de disparo realizado através do transístor MTH e a respectiva tensão de porta VTH. Esta tensão será gerada por um condensador cuja carga será actualizada por um circuito especial de controlo adaptativo deste limiar. Tudo o que MTH faz, é desviar parte da corrente de entrada, de forma a que o circuito de sigmóide seja activado com desvio da corrente de entrada, ou seja, só a partir duma certa corrente de entrada. Aplicando uma tensão variável na porta de MTH, podemos observar as translações sucessivas que a função do circuito sofre, no Grf.13.

O facto do *MOSFET* só reagir a partir de V_t não constitui qualquer problema, já o circuito que vai regular o valor da tensão VTH adaptativamente, corrigirá esse problema de forma automática. A mesma coisa acontecerá com a não-linearidade entre essa tensão e o limiar de disparo resultante.

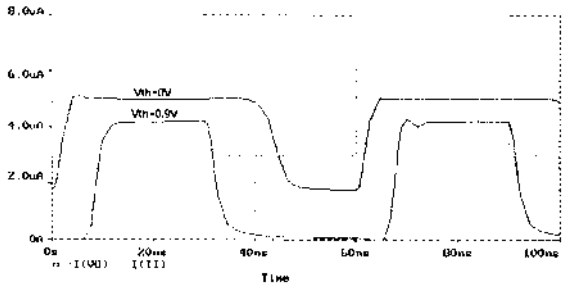
Nos gráficos Grf.14 e Grf.15 mostram-se os resultados das simulações que pretendem testar a rapidez de resposta do circuito completo.



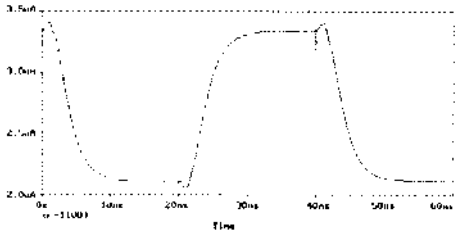
Grf. 12 - Função de transferência com todas as correcções introduzidas excepto a não-linearidade pronunciada na zona inicial da curva.



Grf. 13 - A função sigmóide sofre translações com uma relação quadrática em relação à tensão de limiar, como seria de esperar devido à característica quadrática da corrente do *MOSFET* em relação à sua tensão de porta.



Grf. 14 - Observa-se que o atraso diminui com a presença duma tensão de *threshold*. Significa que o pior caso é para limiares nulos. Os tempos medidos foram os seguintes: $T_{atraso\ subida}$=1.3ns, T_{subida}=3ns, $T_{atraso\ descida}$=7ns, $T_{descida}$=4.5ns. Isto significa que este circuito leva cerca de 4.3ns a responder a uma subida da entrada, e cerca de 11.5ns para uma descida, no pior caso. O circuito limita bem a corrente de saída a 5µA quando a de entrada ultrapassa esse valor.



Grf. 15 - Esta é a resposta à tensão do limiar de disparo. Com uma corrente de entrada de 4µA, os tempos obtidos nesta simulação são os seguintes: $T_{atraso\ subida}$=1.5ns, T_{subida}=7ns, $T_{atraso\ descida}$=1.5ns, $T_{descida}$=7ns. Estes tempos são da mesma ordem de grandeza dos observados para as respostas à entrada em corrente. De qualquer forma, não necessitariam de ser tão curtos, já que um neurónio adapta o seu limiar de forma mais lenta do que os restantes processos de adaptação.

X. CONTROLO AUTOMÁTICO DO LIMIAR DE DISPARO

O controlo automático do limiar de disparo consiste em modificar a tensão de *threshold* aplicada ao transístor que desvia a corrente que entra no gerador de sigmóide. Esta modificação deverá ser tal, que origine o fenómeno da *habituação* e *recuperação* dum neurónio.

Para conseguir simular um comportamento semelhante a este, basta adicionar malhas capacitivas que acumularão a carga equivalente ao limiar de disparo, sob a forma duma tensão VTH. Na Fig.7 mostra-se o aspecto do circuito de adaptação do limiar de disparo, já com todas as dimensões.

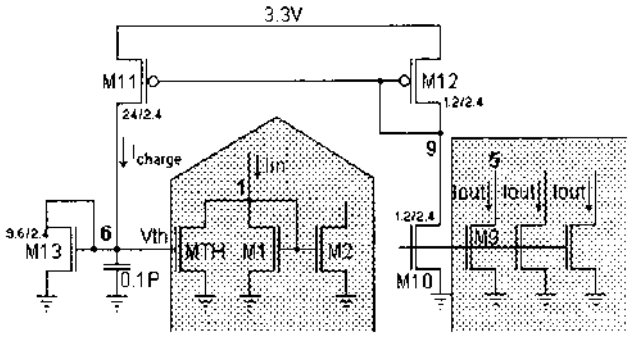


Fig. 7 - O circuito que controla automaticamente a tensão de *threshold* VTH é constituído pelos transístores M10, M11, M12 e M13, e pelo condensador de 0.1pF. M10, M11 e M12 realimentam corrente para o condensador que se carrega com maior ou menor velocidade consoante o valor da saída do neurónio detectado por M12. Saídas elevadas carregam o condensador mais rapidamente, pelo que o limiar sobe mais depressa. Se a saída for reduzida, o condensador carregará mais lentamente. O condensador também estará sempre a descarregar (muito menos) através de M13, pelo que se não houver saída este fará o limiar descer.

Por razões de economia de área, utilizou-se um condensador de baixo valor. As dimensões dos transístores M11 e M13 foram determinadas empiricamente. Para se colocar uma resistência em vez de M13, seria necessária um valor de 15MΩ, o que é inviável em VLSI, além de apresentar a desvantagem de poder descarregar totalmente o condensador.

Os valores relevantes a determinar são as variações de tensão e corrente em pontos importantes.

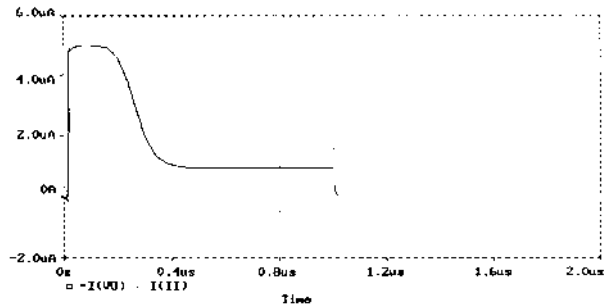
Este é apenas um exemplo. Para cada problema particular é necessário escolher convenientemente os tempos.

Duma forma geral, para alterar estes tempos de resposta do limiar, tem-se de actuar apenas nas relações (L/W) dos transístores M11 e M13.

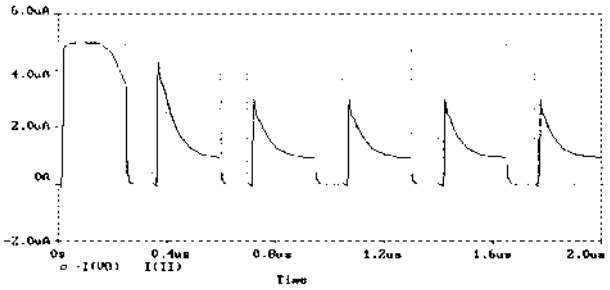
Nos gráficos Grf.16 e Grf.17 mostram-se duas situações em que o circuito de adaptação do limiar entra em acção.

XI. CIRCUITO DE APRENDIZAGEM

Este circuito implementará a *regra de Hebb modificada* que deverá obedecer ao requisitos que se seguem.



Grf. 16 - Curva da corrente de saída foi obtida com as dimensões dos transístores indicadas no texto: M11 – L=24μm, W=2.4μm; M13 – L=9.6μm, W=2.4μm. Como se pode observar pela curva da corrente de saída, esta desceu para aproximadamente 1μA após a habituação completa. Aí se manteve, até que a entrada foi a zero.



Grf. 17 - Resposta do neurónio a impulsos de corrente negativos com uma amplitude de 5μA e à frequência de cerca de 2.9MHz. Ao fim do segundo impulso, já o neurónio estava habituado, pelo que a partir daí ele mantinha a saída mais reduzida. Sempre que o impulso desaparecia, o limiar recuperava algo, até que o próximo impulso o voltava a descer.

- Coincidência de actividade na sinapse e na saída do neurónio implica uma fortificação (aumento de peso) dessa sinapse. Isto é válido para todas as sinapses.
- Soma de todos os pesos dum neurónio constante, ou seja, os pesos serão normalizados. Isto significa que a subida dum peso devida à condição anterior implica uma descida controlada de todos os outros.

Esta *regra de Hebb* é uma regra de aprendizagem não-supervisionada, e tal como já explicado no capítulo das redes neuronais biológicas, consiste simplesmente no incremento da força das sinapses através das actividades simultâneas da saída e entrada. Em termos matemáticos, isto reduz-se à equação (5), ou seja, se a entrada e saída estão activas, então o peso será aumentado em módulo e esse aumento será proporcional ao produto das duas actividades.

$$|\Delta W| = \varepsilon(X \cdot Y) \tag{5}$$

Se a saída ou entrada forem nulas, então pela lei do anulamento do produto não há modificação do peso. Assim, se o peso é positivo, continuará a crescer, e vice-versa. Por outras palavras, a *regra de Hebb* apenas reforça as sinapses, sem lhes mudar o sinal. O objectivo é permitir a existência de sinapses negativas (inibitórias). A constante é uma *constante de aprendizagem* que determina a percentagem de modificação do peso, em relação ao resultado directo da multiplicação. Esta constante é geralmente menor que a unidade (atenuadora).

Para evitar um crescimento indefinido das sinapses, poder-se-ia proceder de duas maneiras distintas:

- O valor do peso satura num máximo.
- A soma dos módulos dos pesos mantém-se constante.

Esta última forma de controlar os pesos é biologicamente plausível, além de ter certas vantagens para o reconhecimento de padrões de estímulos: sempre que um conjunto de pesos sobre, os outros descem. No final ficam apenas alguns pesos activos correspondentes ao padrão de estímulos treinado, enquanto que os restantes ficam a zero (sinapse inexistente).

Esta regra é conseguida através dum circuito multiplicador que multiplica cada entrada pela saída. O problema é a exigência de multiplicação de correntes, além deste circuito ter de ser repetido para cada sinapse. Este circuito está representado na Fig.8.

Um projecto extremamente cuidadoso nesta importantíssima parte da aprendizagem, é indispensável para se obter um comportamento correcto de uma rede neuronal construída com base nestes neurónios. Além da dependência dos restantes factores do projecto, o comportamento global desta rede depende essencialmente da correcta e eficaz concepção dos mecanismos de adaptação (aprendizagem).

XII. CONCLUSÕES E COMENTÁRIOS FINAIS

Este trabalho destinou-se a verificar a implementabilidade de um neurónio 100% analógico em modo de corrente. Esse modo de corrente foi apenas conseguido para as entradas e saídas, bem como para todas as operações internas deste neurónios, exceptuando as entradas dos pesos. Este neurónio destinar-se-ia a áreas de aplicação que exijam o máximo em velocidade de resposta duma rede neuronal. Implementações mais lentas, com menor consumo e mais *standard* poderão ser aplicadas em áreas onde a velocidade não fosse o factor determinante da *performance* dos respectivos sistemas. Aqui poder-se-iam utilizar os transístores na zona de inversão fraca, reduzindo o consumo em várias ordens de grandeza.

Com tecnologias mais modernas em que os menores tamanhos conduzem a capacidades menores, será possível alcançarem-se velocidades ainda maiores, mesmo utilizando os transístores na zona de inversão fraca. Isto pode ter um grande interesse para a implementação de redes de grandes dimensões e baixo consumo. Só assim é que conseguirá estudar o comportamento de redes grandes e utilizá-las em aplicações reais.

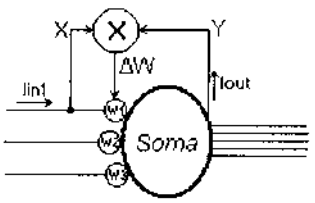


Fig. 8 - Circuito auxiliar de aprendizagem para a primeira sinapse cujo peso é W1. Este circuito multiplica a corrente de saída pela de entrada da respectiva sinapse, fornecendo um sinal de modificação do peso desta sinapse. Este sinal de modificação é proporcional àquela multiplicação, mas pode vir afectado duma constante de atenuação (constante de aprendizagem). Para cada sinapse seria necessário um circuito idêntico. Note-se que a extracção da corrente de saída exige mais um transistor por axónio, devido ao modo de funcionamento em corrente.

REFERÊNCIAS

[1] L.B.Barranco, E.S.Sinencio, A.R.Vásquez, "A CMOS analog adaptive BAM with on-chip learning and weight refreshing", IEEE Transactions on Neural Networks, Vol. 4, N.3, May 1993.

[2] J.Choi, S.H.Bang, B.Sheu, "A programmable analog VLSI neural network processor for communication receivers", IEEE Transactions on Neural Networks, Vol. 4, N.3, May 1993.

[3] J.Alspector, B.Gupta, R.B.Allen, "Performance of a stochastic learning microchip", NIPS - Neural Information Processing Systems vol. 1, Morgan Kaufmann Publ, 1989.

[4] M.Brownlow, L.Tarassenko, A.F.Murray,.A.Hamilton, H.M.Reekie, "Pulse-firing neural chips for hundreds of neurons", NIPS - Neural Information Processing Systems vol. 2, Morgan Kaufmann Publ., 1990.

[5] J.P.A.Carreira, "Circuitos com processamento de corrente para conversão analógico-digital e digital-analógico de sinais de alta frequência"; Tese de Mestrado, Instituto Superior Técnico da Universidade Técnica de Lisboa, 1993.

[6] T.D.Chiueh, R.M.Goodman, "VLSI implementation of a high-capacity neural network associative memory", NIPS - Neural Information Processing Systems vol. 2, Morgan Kaufmann Publ., 1990.

[7] S.P.Deweerth, C.A.Mead, "An analog VLSI model of adaptation in the vestibulo-ocular reflex", NIPS - Neural Information Processing Systems vol. 2, Morgan Kaufmann Publ., 1990.

[8] H.P.Graf, L.D.Jackel, "Analog electronic neural network circuits", IEEE Circuits and Devices, pp. 44-55, July 1989.

[9] P.R.Gray, R.G.Meyer, "Analysis and design of Analog Integrated Circuits", John Wiley & Sons, Inc., third edition, 1993.

[10] S.Grossberg, "Studies of mind and brain - neural principles of learning, perception, development, cognition and motor control", BSPS - Boston Studies in the Philosophy of Science vol. 70, D.Reidel Publishing Company, 1982.

[11] A.Hartstein, R.H.Koch, "A self-learning neural network", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[12] J.R.Mann, S.Gilbert, "An analog self-organizing neural network chip", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[13] C.Mead, "Analog VLSI and Neural Systems", Addison Wesley.

[14] J.L.Meador, C.S.Cole, "A low-power CMOS circuit which emulates temporal electrical properties of neurons", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[15] A.Moopenn, T.Duong, A.P.Thakoor, "Digital-Analog hybrid synapse chips for electronic neural networks", NIPS - Neural Information Processing Systems vol. 2, Morgan Kaufmann Publ, 1990.

[16] P.Mueller, J.V.Spiegel, D.Blackman, T.Chiu, T.Clare, J.Dao, C.Donham, T.Hsieh, M.Loinaz, "A programmable analog neural computer and simulator", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[17] A.F.Murray, A.Hamilton, L.Tarassenko, "Programmable analog pulse-firing", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[18] B.Nabet, R.B.Darling, R.B.Pinter, "Analog implementation of shunting neural networks", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[19] S.Ryckebusch, J.M.Bower, C.A.Mead, "Modeling small oscillating biological networks in analog VLSI", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[20] J.L.Ryckebusch, M.A.Mahowald, C.A.Mead, "Winner-take-all networks of O(N) complexity", NIPS - Neural Information Processing Systems vol.1, Morgan Kaufmann Publ, 1989.

[21] S.Satyanarayana, Y.Tsividis, "A reconfigurable analog VLSI neural network chip", NIPS - Neural Information Processing Systems vol. 2, Morgan Kaufmann Publ, 1990.

From Procedural to Object-Oriented Programming (foundations, distinctions, applications, training, attractive tutorial)

Valery Sklyarov

Resumo- **Este artigo descreve as técnicas fundamentais aplicadas na análise e na programação orientada a objectos (OOD e OOP). O artigo destaca, igualmente, as capacidades da linguagem C++, descreve algumas regras para a escrita destes programas e apresenta uma aplicação de auto-estudo.**

Abstract - **This paper discusses the basic approaches involved in the use Object-Oriented Programming (OOP) and Object-Oriented Design (OOD) for application development. It emphasises the principal distinctions between two widely-used and well-known directions in software development. At present there are many computer languages that incorporate OOP capabilities. One of the most powerful of these languages is C++. This article underlines the main innovation introduced in C++. It recommends some rules for writing C++ object-oriented programs, and considers an animated software tutorial which allows the OOP approach in general, and C++ in particular, to be learned in the fastest possible way.**

I. INTRODUCTION

Programming languages can be considered as a tools for creating different software systems. Each language supports a particular programming technology. The complexity of developing modern software systems is increasing, and there are basic limits in the ability of a particular technology to cope with this complexity. Widely-used approaches in software development are related to procedural and modular programming. The basic idea of these approaches can be represented in the following expression:

PROGRAM = ALGORITHM + DATA

Bjarne Stroustrup defines the ideas being accepted in procedural and modular programming as follows [1]: "Decide which procedures you want; use the best algorithms you can find", "Decide which modules you want; partition the program so that data is hidden in modules". The languages which support procedural and modular programming are especially appropriate for tasks having algorithmic features (mathematical computations, etc.). However, because of the complexity of many tasks, especially having non computation characteristics (computer-aided design, data base, user interfaces, etc.), their effectiveness is limited [2]. This compels us to consider new approaches and new technologies in software development.

The idea of Object-Oriented Programming (OOP) was implemented for the first time in the Simula-67 language. OOP was invented as a tool for dealing with the increasing complexity of software systems. We omit a proof of this assertion and refer to [2].

The basic idea of OOP can be represented in the following expression:

PROGRAM = OBJECTS + MESSAGES

Bjarne Stroustrup defines the idea fundamental in OOP as follows [1]: "Decide which classes you want; provide a full set of operations for each class; make commonality explicit by using inheritance".

II. OOP FOUNDATIONS AND APPLICATIONS

Let us start with foundations of Object-Oriented Technology (OOT). The basic objective of this technology is to introduce new ideas for the design of very complicated software systems. The limitations of the human capacity for dealing with complexity is well known [2]. A good approach to overcoming these limitations is decomposition. OOT directly supports object-oriented decomposition which has many distinctions from algorithmic decomposition. It deals with a set of objects interacting to perform some unique behaviour. An object is a tangible entity encapsulating data to be manipulated and methods which usually provide an external interface.

Consider an example of an object called a register. Suppose that it is a model of real digital electronic scheme. Let us assume, that our register is 10 bits in size, and that we can perform read and write operations on it. This necessitates that we invent an object comprising the following members:

```
state   (a data member);
READ    (a method or function member);
WRITE   (a method or function member).
```

After an object has been described we can suggest a simple scenario which would be something like the following:

```
write a value N to yourself;
```

```
read a value being registered.
```

To perform these simple actions we must send the messages mentioned above to the object. It should be noted that each particular language introduces its own terminology. For example in C++ methods are called member functions, and messages are operations invoking the member functions that provide the external interface, etc.

Designing a complex digital device involves using many kinds of simple registers. Suppose the registers to be considered have different sizes. So on the one hand they are slightly different, but on the other hand they have exactly the same well-defined behaviour. We therefore need to create a class of registers. Basically, classes and objects are the main innovations of OOP. They are closely related each others and cannot be considered independently. The main difference between them is the following: an object is a concrete entity existing in time and space; a class is only an abstraction [2]. Grady Booch defines class as follows [2]: "A class is a set of objects that share a common structure and a common behaviour". Let us consider our class named for instance REGISTER, in a bit more detail. Firstly, in order to define an object of the class REGISTER we must define an instance of the class. When we are defining an object it would be useful to make some initial settings, for example to set a size for a register and an initial state. The setting the size of a register involves memory allocation (we remember, that an object is a model of a particular register in computer memory). Another words we want:

- to allocate memory for setting the size of a particular register;
- to make an initial setting.

OOT supports these operations directly by introducing a special method (a member function) called a constructor.

A constructor is responsible for creating an object and initializing its data members. In our example it creates a particular register having a defined size SIZE, and sets it to a defined state INITIAL_STATE. In C++, the declaration of the class could be following:

```
class REGISTER
{   BOOLEAN *state;
    unsigned size;
public:
    REGISTER(unsigned SIZE,
         BOOLEAN *INITIAL_STATE);
    void WRITE(BOOLEAN *NEW_STATE);
    void READ(BOOLEAN *CURRENT_STATE);
    .........
};
```

As you can see a constructor has the same name as its class, and it returns no value (not even a void). Our constructor can be defined in the following form:

```
REGISTER::REGISTER(unsigned SIZE,
                   BOOLEAN *INITIAL_STATE)
{
    size = SIZE;
    state = new BOOLEAN [SIZE];
    for(int i=0; i<SIZE;state[i++] =
        (INITIAL_STATE)?
        INITIAL_STATE[i]:0);
}
```

Here, BOOLEAN is a user-defined type representing a one dimensional boolean array (each bit of the array corresponds to a related bit of a real register), `new` is a C++ operator (and a C++ keyword) providing dynamic storage allocation. We will omit other explanations of C/C++ instructions and library functions (the reader can find them for example in [3], or in any C/C++ programming guide and library reference).

Let us consider our register in a bit more detail. In most instances it should probably be set to zero (00...0) initially. This leads us to the idea of a constructor which has so called default parameters. Consider the following constructor declaration (inside the class):

```
REGISTER(unsigned SIZE,
         BOOLEAN *INITIAL_STATE = NULL);
```

Now we can define an object in two possible ways (we have assumed that SIZE = 10):

```
REGISTER register1(10);
REGISTER register2(10,INITIAL_STATE);
```

In the first call of the constructor, the second parameter is NULL by default. In the second call the second parameter is a pointer to a particular initial state.

Once an object has been created it occupies computer memory. To free memory that was allocated we must destroy the object. To do that, the OOT provides a special method (member function) called a destructor. A destructor has the same name as the class, but preceded by a tilde (~). As we have already mentioned it destroys the object being constructed. Consider the following possible declaration and definition of a destructor for our example.

```
class REGISTER
{ ........
public:
    REGISTER(...);
    ~REGISTER(void)
    {   delete [size] state;   }
.........
};
```

Here `delete` is a C++ operator (C++ key word) which provides dynamic storage deallocation. Our (and any other) destructor has no arguments.

Let's consider our class named REGISTER further. As we mentioned above it represents a set of objects which

exhibit some well-defined behaviour. A concrete object comprises data members and function members. Data members can be considered as properties of an object and therefore external access to the data members can be strongly restricted. OOT provides member access control which make it possible to hide both data members and function members within class, so that access to them from outside the class is limited. For instance, the C++ language introduces member access control attributes (key words), which are: `public`, `protected` and `private`. All members of a class are `private` by default. `Public` (`protected`, `private`) members must be declared in the public (protected, private) section of a class. The significance of the attributes is: `public` members can be used both inside and outside of a class without any restrictions; `protected` members can be used inside a class, by `friend` functions (see below) and inside derived classes (see below); `private` members can be used inside a class and by `friend` functions (`friend` is a key word of C++). Different sections (public, protected, private) can be repeated any number of times and in any sequence. In the example we considered above, we had one implicitly declared private section (by default) and one explicitly declared public section.

Let us look at an extended version of our example. Suppose we want to introduce a shift register that is a variety of register. Consider the previous REGISTER class declaration. We remember that the WRITE and the READ are member functions of class REGISTER. We need exactly the same functions for a shift register. In addition, our shift register involves a new function performing the shift operation. In other words it is worthwhile to build a hierarchy of classes which might be as follows: base class named REGISTER - derived class named SHIFT_REGISTER. The concept of OOT which emphasizes a hierarchical structure is called inheritance. Basically inheritance denotes a relationship between classes. It makes it possible that one class shares the behaviour and/or structure of one or more other classes. Let us continue with our example and design a base class named REGISTER and a derived class named SHIFT_REGISTER. According to the previous explanation, the class SHIFT_REGISTER will inherit members of the class REGISTER. In addition the new class will introduce some new members, for example, a member function named SHIFT. It leads us to the following declaration:

```
class SHIFT_REGISTER : public REGISTER
// declaring the derived class SHIFT_REGISTER
// we must enumerate
// the base classes in a comma-delimited list
// followed after
// colon (:). In the example there exist a
// single base class
// named REGISTER
     { .......
     public:
          SHIFT_REGISTER(unsigned SIZE,
               BOOLEAN *INITIAL_STATE) :
             REGISTER(SIZE,INITIAL_STATE)    { }
          .......
          void SHIFT(unsigned number);
         .......
};
```

When you declare a derived class you can use the access specifier (`private`, `protected`, or `public`) in front of the class in the base list. Access specifiers allow you to alter the inherited access attributes for members of the derived class (see, for example [4]). If a base class has a constructor defined explicitly with one or more arguments, any derived class must have a constructor as well. In the example a constructor for the class SHIFT_REGISTER is declared as:

```
SHIFT_REGISTER(unsigned SIZE,
    BOOLEAN *INITIAL_STATE) :
        REGISTER(SIZE,INITIAL_STATE)    { }
```

Suppose that the READ function is exactly the same for both (base and derived) classes. However the WRITE function is slightly different for the derived class. The C++ language allows us to perform a redefinition (it is called function overriding) of a function in this case. We want to use a base class version of the function WRITE for our class REGISTER and a derived class version of the function WRITE for our class SHIFT_REGISTER. Let us consider the following definitions:

```
REGISTER reg(...), *pointer1_to_register,
          *pointer2_to_register;
SHIFT_REGISTER shift_register(...);
```

One rule in C++ says that any variable defines as a pointer to a base object (see, for instance, pointer1_to_register, pointer2_to_register) may also be used as a pointer to a derived object. The following statements will be valid in our example:

```
pointer1_to_register = &reg;
pointer2_to_register = &shift_register;
```

Now using pointer2_to_register gives access to all members of the object shift_register, inherited from the class REGISTER. Consider the following expression:

```
pointer2_to_register -> WRITE(...);
```

Which version of the WRITE will be called? If pointer2_to_register was defined as a pointer to the base class, then the WRITE member function of the base class will be called. And how can we invoke the derived class version? The answer lies in a `virtual` function declaration (`virtual` is C++ key word). Virtual functions make it possible to resolve the overloading problem

(polymorphism) during the course of program execution (it is called late binding), rather that at compile time (it is called early binding). They represent functions declared with the specifier `virtual` in a base class and subsequently redefined in one or more derived classes with their names, types of returned values and the number and types of arguments, being unchanged.

Suppose we want to add two extra classes named LEFT_SHIFT_REGISTER and RIGHT_SHIFT_REGISTER to be inherited from class SHIFT_REGISTER. Let us consider the SHIFT member function that is common to both new classes. In our representation the SHIFT_REGISTER class makes sense only as the intermediate base of classes LEFT_SHIFT_REGISTER and RIGHT_SHIFT_REGISTER derived from it. Really, registers that perform a pure shift operation do not exist, because this operation is either a left shift or a right shift. To declare operations like these, C++ provides pure virtual functions. A virtual function is made pure by setting it equal to zero. If a class contains at least one pure virtual function it is called an abstract class. It is impossible to create an object for such a class. It may only be used as a base class for the definition of derived classes which can inherit its members.

So now we could consider the following declarations:

```
class SHIFT_REGISTER : public REGISTER
{  ........
public:
  virtual void SHIFT (unsigned number) = 0;
       ........
};
class LEFT_SHIFT_REGISTER : public
     SHIFT_REGISTER
{ ........
public:
     void SHIFT (unsigned number)
     {   shifting left by number bit
positions     }
    ........
};
class RIGHT_SHIFT_REGISTER : public
     SHIFT_REGISTER
{  ........
public:
     void SHIFT (unsigned number)
     {   shifting right by number bit
positions   }
    ........
};
```

Let`s look at the statements:

```
(register state) <<= number;
(register state) >>= number;
```

Since state is a boolean array we cannot use standard >>= and <<= C/C++ operators. However the C++ language lets you redefine the actions of most standard operators. The compiler distinguishes the various functions by noting the context of the call. In other words it can check the number and types of the operands. The keyword `operator`, followed by the operator symbol, is used to define the function name. Let us assume that we want to override the operator <<=. An example of how you can do this is as follows:

```
class LEFT_SHIFT_REGISTER  :  public
     SHIFT_REGISTER
{ ........
public:
       ........
      BOOLEAN* operator <<= (unsigned number)
       {
          for(unsigned i=0;i<size-number;i++)
             state[i] = state[i+number];
          for(i=size-number;i<size;i++)
             state[i] = 0;
          return state;
        ........
        }
};
```

It should be mentioned that we intend to use the member state which is declared in the base class. To enable this member to be accessed in a derived class, it should be assigned the attribute `protected` (you must avoid the attribute `public` for data members of a class). Now the following member function will work correctly:

```
void SHIFT(unsigned number)
{   *this <<= number;   }
```

Here a new C++ pointer named `this` is invoked (`this` is C++ key word). It is a local variable of the class which does not need to be declared. It is a pointer to the object containing the function being executed and is available in the body of any nonstatic member function. So the return statement:

```
class_type* func(...)
{........
   return this;
  ........ }
```

returns a pointer (for instance &obj) to the object of type class_type, in which the function named func was declared. The return statement:

```
class_type& func(...)
{ ........
   return *this;
  ........  }
```

returns the object (for instance obj) of type class_type, in which function named func was declared. The construction type& lets us create a reference type closely related to a pointer type. In the following statements:

```
LEFT_SHIFT_REGISTER lsr(10);
LEFT_SHIFT_REGISTER &ref_lsr = lsr;
```

the lvalue ref_lsr is an alias for lsr. Any operation on ref_lsr has precisely the same effect as that operation would on lsr.

Consider the expression `*this <<= number` which appeared above. Basically, an operator function must either be a nonstatic member function, or have at least one argument of the class type. Our nonstatic class member operator function has two arguments which are:

- an implicitly defined argument, of type LEFT_SHIFT_REGISTER;
- an explicitly defined argument named number, of type unsigned integer.

The expression `*this` lets refer to an object of a class LEFT_SHIFT_REGISTER vie its implicitly defined pointer this (remember that `this` is a pointer and `*this` is an object).

Now you can use expressions:

```
object_name.SHIFT(number);
pointer_to_object->SHIFT(number);
```

Suppose you want to use another expression which will look something like the following:

```
object_name <<= number;
```

In this case you have to change the operator<<=(...) function as follows:

```
LEFT_SHIFT_REGISTER& operator <<= (unsigned
number)
{ ... (see considered above statements) ...
       return *this;
}
```

At last if you want to consider the following statement:

```
object_name << number;
```

you have to overload the predefined bitwise left shift "<<" operator:

```
void operator << (unsigned number)
{ ... (see considered above statements,
       excepting "return ... ") ...  }
```

Let us consider another possible task. We wish to compare states for two objects which have different types. It could be states of a left shift register and a right shift register. We want to invoke operations like Equal To, Not Equal To, etc. and perform them in a function named comp_two_reg. Remember that state has the attribute protected and therefore is a hidden class member. So how can we access it? The answer lies in the `friend` function declaration (`friend` is C++ key word). Using `friend` declarations, C++ provides a mechanism for access to private (protected) members of a class from functions which are not members of that class. This is enabled when the functions have the specifier `friend`. For the example we are considering, we should declare the function comp_two_reg in LEFT_SHIFT_REGISTER and RIGHT_SHIFT_REGISTER with the specifier `friend`. For the general case, the necessary statements are outlined below.

```
class LEFT_SHIFT_REGISTER : public
     SHIFT_REGISTER
{........
friend void
     comp_two_reg(LEFT_SHIFT_REGISTER *lsr,
     RIGHT_SHIFT_REGISTER *rsr);
     // we assume that our function has
     // returned value of type void
};
class RIGHT_SHIFT_REGISTER : public
     SHIFT_REGISTER
{ ........
friend void
     comp_two_reg(LEFT_SHIFT_REGISTER *lsr,
     RIGHT_SHIFT_REGISTER *rsr);
     ........
};
void comp_two_reg(LEFT_SHIFT_REGISTER
     *lsr, RIGHT_SHIFT_REGISTER *rsr)
     {if((lsr->size)!=(rsr->size))
        cout  << "states can not be compared"
              << endl;
        for(unsigned i=0;i<lsr->size;i++)
           if (lsr->state[i]!=rsr->state[i])
           {   cout << "states are not equal"
                    << endl;
               return;
           }
         cout << "states are equal" << endl;
}
```

Remember that `friend` functions are not member functions and so they do not have the pointer `this`.

Let us build a more complicated model of the shift register. For example we want to obtain information about a fixed state named state_fixed which is being defined independently of a particular object. For these purposes a new storage class specifier, `static`, can be used. Consider the following declaration of the state_fixed LEFT_SHIFT_REGISTER static data member:

```
static BOOLEAN *state_fixed;
```

The member state_fixed is called a `static` member. Static members have different properties from nonstatic members. With nonstatic members, a distinct copy exists for each object of the class. With static members only single copy exists, shared by all objects of the class. It is allowed to initialize static data members outside of a class

(if they are accessible), even a particular object has not been defined yet, for instance:

```
LEFT_SHIFT_REGISTER::state_fixed = NULL;
or
LEFT_SHIFT_REGISTER::state_fixed =
     new BOOLEAN[25];
```

The basic use for static members is to keep track of data that is common to all objects of a class. Because there is only a single copy of a `static` function for many objects of the same class, it does not have the pointer `this`, and therefore can only access nonstatic members by explicitly specifying a concrete object with the . or -> selection operator. For example we can consider an alternative way making it possible to set a fixed state. In this case we are keeping our fixed state in a `static` function named, for instance, S_FIXED. This function would be defined as the following:

```
// the first statement represents the
// function declaration within the class
static void S_FIXED(LEFT_SHIFT_REGISTER *lr);
// the following statements show the possible
// function definition
void LEFT_SHIFT_REGISTER::S_FIXED(
     LEFT_SHIFT_REGISTER *lr)
     {   BOOLEAN fixed_state[] =
          { 1,0,1,1,...,0 };
// accessing to class data members explicitly
// using their names (pointers), for example,
// lr -> size; or lr -> state[i];
}
```

The final step of our example is devoted to memory of type register. Suppose we want to consider arrays of registers, which can be:

an array of left shift registers;
an array of right shift registers.

In other words we intend to build containers of registers. They might represent a logical model of a physical scheme located on a single chip, which is designed for general application. Basically our task is aimed at constructing a family of related classes which provide an array of left shift registers and an array of right shift registers. This can be done within the boundaries of OOT using so called templates (`template` is C++ key word). These allow you to define a pattern for either class definitions, or a family of related functions, by setting the data type itself as a parameter. Borland C++ container classes such as stacks and arrays are good example of using templates [4].

Suppose l_or_r is a type which can be either

LEFT_SHIFT_REGISTER

or

RIGHT_SHIFT_REGISTER.

A declaration "template<class l_or_r>" says that l_or_r is a type name. A class `template` specifiers how individual classes can be constructed. For our example the individual classes might be:

the class of left shift registers;
the class of right shift registers.

As a result our class `template` would be look something like the following:

```
template<class l_or_r> class ARRAY:
     public LEFT_SHIFT_REGISTER,
     public RIGHT_SHIFT_REGISTER
{  l_or_r **data;
   unsigned array_size;
 public:
   ARRAY(unsigned ARRAY_SIZE,
         unsigned SIZE,
         BOOLEAN *INITIAL_STATE = NULL);
   ~ARRAY(void);
   l_or_r& operator [] (unsigned x)
      { return *data[x]; }
};
template<class l_or_r> ARRAY<l_or_r>::
ARRAY(unsigned ARRAY_SIZE,
     unsigned SIZE,
     BOOLEAN *INITIAL_STATE) :

LEFT_SHIFT_REGISTER(SIZE,INITIAL_STATE),

RIGHT_SHIFT_REGISTER(SIZE,INITIAL_STATE)
     {  data = new l_or_r* [ARRAY_SIZE];
        for(int i=0;i<ARRAY_SIZE;i++)
          data[i] = new l_or_r(SIZE);
        array_size = ARRAY_SIZE;
     }
     template<class l_or_r>
     ARRAY<l_or_r>::~ARRAY()
     {  for(int i=0;i<array_size;i++)
          delete data[i];
        delete [] data;          }
```

We can continue a refinement of our example. Another varieties of devices (counters, decoders, etc.) could be investigated. So a new hierarchical structure could be built. In turns our devices might be considered as a building blocks for more complicated digital schemes, such as microprocessors, microcontrollers and microcomputers. They could be further developed in modern computers and computer systems. In any possible level, object models are applicable in a habitual and natural form. Starting from very simple devices, being considered above, we can develop our application to solve, for example, different tasks of logical simulation and digital synthesis (see, for instance [5]).

III. CONCLUSIONS

We have discussed a simple example demonstrating how OOT in general, and the C++ language in particular, can

be used to represent logical models of digital devices. Briefly, the consequences resulting from what we have discussed are the following.

1. The key concepts of OOP are: encapsulation (class declaration, objects definition, protecting class members, defining object access rules); inheritance (single inheritance, multiple inheritance, abstract classes); polymorphism (function overloading, operator overloading, virtual functions, templates).

2. OOP defines techniques that provide for [3]: describing an object structure, or `class` (in C++ classes also can be represented by structures and unions with slightly different rules for accessing members); describing methods or member functions for the manipulation of objects using unique object inheritance principles; protecting object members and defining the rules for accessing objects; passing messages between objects; eliminating or at least reducing global data.

3. OOP directly supports hierarchical ordering which can be considered as one of the most powerful tools for managing complexity. Using inheritance, object-oriented program can be organized as a set of trees or directed acyclic graphs of classes [1]. The physical building block in object-oriented languages is the module, comprising a set of classes instead of subprograms as in procedural languages. In large systems the object model scales up [2]. Clusters of abstractions can be built in layers on top of one another [2].

4. As followed from point 3, an object model is closely related to finite automata models [5]. It is also mentioned in [2, see p. 90]. Let us return to our example. On the one hand a register is an object. On the another hand it can be considered as tiny, independent machine referring to finite automata theory being developed for a long time. Digital schemes, containing digital devices like registers, counters, decoders, etc., can be described as a set of classes using the basic OOP principles mentioned above. However they can be also considered as a network of machines from the perspective of finite automata theory. Moreover it is an approach which can be used to design parameterized matrix digital schemes [5]. At present templates are developing in nearly the same direction.

5. These are a brief description of new C++ basic key words:

"`class`" which is used to represent a general structure of a set of objects (see also "`struct`" and "`union`" key words [3,4]);

"`delete`", "`new`" which are used to dynamically allocate and deallocate memory;

"`friend`" which is used to design non-member functions accessing private and protected class members;

"`operator`" which is used to overload most of the predefined operators in C++;

"`private`", "`protected`", "`public`" which are used to set predefined rules for accessing class members. You can assign member attributes (private, protected, public) when you declare class members, and when you specify the base classes for derived classes;

"`template`" which is used to build parameterized types - in other words to set the type itself as a parameter;

"`this`" which is used to represent a hidden pointer which is unique for each object, and points (addresses) to the object itself in computer memory;

"`virtual`" which is used to support overloading during execution (to provide late binding).

C++ introduces also some additional key words that are: "`catch`", "`throw`", "`try`" (exception handling, see, for example, [1,4]), "`inline`" (to make function as inline function, see, for instance [3]).

Finally, there is a comprehensive, interactive, animated, graphical tutorial program on C++ and object-oriented concepts [3]. You can refer to [3, p 5-7] to understand how to use the accompanying disk containing the tutorial. The teaching program makes learning C++ as easy as possible. There are also many C++ examples on the tutorial disk.

REFERENCES

[1] Bjarne Stroustrup. The C++ programming language. Second Edition, Addison-Wesley Publishing Company, 1994, 691 p.

[2] Grady Booch. Object-Oriented Analysis and Design. Second Edition. The Benjamin/Cummings Publishing Company, Inc.,1994, 589 p.

[3] Valery Sklyarov. The Revolutionary Guide to Turbo C++. Birmingham, WROX, 1992, 352 p.

[4] Borland C++. Programmer's Guide, Borland International, Inc., 1993, 326 p.

[5] V.A.Sklyarov. Matrix LSI Finite Automata Synthesis. Minsk,Science and Technique, 1984, 372 p.

Nota acerca de desmodulação diferencial de sinais CPFSK

A. Gameiro

Resumo - Neste artigo considera-se a desmodulação diferencial de sinais CPFSK com índices de modulação 1/2 (MSK) e 1. Usando as envolventes complexas dos sinais, mostra-se que a presença no sinal modulado de componentes discretas com a fase adequada, pode numa operação de desmodulação diferencial servir como referência local. Tal tem vantagens na desmodulação de sinais afectados de interferência entre símbolos (IES). Em geral a natureza não-linear do desmodulador diferencial causa um acréscimo na IES, enquanto que se o sinal contiver uma componente periódica que possa servir de referência tal não acontece. Esta análise permite-nos compreender a superioridade do índice de modulação 1 relativamente a MSK quando se trata de desmodular por atraso e multiplicação sinais deste tipo fortemente filtrados.

Abstract - This paper deals with differential demodulation of CPFSK signals with modulation index 1/2 (MSK) and 1. Making use of the signal complex envelopes, we show that the existence of discrete lines with adequate phase in the modulated signal, can be used in a differential demodulation operation as a local reference. This has advantages when demodulating signals with intersymbol interference (ISI). In general because of its non-linear nature the differential demodulator increases the output ISI, unless the signal carries a component that can be used as a reference. This analysis allows us to understand the reasons of the superiority of the modulation index 1 relatively to MSK when demodulating through a delay and multiply structure tightly filtered versions of these signals.

1. INTRODUÇÃO

Modulação CPFSK (Continuous phase frequency shift keying) é um tipo de formatação de sinais digitais por portadora modulada, em que a informação vai transportada na frequência do sinal modulado, cuja fase como o nome indica é contínua.

Este tipo de modulação tem sido bastante estudado e usado em sistema de radiocomunicações [1-4], sendo por exemplo usado hoje em dia nos sistemas de comunicação móvel GSM [5], onde o método de modulação GMSK (Gaussian minimum shift keying) é uma versão filtrada de CPFSK com índice de modulação 1/2. Mais recentemente este tipo de modulação foi também estudado com bastante detalhe, e utilizado em protótipos no âmbito de sistemas de comunicação ópticos coerentes (ver [6] para referências).

Entre os dois tipos de aplicação há no entanto diferenças significativas. Enquanto que em radiocomunicações devido aos ritmos de transmissão serem relativamente baixos, o uso de detecção coerente que possibilita um melhor desempenho tem sido quase generalizado, no caso dos sistemas ópticos, os altos débitos para os quais o canal óptico está especialmente vocacionado requerem simplicidade no método de desmodulação, tendo sido usada com bastante frequência detecção diferencial, e os resultados obtidos vieram lançar alguma confusão quanto ao mérito relativo dos vários índices de modulação na sinalização CPFSK. Assim em radiocomunicações com detecção coerente, o sistema está bem estudado e compreendido, e o caso particular de CPFSK com índice de modulação 1/2, que é geralmente designado por MSK (minimum shift keying), é considerado como um método de modulação eficiente quer em banda quer em potência e bastante tolerante a não-linearidades do canal. O estudo dos desmoduladores diferenciais estudados em comunicações ópticas veio relativizar o mérito de MSK. A primeira diferença surge devido à existência de ruído de fase originado pelos lasers do emissor e do receptor que representa um tipo de distorção significativo ao invés do que sucede em radiocomunicações onde a pureza dos osciladores é suficiente para na maioria dos casos considerar este tipo de ruído como desprezável. O ruído de fase de lasers é geralmente modelado por um processo de Wiener, cuja variância cresce com o tempo de modo que índices de modulação que na detecção diferencial requerem atrasos menores são sob esse aspectos preferíveis, o que significa que se pode conseguir melhores resultados usando índices de modulação superiores a 1/2. O mérito de índices de modulação elevados está pois associado a um novo tipo de distorção introduzido pelo canal de transmissão sendo tal um facto perfeitamente compreendido que não causa qualquer ambiguidade com os resultados alcançados em radiocomunicações. O resultado mais surpreendente alcançado pela investigação em comunicações ópticas, é a maior tolerância dos índices de modulação elevados (até 1) a operações de filtragem que gerem interferência entre símbolos (IES), mesmo no caso do ruído de fase ser desprezável. Kazovsky e Jacobsen [7] verificaram que para um limite de 1dB na penalidade em potência causada pela IES se requer uma banda passante de 3.7/T para MSK e apenas 2.2/T para CPFSK com índice de modulação 1 (1-CPFSK), isto embora o primeiro lóbulo do espectro do sinal MSK tenha uma largura de 1.5/T enquanto que o de 1-CPFSK tem largura 3/T. Tais resultados cuja justificação é insatisfatória colidem frontalmente com a noção de ser MSK um método de modulação eficiente em banda.

É objectivo deste artigo esclarecer esta ambiguidade, e justificar de maneira rigorosa a fraca tolerância de CPFSK com baixos índices de modulação a operações de filtragem que gerem IES. O artigo está organizado do seguinte modo. Na secção seguinte é apresentado o modelo da modulação CPFSK. Na secção III apresenta-se o comportamento de sinalização MSK e 1-CPFSK com desmodulação diferencial e clarificam-se as contradições quanto ao mérito dos vários tipos de modulação consoante o tipo de detecção. Finalmente na secção IV são apresentadas as conclusões principais deste trabalho.

I. A SINALIZAÇÃO CPFSK

CPFSK pertence à classe de esquemas de modulação com amplitude constante, sendo a expressão do sinal modulado

$$s(t) = \sqrt{\frac{2E_b}{T}} \cos(2\pi f_0 t + \varphi(t)) \tag{1}$$

A informação está contida na fase da portadora

$$\varphi(t) = 2\pi h \sum_{k=-\infty}^{\infty} a_k q(t - kT) \tag{2}$$

onde $q(t)$, que representa o impulso elmentar da sequência de informação passado por um integrador, é normalizado de modo que $q(\infty) = 1/2$. No caso de modulação CPFSK, $q(t)$ é um impulso rectangular de duração T igual ao inverso do ritmo de transmissão. Nessas condições, a variação de fase em cada intervalo de símbolo é $\pm h\pi$, e o desvio de frequência $\pm h/2T$, no caso de $a_k \in \{-1, 1\}$.

A derivação da densidades espectral de potência de sinais CPFSK já foi extensivamente tratada na literatura, e em particular nos dois casos que vamos estar especialmente interessados, MSK e 1-CPFSK, obtém-se

$$\begin{cases} S_{1-CPFSK} = \dfrac{E_b}{4T}\left(\delta(f - \dfrac{1}{2T}) + \delta(f + \dfrac{1}{2T})\right) + \\ \qquad + E_b \dfrac{4}{\pi^2} \dfrac{\cos^2(\pi f T)}{\left(1 - (2fT)^2\right)^2} \\ S_{MSK} = E_b \dfrac{16}{\pi^2} \dfrac{\cos^2(2\pi f T)}{\left(1 - (4fT)^2\right)^2} \end{cases} \tag{3}$$

que estão representados na Figura 1.

É pois de notar que a parte contínua do espectro tem a mesma forma para ambos os tipos de modulação, com uma expansão na escala das abcissas de um factor de 2 para 1-CPFSK . Este último apresenta ainda uma componente discreta às frequências $f_0 \pm 1/(2T)$ que transporta metade da potência.. Observando os dois espectros da Figura 1, seria intuitivamente de esperar que para a mesma largura de banda no filtro passa-banda a distorção sofrida por um sinal MSK fosse menor que a de um sinal 1-CPFSK. Tal é relativamente fácil de confirmar se usarmos um detector coerente simples (translação de frequências), no entanto para o caso da detecção diferencial tal não sucede como já foi referido anteriormente. A justificação de Kazovsky e

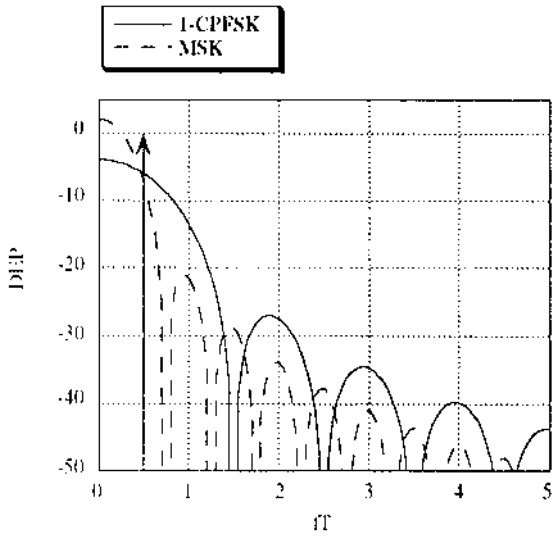


Fig. 1 Densidades espectrais de potência (equivalente banda-base) de 1-CPFSK e MSK.

Jacobsen foi " o sinal 1-CPFSK tem lóbulos inferiores aos de MSK". Uma tal afirmação não é óbvia em face da observação da Figura 1, e além disso carece de rigor. Como o que nos interessa é a distorção em instantes de tempo específicos e não a distorção média ao longo do tempo, a informação contida no espectro não é a mais adequada, e uma análise rigorosa deveria considerar a soma das várias repetições do espectro ("folded-spectrum"), o que daria efectivamente uma fórmula para cálculo da IES no caso de um desmodulador coerente simples. Para o caso do desmodulador diferencial, uma vez que a operação deste desmodulador não se resume a uma simples translação de frequências, a análise é ainda mais complicada e não pode ser reduzida a uma simples comparação de espectros. Na secção seguinte iremos esclarecer este aspecto.

III DETECÇÃO DIFERENCIAL DE SINAIS CPFSK

O diagrama de blocos de um desmodulador diferencial de sinais CPFSK está representado na Figura 2.

Na nossa análise usaremos o equivalente banda-base (envolvente complexa) dos sinais modulados o que permite representar a operação do desmodulador diferencial de acordo com o diagrama de blocos da Figura 3 [8] onde $g(t)$ é o equivalente banda-base do filtro passa-banda. Assim em notação complexa

$$s(t) = \sqrt{\frac{2E_b}{T}} \mathrm{Re}[e^{j(2\pi f_0 t + \varphi(t))}] \tag{4}$$

e logo a envolvente complexa $v(t)$ vem dada por

$$v(t) = \sqrt{\frac{2E_b}{T}} e^{j\varphi(t)} \tag{5}$$

Para o caso de sinalização MSK, é possível demonstrar que se tem

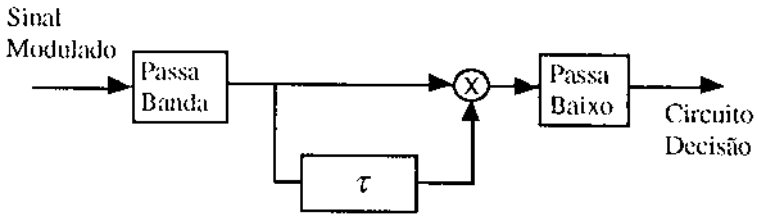


Fig. 2 Diagrama de blocos de um desmodulador diferencial

$$v(t) = \sqrt{\frac{E_b}{T}} \sum_{k=-\infty}^{\infty} \gamma_k h(t - kT) \tag{6}$$

onde E_b é a energia do sinal por bit, T a duração de um intervalo de símbolo, $h(t)$ o impulso elementar dado por

$$h(t) = \begin{cases} \cos(\frac{\pi t}{2T}) & -T \le |t| \le T \\ 0 & \text{outros } t \end{cases} \tag{7}$$

e $\{\gamma_k\}$ é uma sequência de símbolos complexos relacionados com a sequência de informação $\{a_k\}$, através de

$$\gamma_k = j a_k \gamma_{k-1} \tag{8}$$

Quando $a_k \in \{-1, 1\}$, γ_k toma alternativamente valores reais e imaginários, i.e. $\gamma_{2k} \in \{-1, 1\}$ e $\gamma_{2k+1} \in \{-j, j\}$, supondo que γ_0 é real.

Para o sinal 1-CPFSK, obtém-se

$$v(t) = \sqrt{\frac{2E_b}{T}} \left(-\sin(\frac{\pi t}{T}) + j \sum_{k=-\infty}^{\infty} (-1)^k a_k h_1(t - kT) \right) \tag{9}$$

onde

$$h_1(t) = \begin{cases} \cos(\frac{\pi t}{T}) & |t| < T/2 \\ 0 & \text{outros } t \end{cases} \tag{10}$$

Vamos começar por considerar que o filtro passa-banda não distorce o sinal, e que o filtro passa-baixo apenas elimina as componentes à frequência dupla. Nessa situação com sinalização MSK, obtemos para $\tau = T$ e com f_o e T relacionados de forma a que $e^{j2\pi f_o T} = -j$

$$y_k = \mathrm{Im}[v(kT)v^*(kT + T)] \tag{11}$$

Sendo o impulso elementar $h(t)$ um impulso livre de IES, obtemos

$$y_k = \mathrm{Im}[\gamma_k \gamma_{k-1}^*] = a_k \tag{12}$$

Se usarmos 1-CPFSK, o atraso deve ser reduzido para $T/2$ e temos para $e^{j2\pi f_o T/2} = -j$

$$\begin{aligned} y_k = {} & \mathrm{Im}[v(kT)]\,\mathrm{Re}[v(kT - T/2)] \\ & - \mathrm{Re}[v(kT)]\,\mathrm{Im}[v(kT - T/2)] \end{aligned} \tag{13}$$

De (7), sabemos que para 1-CPFSK $\mathrm{Re}[v(t)]$ é uma componente periódica $(-\sin(\pi t/T))$ enquanto que $\mathrm{Im}[v(t)]$, representa o sinal de informação. Temos pois que a saída do desmodulador diferencial é o produto de uma onda sinusoidal por um sinal de informação, ou por outras palavras que as componentes discretas às frequências $f_o \pm \frac{1}{2T}$, funcionam como referência na desmodulação diferencial de um sinal 1-CPFSK. Tal significa que para sinalização 1-CPFSK o desmodulador diferencial é equivalente (esta equivalência já não é válida se incluirmos ruído) ao diagrama de blocos da Figura 4. É esta característica do sinal 1-CPFSK, de transportar uma componente periódica que vai servir como referência na desmodulação diferencial que lhe confere maior imunidade a IES entre símbolos eventualmente causada por operações de filtragem banda-base. Embora o cálculo exacto da IES para uma operação de filtragem passa-banda arbitrária não seja excessivamente difícil recorrendo à formulação usada neste artigo, neste momento apenas estamos interessados na análise qualitativa do problema e não vamos efectuar cálculos precisos.

Assim considere-se para o caso de sinalização MSK, que uma operação de filtragem (filtro $g(t)$ na Fig. 4) dá origem ao sinal desejado S afectado por IES D. Num dos ramos do desmodulador vamos ter $S_1 + D_1$, e no outro

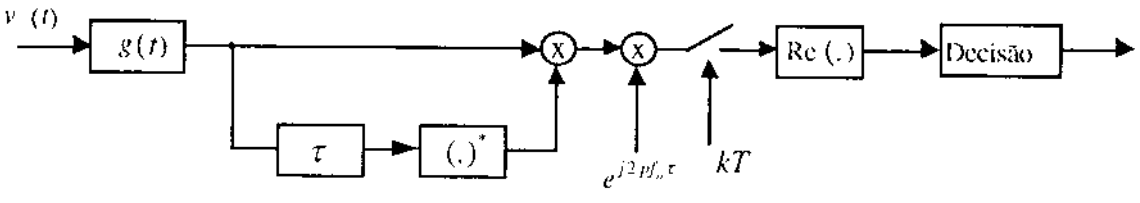


Fig. 3 Diagrama de blocos do equivalente banda-base do desmodulador diferencial.

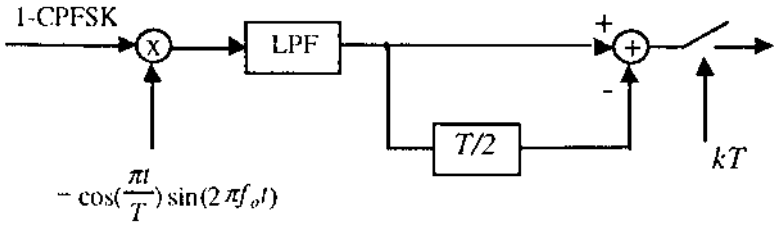


Fig. 4 Representação equivalente do desmodulador diferencial para o sinal 1-CPFSK.

$S_2 + D_2$. O sinal de saída virá então dado por

$$Y = \mathrm{Im}[S_1 S_2^*] + \mathrm{Im}[S_1 D_2^*] + \mathrm{Im}[S_2 D_1^*] + \mathrm{Im}[D_1 D_2^*] \quad (14)$$

No caso de sinalização 1-CPFSK se tivermos à saída do passa-banda $C + S' + D'$, onde C representa a componente periódica, obtemos na saída

$$Y' = C_{90}(S' + D') \quad (15)$$

onde C_{90} representa a componente periódica à saída do passa-banda com um desfasamento de 90º.

Para MSK as potências médias de S_1 e S_2 e as variâncias de D_1 e D_2,são iguais pelo que é imediato concluir que para relações sinal/IES ignais à saída do filtro passa-banda, i.e. $S/D = S'/D'$, o sinal MSK apresenta à saída do desmodulador diferencial uma maior distorção devida a IES causada por filtragem passa-banda. A saída vai estar afectada pela IES de cada um dos ramos e ainda por um termo que representa a interacção entre elas. Uma medida não exacta mas também não excessivamente grosseira é de que esse excesso de distorção é de aproximadamente 3dB. É de notar ainda que filtros passa-banda idênticos provocam em geral níveis de IES diferentes em MSK ou 1-CPFSK, e consequentemente na decisão a distorção induzida pela IES é devida ao efeito conjugado de dois factores. Primeiro há a operação de filtragem passa-banda que introduz IES cujo nível depende da resposta do equivalente banda-base desse filtro à envolvente complexa do sinal ($g(t) * h(t)$ ou $g(t) * h_1(t)$). Segundo há a operação do desmodulador diferencial que por se tratar de uma operação não linear (na representação em banda-base) irá na maioria dos casos provocar um aumento da distorção devida a IES, a menos que o sinal tenha uma estrutura tal que permita explorar a operação de desmodulação diferencial sem provocar acréscimo de IES. Tal situação acontece com sinalização 1-CPFSK, onde o termo que não se anula é o produto do sinal periódico de um dos ramos pelo sinal de informação do outro ramo.

Finalmente uma referência quanto a geração de IES pela operação de filtragem passa-banda. Não é possível generalizar e dizer que filtragem de banda limitada irá afectar mais um ou outro tipo de sinal. O resultado depende do tipo de filtro e largura de banda. Assim uma operação de filtragem rectangular provoca para a maioria das larguras de banda maior IES no sinal MSK que no sinal 1-CPFSK, embora haja algumas larguras de banda ($<1/T$) para as quais o inverso se verifica. Consequentemente a causa fundamental da superioridade de 1-CPFSK filtrado com desmodulação diferencial deve-se fundamentalmente à operação do detector diferencial ser mais adaptada a este tipo de sinal. Usando a metodolgia seguida neste trabalho, seria no entanto relativamente simples obter resultados numéricos quanto às estatísticas da IES para ambos os casos de sinalização.

IV CONCLUSÕES

Neste artigo tratámos da desmodulação diferencial de sinais CPFSK com índices de modulação de 1/2 e 1. A motivação para este trabalho, foi o facto de resultados recentes baseados em métodos semi-analíticos terem apontado para a superioridade do índice de modulação 1 quando se efectua filtragem de banda limitada seguida de uma desmodulação diferencial. Tal resultado contradiz numa primeira observação a convicção corrente de MSK ser um método de modulação eficiente em banda. Neste trabalho recorrendo às expressões da envolvente complexa dos dois tipos de sinais, explicou-se que tal sucede devido ao facto de o sinal 1-CPFSK conter uma componente periódica. Essa componente periódica vai na operação de desmodulação diferencial funcionar como referência local, o que significa que não é introduzida nenhuma IES adicional, induzida pela operação de atraso e multiplicação ao contrário do que sucede com a sinalização MSK.

A metodologia seguida neste artigo pode ser usada pra se obterem resultados analíticos para as estatísticas da IES no instante de decisão possibilitando pois um tratamento analítico completo da operação de desmodulação diferencial para os dois tipos de sinais.

Além disso a interpretação obtida neste trabalho para a operação do desmodulador diferencial, para além da explicação deste caso específico pode ter interesse no projecto de técnicas de modulação especialmente vocacionadas para desmodulação diferencial.

REFERÊNCIAS

[1] C.E. Sundberg, "Continuous Phase Modulation", IEEE Commun. Mag., April 1986, pp.25-38

[2] R. De Buda, "Coherent demodulation of frequency shift keying with low deviation ratio", IEEE Trans. Com., , June 1972, pp.429-435.

[3] K. Hirade et al, "Error-rate performance of digital FM with differential detection in land mobile radio channels", IEEE Trans. Veh. Tech.June 1979, pp.204-212.

[4]J. H. Lodge, M. L. Moher and S. N. Crozier, "A comparison of data modulation techniques for land mobile satellite channels", IEEE Trans. Veh. Tech, Feb. 1987, pp. 28-37.

[5] D. J. Goodman, "Trends in cellular and cordless communications", IEEE Com. Mag., June 1991, pp.31-40.

[6] G. Jacobsen, Noise in Digital Optical Transmission Systems, Artech House 1994.

[7] L. G. Kazovsky and G. Jacobsen, "Multichannel CPFSK coherent optical communications systems", IEEE J. Light Tech., June 1989, pp. 972-982.
[8] J. C. Bic, D. Duponteil and J. C. Imbeaux, Elements of Digitall Communications, Willey 1991.

Design and Characterization of a 20 Gbit/s Clock Recovery Circuit

P. Monteiro, J. N. Matos, A. Gameiro, P. A. Matos, J. F. da Rocha

Resumo- **Neste artigo é descrito o projecto e a implementação de uma unidade de recuperação de relógio, para um sistema de transmissão por fibra óptica a 20 Gbit/s. Esta unidade está inserida no demonstrador do projecto "TRAVEL", RACE 2011. A unidade de recuperação de relógio tem uma estrutura em malha aberta baseando-se num filtro de elevado Q com ressonador dieléctrico.**

Apresenta-se a caracterização eléctrica desta unidade e a sua sensibilidade à dessintonia. Os resultados experimentais mostram que este tipo unidade é um solução bastante atractiva para os sistemas ópticos a muito alto ritmo de transmissão englobados na futura norma STM-128.

Abstract- **In this communication we report the design of a clock recovery circuit produced for the 20 Gbit/s demonstrator of the RACE 2011 project "TRAVEL" of the European Community. The clock recovery circuit is based on an open loop structure using a dielectric resonator narrow bandpass filter with high Q.**

A detailed electrical characterization of the circuit and also its sensitivity to temperature and detuning variations are presented. The experimental results show that the circuit is a very attractive solution for the forthcoming STM-128 optical links.

I. INTRODUCTION

For very high transmission rates, the clock recovery circuit (CRC) is the one of the most critical receiver units. It is essential for any retimed receiver to have a high quality CRC to provide from the received data signal, a very precise local clock in order to perform accurately the regeneration and demultiplexing operations.

Clock recovery circuits can be classified into two groups: open-loop structures and closed loop or adaptive structures. The former is usually built using a nonlinearity which is necessary whenever the incoming signal lacks a discrete spectral line at the transmission rate and a narrowband filter to take out this spectral line from the remain spectrum. The latter approach makes use of the phase-lock principle and it may simply consist of a nonlinearity followed by a PLL or it can take more complex forms such as maximum-likelihood trackers [1] or early late gate bit synchronizers [2]. From a conceptual point of view the adaptive approach is preferable since the phase-lock principle has inherent AFC capabilities and thus one can offer low bandwidths without loosing tuning capabilities. Also this approach offers the potential of complete integration. However for the highest bit rates pursued at the moment the technology needed for a good design is still not completely mature and the open loop solution based on a high Q filter is preferable.

In this communication we report the design and characterization of a clock recovery circuit based on this second solution incorporated into 20 Gbit/s demonstrator of the RACE 2011 project "TRAVEL" ("(HD) TV Transport on Very High Bitrate Optical Links"). Table 1 summarises the required electrical specifications.

TABLE I
ELECTRICAL SPECIFICATIONS OF THE CLOCK RECOVERY CIRCUIT

Operating Frequency	F1=19.90656 Gbit/s (2*STM-64)
Data Input	250 mVpp/side, 50Ω to ground
Clock Output	250 mVpp/side, 50Ω to ground
Dynamic Jitter clock output	< 0.025 UI (RMS)
Static Jitter clock output	< 0.1 UI
Operating Temperature	+10 to 50 (°C)

II. CIRCUIT DESIGN

The implemented CRC based on open-loop structure is illustrated by the block diagram depict in Fig. 1. The circuit includes a pre-amplifier, a pre-filter, a nonlinear circuit (NLC), a dielectric resonator (DR) filter and a narrow bandwidth amplifier.

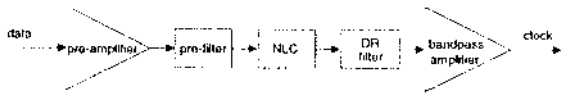


Fig. 1 Simplified block diagram of a clock recovery circuit based on open loop structure.

The pre-amplifier has the main purpose of providing a suitable level to drive the non-linear circuit. The pre-filter is intended to shape the signal so that the jitter caused by the pattern fluctuations is reduced [3]. This filter consists of a microstrip coupled line section centred at 15 GHz. The overall transfer function of the pre-amplifier more pre-filter when combined with input data signal of raised-cosine formatting with approximately 50% roll-off gives a

significant reduction of the jitter caused by the data fluctuations.

The non-linearity (NL) was built using a GaAs FET device biased in the subthreshold region in such a way that half the signal excursion fall below the pinchoff voltage, providing a nonlinearity close to a truncated square law device. For the bit rate under consideration the solution of using an unbalanced NL was chosen instead of the balanced NLs. The unbalanced NLs are simpler to implement and robust enough to be used in a wide class of practical systems at multigigabit regime [4]. It was demonstrated from simulation study [5] that for raised cosine elementary pulses with moderate to high roll-off factors the degradation in performance by using this type of truncated NL is negligible compared with the more common absolute value and full squarer NLs.

The performance of the CRC is highly dependent on the narrow band filter efficiency in filtering the clock line from the continuos spectrum. This filter must have a very high quality factor, a precise center frequency with a low temperature drift. Fig. 2 shows the range of frequencies where different technologies used to build the narrow band filter.

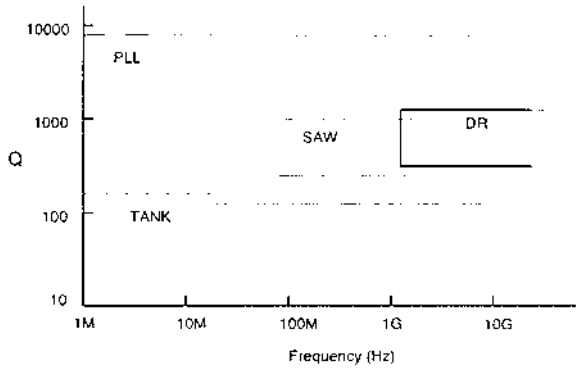


Fig. 2 Different technologies used to build the narrow band filter as a function of the clock rate

At low bit rates the narrowband filter is usually a tuned LC network. Up to a couple of Gbit/s, filters that exploit the interference of acoustic surface waves (SAW) have been used [6]. However for bit rates above a couple of Gbit/s, SAW filters become very difficult to manufacture, because the required spacing between electrodes is very small. For these frequencies the most useful device to build the passive filter is the DR filter. Considerable progress has been done in the last years towards providing DR with good stability in temperature and losses considerably inferior to the ones obtained with SAW devices. These features make the DRs a good choice for the bit rates in the range 5-20 Gbit/s [7-9]. Furthermore due to the high dielectric constant, the devices are available for these frequencies in small sizes which enables circuit miniaturisation. Table 2 summarises the electrical and physical characteristics of the commercial DRs suitable to implement the high Q filter for 20 Gbit/s CRCs [10-11].

TABLE 2
DRS CHARACTERISTICS APPLIED IN FILTER PROTOTYPES

Manufacturer	Siemens	Murata
Reference	B69-500-C2008	DRD033EC015
Dielectric Constant (ε_r)	29	24.5
Q_u at 20 GHz (=1/tangδ)	4200	10000
Shape	cylindrical	cylindrical
Diameter (mm)	3.1	3.33
Thickness (mm)	1.2	1.48
Temp. Coefficient (ppm/°C)	0	0
Available Temp. Coefficients (ppm/°C)	-3; 0; 3; 6; 9; 12	0; 2; 4; 6

The Q of the DR filter represents a trade-off between two factors: the allowable dynamic jitter and the sensitivity to detuning. In order to minimise the dynamic jitter the Q value should be the highest possible. However due to ageing and temperature effects the filter may undergo some detuning. The detuning affects the performance in two ways:

- It will cause a static phase offset of the recovered clock which shifts the average sampling point from its optimum position and causes a performance degradation.
- It will give rise to an increase of the dynamic jitter. With the DR detuned the amplitude of the filtered discrete spectral line is reduced thus enhancing the relative power of the noise sidebands. The increase in jitter is still more pronounced than the inverse of the discrete line reduction for the following reason: the disturbance at the input of the filter is a nonstationary process and its in-phase component is stronger than the quadrature component [12]. If the DR is perfectly tuned only the quadrature component causes phase jitter, the in-phase component causes only amplitude fluctuations. With detuning, part of the AM noise will be converted into FM noise thus increasing the jitter. For robustness against detuning, the value of Q should be low.

Assuming a maximum detuning of ±0.5/1000 and taking the targets of 0.025UI rms for the dynamic jitter and 0.1 UI maximum static offset, it was found following an approaching similar to Gameiro [13] that the optimum Q for the DR should be near 700.

Several filter versions have been tested and implemented using DRs whose characteristics are illustrated in Table 2. The DRs were polished to decrease slightly the thickness with the intend that the resonant frequency was in the range -10% to -5% of resonant frequency. The final frequency tuning was accomplished by a metallic tuning screw. Using these procedures the DRs can be tuned for the desired resonant frequencies without significant degradation of the Q-factor and also good temperature stability.

Table 3 summarises the experimental maximum variation range of the insertion loss and phase slope for a Q_L-factor in range 600 to 800.

TABLE 3
ELECTRICAL CHARACTERISTICS OF DR-FILTERS FOR A Q-FACTOR IN RANGE OF 600 TO 800

Q-factor	600 to 800
Phase slope	3°/MHz to 4°/MHz
Insertion loss at center frequency	<4.5 dB (with conectores)

The experimental magnitude and phase responses of the DR filter are illustrated in Fig. 3. The insertion loss, including the SMA connectors, is 2.7 dB, the Q value is approximately 750 and the phase slope is 4.4°/MHz.

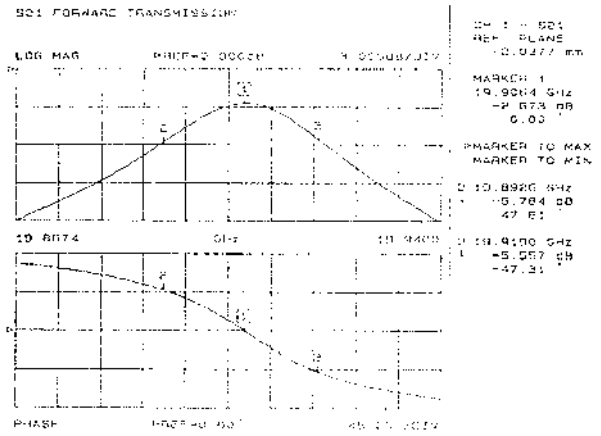


Fig. 3 - The experimental magnitude and phase responses of the dielectric resonator filter

Fig. 4 shows the resonance frequency drift of the filter when a DR with a temperature coefficient of 0ppm/°C was used. From the experimental data the temperature coefficient of DR filter is -4ppm/°C (-80KHz/°C). This temperature dependence can be compensated by using a DR with a symmetric temperature coefficient. Such DRs are commercially available, for example from Murata DRD033EE015 [10]. However this has not been done since for the whole temperature operation (+10 to 50°C) the static jitter at the clock output is less than the project specifications shown in Table 1 as will be illustrated by the experimental results presented in next section.

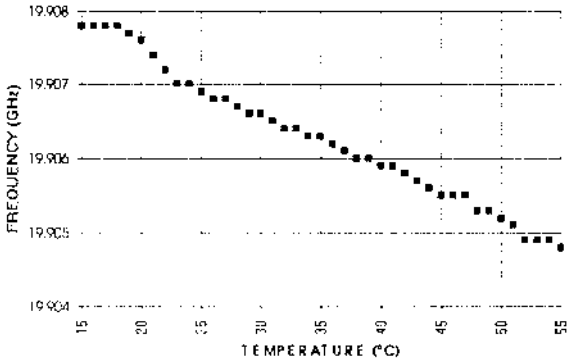


Fig. 4 - Temperature behaviour of DR filter

Figure 5 shows the produced 20 Gbit/s DR filter prototypes with and without the cover case. The physical sizes of these units are 19.5×35.1×18 mm (L×W×H) and use SMA connectors.

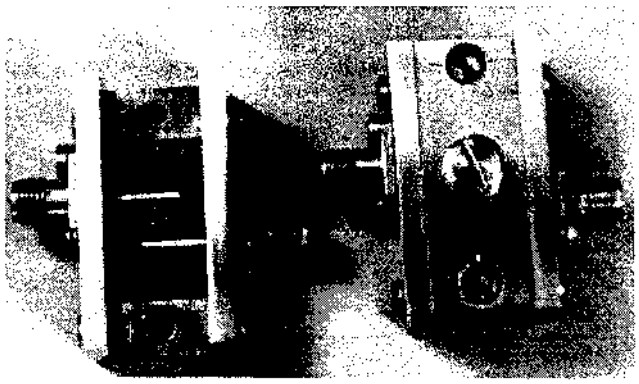


Fig. 5 - 20 Gbit/s DR filter prototypes with and without the cover case.

The clock amplification stage has three main purposes: To provide the required clock signal levels, to reduce the magnitude of the DR filter spurious modes and also to isolate the clock recovery circuit system from the subsequent units. This amplifier stage consists of two caseaded Ka-band pseudomorphic HJ FET chip transistors (NEC-NE32400) assembled on a hybrid form. Microstrip coupled lines instead of chip capacitors were used for DC decoupling. Using this approach a symmetric narrow bandwidth with a high roll-off attenuation out-of-band were achieved as illustrated from the measured S parameters depict in Fig. 6. The pass-band gain is approximately 12.7 dB.

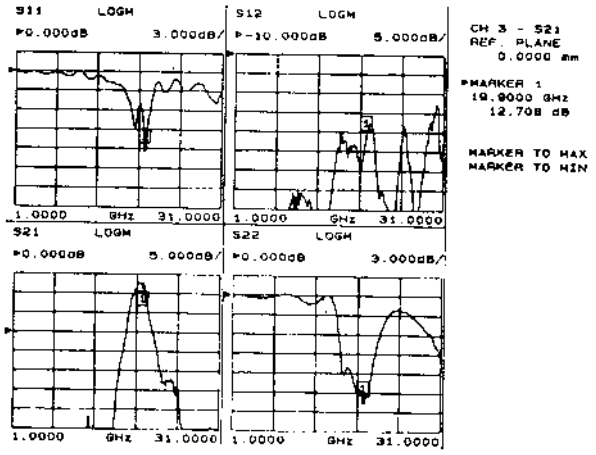
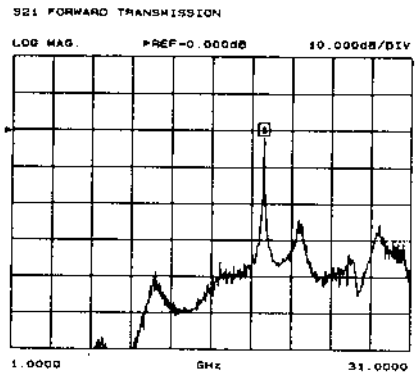


Fig. 6 - Frequency characterization of clock amplification stage

The advantages of using bandwidth amplifier as active filter is better illustrated in Fig. 7 showing the frequency responses of the DR filter and DR filter + Amplifier. From these figures it can be concluded that the out-of-band insertion loss was significantly increased by using this amplifier stage.

a)



b)

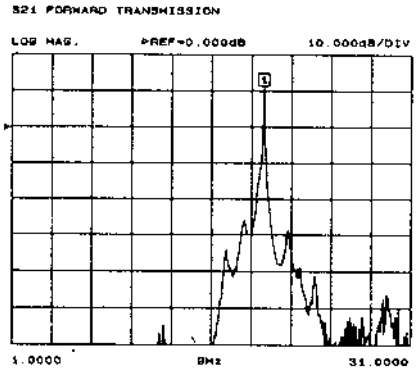


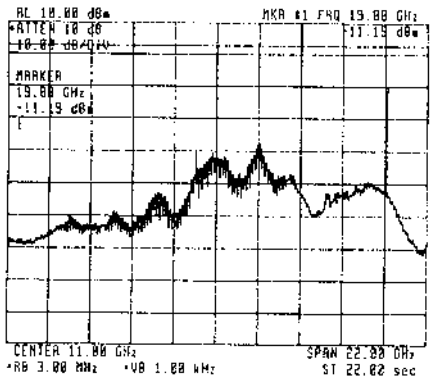
Fig. 7- Frequency response of DR filter without bandpass amplifier (a) and with bandpass amplifier (b).

III. EXPERIMENTAL RESULTS OF OVERALL CIRCUIT

Due to the lack of commercial data pattern generators for the bit rates under consideration, the 20 Gbit/s signal was generated using a 10 Gbit/s pseudo-random bit sequence (PRBS), which was split into two sequences and one of them delayed. After that the two sequences were multiplexed giving the 20 Gbit/s signal. Using this setup, we obtained the spectral and time domain measurements of the clock recovery circuit, for two multiplexed 2^{23}-1 NRZ pseudo-random bit sequences (PRBS), illustrated in Figs. 8 and 9.

Fig. 8a shows the signal spectrum at the nonlinearity output, clearly showing a well-defined discrete line at the bit rate, while Fig. 8b illustrates the spectrum at the CRC output showing a recovered clock line with a power of approximately -2 dBm on 50Ω.

a)



b)

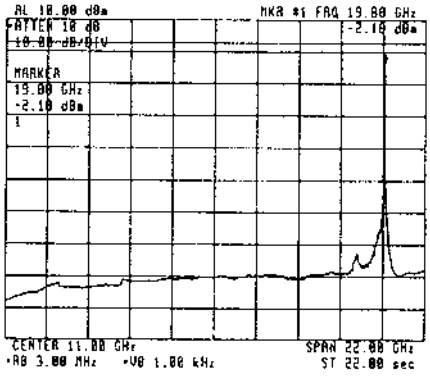


Fig. 8 - Signal spectra at nonlinearity output (a) and at clock recovery unit output (b)

Fig. 9 shows the eye diagram of the input waveform and the recovered clock with an amplitude of approximately 500 mVpp.

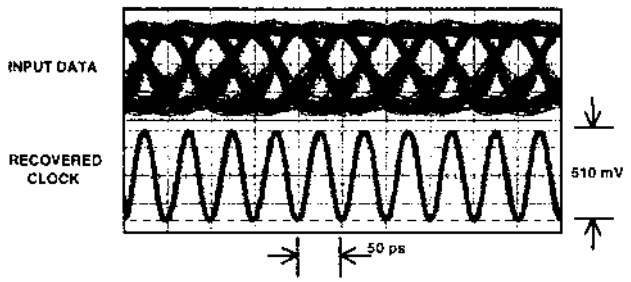


Fig. 9- Time domain waveforms

The jitter performance of the CRC for the PRBS of lengths 2^7-1 and 2^{23}-1 is shown in Figs. 10 where the measured RMS dynamic jitter is plotted for various values of the filter detuning. For a PRBS of length 2^{23}-1 it can be seen that specifications for a maximum dynamic jitter of 0.025 UI allows a detuning larger than ±10MHz. Clock phase deviation versus filter detuning is illustrated in Fig. 11. For a maximum phase deviation (static jitter) of 0.1 UI the CRU allows a detuning of ±6MHz. As refereed in previous section, the measured temperature sensitivity of the DR filter was -4 ppm/ºC (-80KHz/ºC) and consequently the allowable detuning range of ± 6 MHz accommodates a temperature span higher than the operating temperature specifications of 10ºC to 50ºC.

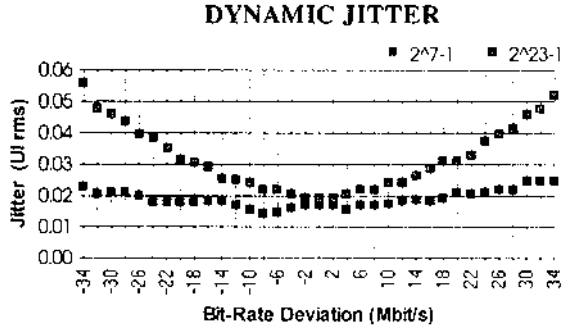


Fig. 10 - Rms dynamic jitter versus bit-rate deviation for the PRBS of 2^7-1 and 2^{23}-1

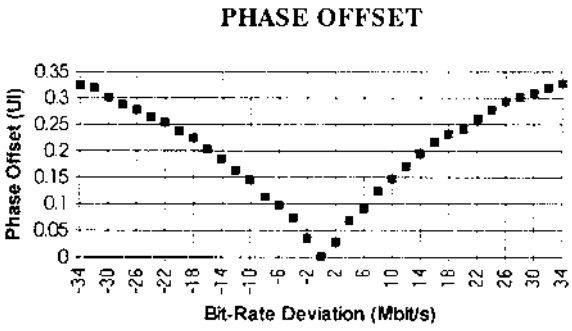


Fig. 11 - Clock phase deviation as a function of bit-rate deviation (filter detuning).

Fig. 12 illustrates the clock amplitude versus filter detuning. For an allowable detuning of ± 6 MHz the clock amplitude varies between 480 mVpp and 510 mVpp that represents only 6% of variation.

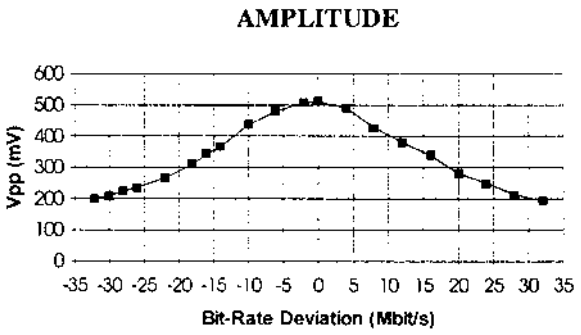


Fig. 12- Clock amplitude versus bit-rate deviation (filter detuning).

IV. CONCLUSIONS

A clock recovery circuit for an optical fiber communication system at 20 Gbit/s was build and characterized. This unit uses a dielectric resonator to implement a high Q filter, a pre-filter to reduce data pattern jitter and also a bandpass amplifier with sharp cut-off to reduce the DR spurious resonances and noise. Particular emphasis has been put on the design of the circuit to guarantee easy reproducibility, manufacturing and good performance over a wide range of operating conditions to make the circuit suitable for practical systems.

As a main conclusion, it turns out that the approach followed provides an economic and robust solution for practical implementation of the very high speed bit rate receivers for the forthcoming SDH hierarchies.

V. ACKNOWLEDGEMENTS

The authors would like thank to the Departamento de Senales, Sistemas y Radiocomunicaciones of E.T.S.I. de Telecomunicacion, UP-Madrid for providing the assembler facilities and to the Dep. ZFZ/NO of SEL-ALCATEL Research Center, Stuttgart, for the measurement facilities. This work has been supported in part by the European Community through the RACE 2011 project.

VI. REFERENCES

[1] L. E. Franks,"Carrier and bit synchronization in data communications- A tutorial review", *IEEE Trans. Com.*, pp. 1107-1121, Aug. 1980.

[2] M. K. Simon, "Nonlinear analysis of an absolute value type of an early-late-gate bit synchronizer", *IEEE Trans. Commun.* Tech., vol.18, pp.686-690, Oct. 1970.

[3] L. E. Franks and J. P. Bubrouski, "Statistical properties of timing jitter in a PAM timing recovery scheme", *IEEE Trans. Commun.*, Vol. 22, pp. 913-920, July 1974.

[4] J. N. Matos; P. Monteiro; A. Gameiro; J.R.F. da Rocha; "Bit Synchronisation in Multigigabit Receivers", *SPIE Proceedings*, Vol. 1974, pp.148-159, Berlim 1993.

[5] J. N. Matos; A. Gameiro; P. Monteiro; J.R.F. da Rocha; "Dielectric Resonator Based Synchronizer for Multigigabit Optical Communications", *Proceedings of 23rd European Microwave Conference*, pp. 941-943, Madrid 1993.

[6] R. L. Rosenberg, C. Chamzas and D. A. Fishman, "Timing recovery with SAW transversal filters in the regenerators of undersea long haul fiber transmission", *J. Lightwave Technol.*, Vol. 2, pp. 917-925, Dec.1984.

[7] K. Hagimoto and K. Aida, "Multigigabit-per-Second Optical Baseband Transmission System", *J. Lightwave Technol.*, Vol. 6, No. 11, November 1988, pp. 1678-1685.

[8] P. Monteiro; J.N. Matos; A. Gameiro; J.R.F. da Rocha; "10 Gbit/s Timing Recovery Circuit Using Dielectric Resonator and Active Bandpass Filters", *Electronics Letters*, Vol. 28, Nº9, pp. 819-820, Abril 1992.

[9] P. Monteiro; J.N. Matos; A. Gameiro; J.R.F. da Rocha; "20 Gbit/s DR Based Timing Recovery Circuit", *Electronics Letters*, Maio 1994, Vol. 30, No 10, pp. 799-800.

[10] Microwave Ceramic Components, Resomics, Murata, cat nº095E-3, 1990

[11] Microwave Ceramics, Siemens, Edition 1990/91

[12] F. M. Gardner, "Self-noise in synchronizers", *IEEE Trans. Com.*, pp.1159-1163, Aug. 1980.

[13] A. M. S. Gameiro and da Rocha J. R. F., "The performance of the digital delay and multiply bit synchronizer in optical communications" *3rd Bangor International Symposium in Communications*, pp. 248-253, Bangor 1991.

Laguerre Filters – An Introduction

Tomás Oliveira e Silva

Abstract - **In this tutorial paper we present a generalization of the transversal filter, called Laguerre filter, and study some of its more remarkable properties. This filter is obtained by replacing each delay of the transversal filter by a first order all-pass section, and by applying a first order low-pass filter (with the same pole used in the all-pass sections) to the filter's input signal. Both the transversal and the lattice forms of the Laguerre filter are discussed. We also deduce the stationarity conditions of the mean-square error of a Laguerre filter (transversal or lattice) with respect to its pole position.**

Resumo - **Neste trabalho apresentamos uma generalização dos filtros transversais, os chamados filtros de Laguerre, e estudamos algumas das suas propriedades mais notáveis. Estes filtros são obtidos substituindo cada atraso dos filtros transversais por uma secção passa tudo de primeira ordem e preprocessando o sinal de entrada do filtro com um filtro passa baixo de primeira ordem (com o mesmo pólo das secções passa tudo). São estudadas as formas transversal e "lattice" do filtro de Laguerre. Deduzimos também as condições de estacionaridade do erro quadrático médio de um filtro de Laguerre (forma transversal ou "lattice") em relação à posição do seu pólo.**

I. INTRODUCTION

The transversal filter and some other filter structures related with it, such as the lattice filter, are very popular among the models of linear systems, specially if adaptation of its parameters is desired [1]–[3]. Some applications where these filters have attained considerable success include, among others, system identification, linear prediction, channel equalization, and echo cancellation. The reason for this success is, besides the simplicity of the transversal filter structure, the unimodality of its error surface, and the existence of fast and efficient adaptive algorithms to adjust its parameters [1]–[4].

The principal problem of the transversal filter, which is also related to its advantages, is that its impulse response has a finite duration (it is a FIR filter). For this reason, when this filter is used to approximate a system with a long (possibly infinite) impulse response the minimum number of delays of the filter required to provide an acceptable approximation can be quite high. This problem can be partially solved using filters with an infinite impulse response (IIR filters). However, these filters have their own problems, specially if output error models are used [5], [6]. Among these are possible multimodal error surfaces [7], and possible instability problems related to the adaptation of the poles of these filters [6].

Another disadvantage of the transversal filter is that its continuous-time (analog) version requires delay lines, which are difficult to implement. In an attempt to solve this problem, in [8] each delay of the transversal filter was replaced by an all-pass filter. This preserves many of the properties of transversal filters and gives rise to continuous-time (and discrete-time) generalizations of the transversal filters that have infinite impulse responses. If the all-pass filter is chosen properly, these filters are usually able to produce acceptable approximations of systems with long impulse responses with a much smaller number of parameters that a transversal filter.

The Laguerre filter, which is another generalization of the transversal filter, has its roots in the pioneering work of Wiener and Lee concerning the synthesis of electric networks using Laguerre functions [9], [10]. The early papers about this subject used truncated Laguerre series to approximate the impulse response of a given continuous-time system [11]–[18]. The discrete-time counterparts of these papers, based on the Laguerre sequences [19]–[21], appeared some years later, and gave rise to the so-called Laguerre filters [22]–[30]. In the last few years Laguerre models (or filters) were applied successfully to several problems in the automatic control field [31]–[37], [24], [38]–[40]. Other recent applications of the Laguerre functions and sequences in signal processing can be found in [41]–[43].

The main advantage of the Laguerre filter in relation to the transversal filter is that the former is an IIR filter with one adjustable multiple pole and the latter is a FIR filter with a fixed multiple pole at the origin. As we will see later on, if the pole of the Laguerre filter is placed at the origin the Laguerre filter degenerates into the transversal filter, i.e., we may consider the Laguerre filter to be a generalization of the transversal filter. By adjusting the pole position of the Laguerre filter it is possible to control the rate of decay of its impulse response, which is quite useful to provide good approximations of systems with long impulse responses.

Due to space limitations we will only discuss in this paper the discrete-time Laguerre filters. Similar results can be easily obtained for the continuous-time Laguerre filters, which are left as an exercise to the interested reader (see also [44]). For the same reason we will also not discuss here the adaptation of the weights and of the pole of Laguerre filters.

The structure of this paper is the following. In section II we review some mathematical material necessary for the understanding of this paper. In section III we describe briefly the main properties of transversal filters. These filters are then generalized in section IV, giving rise to the so-called Laguerre filters. As these filters have one additional parameter, the Laguerre pole position, which affects considerably their performance, we present in subsection IV-A a simple condition that the optimal value of this parameter must satisfy.

In section V we introduce the lattice form of the Laguerre filter, which is the base of another proof of the "optimality" condition for the Laguerre pole position. In section VI we present a simple example that illustrates some of the results of this paper. Finally, in section VII we summarize the contents of this paper and describe briefly other generalizations of transversal filters.

II. NOTATION, DEFINITIONS, AND SOME USEFUL FACTS ABOUT HILBERT SPACE THEORY AND LEAST MEAN-SQUARE APPROXIMATIONS

The majority of the definitions and results presented in this section can be found in [45], [3], [46]–[48].

We will denote matrices and vectors respectively by upper and lower case bold letters. The indexes of the elements of matrices and vectors will start from zero and not from one. The letter k will be the discrete time variable. The Kronecker's delta will be denoted by δ_{ij} (it is equal to one if $i = j$ and equal to zero otherwise).

A. The Hilbert space ℓ^2

A Hilbert space is an inner product space which is a complete metric space with respect to the metric induced by its inner product [46]. This means that a Hilbert space is a linear vector space, possibly of infinite dimension, with an inner product operation defined between any two of its elements (an inner product space). This inner product is used to define the norm of an element of that space (norm induced by the inner product), which is simply the square root of the inner product of that element with itself. This norm is used in turn to define a distance (metric) between any two elements of that space (a metric space), that is the norm of the difference between these two elements. The remaining characteristic of a Hilbert space is that the metric space is complete (or closed). This means that any convergent (Cauchy) sequence of elements of that space converges to an element of that space. Two elements of a Hilbert space are said to be orthogonal if their inner product is zero. An element of a Hilbert space is said to be normal if its norm is equal to one. For an introductory exposition of Hilbert spaces we refer the reader to [46].

The problem of finding the best approximation of an arbitrary element of a Hilbert space by an element of a (linear) subspace of that Hilbert space is solved by the principle of orthogonality (a consequence of the projection theorem [46], [47]), which states that the error of the (unique) best approximation is orthogonal to the subspace in question.

A real sequence $f(k)$ belongs to the Hilbert space ℓ^2, the space of all square-summable causal sequences, if and only if [46]

$$\sum_{k=0}^{+\infty} f^2(k) < \infty.$$

Note that all absolutely summable causal sequences belong to ℓ^2 [45], i.e., the impulse responses of all causal stable linear systems belong to this Hilbert space.

The Fourier transform of a sequence $f(k)$ belonging to ℓ^2 is defined by[1]

$$F(e^{\mathrm{j}\omega}) = \sum_{k=0}^{+\infty} f(k)\, e^{-\mathrm{j}\omega k}.$$

The function $F(e^{\mathrm{j}\omega})$ is a square-integrable function, in the sense of Lebesgue, on the unit circle [48].

The inner product between any two sequences, $f(k)$ and $g(k)$, of ℓ^2 is defined by

$$\langle f(k), g(k)\rangle = \sum_{k=0}^{+\infty} f(k)g(k).$$

Because both $F(e^{\mathrm{j}\omega})$ and $G(e^{\mathrm{j}\omega})$ are square-integrable functions on the unit circle we may also evaluate this inner product by the formula (Parseval's theorem)

$$\langle f(k), g(k)\rangle = \frac{1}{2\pi}\int_{\pi}^{+\pi} F(e^{\mathrm{j}\omega})G^*(e^{\mathrm{j}\omega})\, d\omega.$$

In particular,

$$\langle f(k), f(k)\rangle = \frac{1}{2\pi}\int_{-\pi}^{+\pi} \left|F(e^{\mathrm{j}\omega})\right|^2 d\omega. \qquad (1)$$

A set of sequences of ℓ^2 is said to be complete if *any* sequence of that Hilbert space can be approximated arbitrarily well (in the norm induced by the inner product) by a linear combination of the sequences of that set. If these sequences are orthonormal (both orthogonal and normal) then the set is called an orthonormal basis of ℓ^2.

Let $\{f_i(k)\}_{i=0}^{+\infty}$ be an orthonormal basis of ℓ^2. Then, any sequence $g(k)$ belonging to ℓ^2 can be expanded in the form (orthonormal expansion)

$$g(k) = \sum_{i=0}^{+\infty} c_i f_i(k)$$

where $c_i = \langle g(k), f_i(k)\rangle$ are the Fourier coefficients of $g(k)$ with respect to the orthonormal set $\{f_i(k)\}_{i=0}^{+\infty}$.

The canonical basis of ℓ^2 is the orthonormal set of sequences $\{\delta(k-i)\}_{i=0}^{+\infty}$, where $\delta(k-i) = \delta_{ki}$ are the "pulse sequences" (these sequences are nonzero only for $k = i$). The canonical basis is the simplest example of a complete orthonormal set of ℓ^2.

B. Stochastic processes

Let X and Y be two real random variables with zero mean and finite variance. The inner product between these two random variables is defined by their covariance (or correlation), i.e., by

$$\langle X, Y\rangle = \mathrm{E}[XY]$$

where $\mathrm{E}[\cdot]$ denotes mathematical expectation. In particular, $\langle X, X\rangle = \mathrm{Var}[X]$ is the variance (mean-square value) of X.

[1] The symbol j (not to be confused with j) denotes the square root of -1.

Let $x(k)$ and $y(k)$ be two real wide-sense stationary stochastic processes with zero mean and finite variance.[2] The cross-correlation function between these two stochastic processes is defined by

$$R_{xy}(\tau) = \mathrm{E}[x(k+\tau)y(k)].$$

Because these stochastic processes are wide-sense stationary their cross-correlation does not depend on the time variable k. The cross-power spectral density between $x(k)$ and $y(k)$ is the Fourier transform of $R_{xy}(\tau)$, given by

$$\Phi_{xy}(e^{\mathrm{j}\omega}) = \sum_{\tau=-\infty}^{+\infty} R_{xy}(\tau)\, e^{-\mathrm{j}\omega\tau}.$$

It should be stressed that this expression must be used with care, because it may not converge for some values of ω (on a set of measure zero). If $R_{xy}(\tau)$ is an absolutely summable sequence $\Phi_{xy}(e^{\mathrm{j}\omega})$ can be considered to be the z-transform of $R_{xy}(\tau)$ evaluated on the unit circle. (Note that in this case $\Phi_{xy}(z)$ need not be analytic on the unit circle. Without stronger conditions the best that can be said is that it converges uniformly there.)

It is possible to recover the cross-correlation function from the cross-spectral density by the formula

$$R_{xy}(\tau) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} e^{\mathrm{j}\omega\tau}\Phi_{xy}(e^{\mathrm{j}\omega})\, d\omega.$$

Due to the aforementioned possible convergence problems of $\Phi_{xy}(e^{\mathrm{j}\omega})$ this integral must be evaluated with care.[3]

The functions $R_{xx}(\tau)$ and $\Phi_{xx}(e^{\mathrm{j}\omega})$ are called respectively autocorrelation and power spectral density of the stochastic process $x(k)$. Note that the variance of $x(k)$ is given by

$$\mathrm{Var}[x(k)] = R_{xx}(0) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} \Phi_{xx}(e^{\mathrm{j}\omega})\, d\omega.$$

Let $H(z)$ be the transfer function of a stable linear system, $x(k)$ its input, and $y(k)$ its output. The input-output relation of this system has to be expressed in the time domain when $x(k)$ (and hence $y(k)$) is a stochastic process, because is is not clear how to apply z-transforms to this kind of signals (see figure 1). It is easy to verify that the power spectral density of $y(k)$ is given by [45]

$$\Phi_{yy}(e^{\mathrm{j}\omega}) = \left|H(e^{\mathrm{j}\omega})\right|^2 \Phi_{xx}(e^{\mathrm{j}\omega}).$$

[2]Note that in this case $x(k)$ and $y(k)$ are sequences of random variables. We will reserve the letters f, g, and h, to represent nonrandom sequences.

[3]Strictly speaking, the above integral should be replaced by the following Stieltjes integral [47]

$$R_{xy}(\tau) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} e^{\mathrm{j}\omega\tau}\, d\Psi_{xy}(\omega)$$

where

$$\Psi_{xy}(\omega) = \int_{-\pi}^{\omega} \Phi_{xy}(e^{\mathrm{j}\nu})\, d\nu$$

is the cross-spectral distribution function. This formalism accounts for possible impulses (Dirac delta distributions) in $\Phi_{xy}(e^{\mathrm{j}\omega})$ without resorting to the theory of distributions. We will avoid such technicalities here.

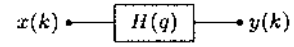


Fig. 1 - Block diagram of a causal stable system with transfer function $H(z) = \sum_{i=0}^{+\infty} h(i)z^{-i}$. The symbol q is the advance operator, i.e., $q[x(k)] = x(k+1)$, and q^{-1} is its inverse (the delay operator). The output of this system is given by the convolution of $x(k)$ with $h(k)$, i.e., by $y(k) = \sum_{i=0}^{+\infty} h(i)x(k-i) = H(q)x(k)$.

Because $H(z)$ converges uniformly and is bounded on the unit circle if the system is stable (remember that $h(k)$ is in this case an absolutely summable sequence), it is easy to verify that if $x(k)$ has zero mean and finite variance then so will $y(k)$. Therefore, the variance of $y(k)$ is given by

$$\langle y(k),y(k)\rangle = \frac{1}{2\pi}\int_{-\pi}^{+\pi} \left|H(e^{\mathrm{j}\omega})\right|^2 \Phi_{xx}(e^{\mathrm{j}\omega})\, d\omega. \quad (2)$$

Let $F(z)$ and $G(z)$ be two stable linear systems excited respectively by the stochastic processes $x(k)$ and $y(k)$, assumed to be correlated, with outputs $u(k) = F(q)x(k)$ and $v(k) = G(q)y(k)$, respectively. The inner product between $u(k)$ and $v(k)$ is given by

$$\begin{aligned}\langle u(k),v(k)\rangle &= \sum_{i,j=0}^{+\infty} f(i)\mathrm{E}[x(k-i)y(k-j)]g(j)\\ &= \sum_{i,j=0}^{+\infty} f(i)\left[\frac{1}{2\pi}\int_{-\pi}^{+\pi} e^{\mathrm{j}\omega(j-i)}\Phi_{xy}(e^{\mathrm{j}\omega})\, d\omega\right] g(j)\\ &= \frac{1}{2\pi}\int_{-\pi}^{+\pi} F(e^{\mathrm{j}\omega})G^*(e^{\mathrm{j}\omega})\,\Phi_{xy}(e^{\mathrm{j}\omega})\; d\omega. \quad (3)\end{aligned}$$

This formula will be useful later on. Note that (2) is a special case of (3).

C. Least mean-square approximations

Consider the problem of the approximation of a random variable Y by a linear combination of $n+1$ other random variables $X_0,\ldots,X_n$, correlated with Y, such that the variance (mean-square) of the approximation error is as small as possible. Let Y_n be the approximation of Y, given by

$$Y_n = \sum_{i=0}^{n} w_{n,i}X_i,$$

and let $E_n = Y - Y_n$ be the approximation error, whose variance is $\xi_n = \langle E_n,E_n\rangle$. This approximation problem is naturally formulated and solved in the context of Hilbert spaces, in this case the Hilbert space of random variables with zero mean and finite variance [47].

In our specific problem the subspace where the approximation lies is composed of all linear combinations of the random variables $X_0,\ldots,X_n$, and the principle of orthogonality states that

$$\langle E_n,X_i\rangle = 0, \qquad i = 0,\ldots,n. \quad (4)$$

These equations are usually called normal equations and can be deduced without resorting to Hilbert space theory, equating the partial derivatives of ξ_n with respect to each of the

$w_{n,i}$'s to zero. Assuming that the normal equations are satisfied the (least) mean-square error of the approximation is given by

$$\xi_n = \langle E_n,Y\rangle = \langle Y,Y\rangle - \sum_{i=0}^{n} w_{n,i}\,\langle Y,X_i\rangle. \qquad (5)$$

The normal equations (4) can be put together in only one equation, of the form

$$\begin{bmatrix} \langle X_0,X_0\rangle & \cdots & \langle X_0,X_n\rangle \\ \vdots & \ddots & \vdots \\ \langle X_n,X_0\rangle & \cdots & \langle X_n,X_n\rangle \end{bmatrix} \begin{bmatrix} w_{n,0} \\ \vdots \\ w_{n,n} \end{bmatrix} = \begin{bmatrix} \langle Y,X_0\rangle \\ \vdots \\ \langle Y,X_n\rangle \end{bmatrix}.$$

This equation can be written in the condensed form $\mathbf{R}_n\mathbf{w}_n = \mathbf{p}_n$, with obvious definitions for the square matrix $\mathbf{R}_n$, and for the vectors $\mathbf{w}_n$ and $\mathbf{p}_n$. The matrix $\mathbf{R}_n$ is symmetric and nonnegative definite. This second fact is a trivial consequence of

$$\mathbf{w}_n^{\mathrm{T}}\mathbf{R}_n\mathbf{w}_n = \langle Y_n,Y_n\rangle \geq 0. \qquad (6)$$

A very important case of the approximation problem studied here occurs when $\langle X_i,X_j\rangle = \delta_{ij}$. In this case $\{X_i\}_{i=0}^n$ is an orthonormal set and $\mathbf{R}_n$ is the $n+1 \times n+1$ identity matrix, which implies that $w_{n,i} = \langle Y,X_i\rangle \triangleq c_i$ does not depend on n. The best approximation to Y is then given by the simple formula

$$Y_n = \sum_{i=0}^{n} c_i X_i,$$

and the least mean-square error of the approximation is given by

$$\xi_n = \langle Y,Y\rangle - \sum_{i=0}^{n} c_i^2.$$

Note that $Y_n = Y_{n-1} + c_n X_n$, and that $\xi_n = \xi_{n-1} - c_n^2$, with the initial values $Y_{-1} = 0$ and $\xi_{-1} = \langle Y,Y\rangle$.

From a given linearly independent set $\{X_i\}_{i=0}^n$ it is possible to construct an orthogonal set $\{X_i^b\}_{i=0}^n$ where $X_i^b = \sum_{j=0}^{i} b_{ij}X_j$ with the restriction $b_{ii} = 1$, using the Gram-Schmidt orthogonalization procedure [49]. Note that the constants b_{ij} are such that $\langle X_i^b,X_j^b\rangle = 0$ for $0 \leq j < i$, which in turn implies that $\langle X_i^b,X_j\rangle = 0$ also for $0 \leq j < i$. Remembering the orthogonality principle of best approximations in Hilbert spaces it is easy to verify that X_i^b is precisely the error of the best approximation of X_i by a linear combination of the random variables $X_0,\ldots,X_{i-1}$.

D. Approximations of linear systems

Consider the problem of the approximation of a given stable and causal system $H(z)$ by another stable and causal system $H_n(z)$. (The exact form of $H_n(z)$ is irrelevant to the present discussion.) Both systems are excited by the same stochastic process $x(k)$ and the objective of the approximation is to minimize the variance of the error signal $e_n(k)$, which is the difference between the outputs of both systems.

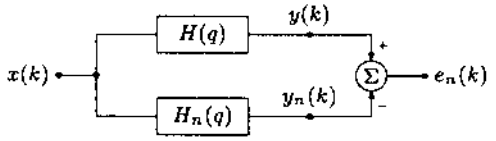


Fig. 2 - Model of the approximation of a general system, $H(z)$, by another system, $H_n(z)$. The objective of the approximation is the minimization of the variance of $e_n(k)$ by adjusting some parameters of $H_n(z)$.

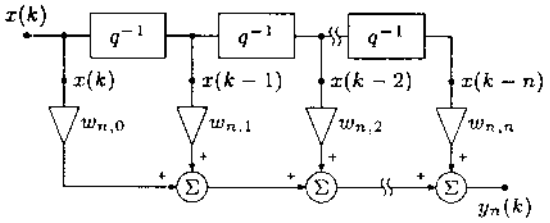


Fig. 3 - Transversal filter of order n (with n delays). The output of this filter is given by $y_n(k) = \sum_{i=0}^{n} w_{n,i}\,x(k-i)$, and is used to approximate a given desired signal, $y(k)$, correlated with $x(k)$.

This situation is depicted in figure 2. Applying (2) to this case gives for the variance of $e_n(k)$ the formula

$$\xi_n = \frac{1}{2\pi}\int_{-\pi}^{+\pi} \left|H(e^{\mathrm{j}\omega}) - H_n(e^{\mathrm{j}\omega})\right|^2 \Phi_{xx}(e^{\mathrm{j}\omega})\,d\omega. \qquad (7)$$

We are interested in the comparison of this formula with the following ℓ^2 inner product

$$\frac{1}{2\pi}\int_{-\pi}^{+\pi} \left|H(e^{\mathrm{j}\omega}) - H_n(e^{\mathrm{j}\omega})\right|^2 d\omega. \qquad (8)$$

If $\Phi_{xx}(e^{\mathrm{j}\omega})$ is an essentially bounded Lebesgue measurable function on the interval $[-\pi,+\pi]$ and if (8) converges to zero when n goes to infinity then (7) will also converge to zero. For example, this will happen if $R_{xx}(\tau)$ is absolutely summable and if $h_n(k)$, the impulse response of $H_n(z)$, is a linear combination of the first $n+1$ sequences of a complete set of ℓ^2. (Obviously, each one of the sequences of that set must be absolutely summable, otherwise $H_n(z)$ could be unstable.) For a general power spectral density the same result holds if $H_n(e^{\mathrm{j}\omega})$ converges to $H(e^{\mathrm{j}\omega})$ for all frequencies where $\Phi_{xx}(e^{\mathrm{j}\omega})$ has a Dirac delta distribution. For example, this will happen if the complete set used to form $h_n(k)$ is the canonical basis of ℓ^2.

Note that (7) may be null even when (8) is non-null if $\Phi_{xx}(e^{\mathrm{j}\omega})$ vanishes on a set of $[-\pi,+\pi]$ with nonzero measure, i.e., if $x(k)$ is a band limited process. This cannot happen if $\Phi_{xx}(e^{\mathrm{j}\omega})$ is strictly positive for (almost) all $\omega \in [-\pi,+\pi]$, a condition usually called persistence of excitation (of infinite order) [50].

III. THE TRANSVERSAL FILTER

Consider the transversal filter of figure 3. The weights of this filter that minimize the variance of the error of the approximation of a given desired signal $y(k)$, correlated with $x(k)$, by $y_n(k)$ satisfy the Wiener-Hopf equations [3] (cf. the normal equations (4))

$$\sum_{i=0}^{n} w_{n,i}\,\langle x(k-i),x(k-j)\rangle = \langle y(k),x(k-j)\rangle$$

for $0 \leq j \leq n$. These equations can be put in the form

$$\mathbf{R}_n \mathbf{w}_n = \mathbf{p}_n \tag{9}$$

where the elements of $\mathbf{R}_n$, which can be computed easily using (3), are given by

$$\begin{aligned} r_{ij} &= \langle x(k-i), x(k-j) \rangle \\ &= \frac{1}{2\pi} \int_{-\pi}^{+\pi} e^{\mathrm{j}\omega(j-i)} \Phi_{xx}(e^{\mathrm{j}\omega})\, d\omega \end{aligned} \tag{10}$$

$(0 \leq i, j \leq n)$, and where those of $\mathbf{p}_n$ are given by

$$\begin{aligned} p_i &= \langle y(k), x(k-i) \rangle \\ &= \frac{1}{2\pi} \int_{-\pi}^{+\pi} e^{\mathrm{j}\omega i} \Phi_{yx}(e^{\mathrm{j}\omega})\, d\omega \end{aligned}$$

$(0 \leq i \leq n)$. Note that r_{ij} only depends on $|i-j|$. This means that the elements of each diagonal of $\mathbf{R}_n$ are equal, i.e., $\mathbf{R}_n$ is a symmetric Toeplitz matrix. It is possible to explore the Toeplitz structure of this matrix to solve (9), as done for example in the Levinson algorithm [3], [49]. This gives rise to the so-called lattice filters, which we will discuss later on in the context of Laguerre filters.

The smallest and largest eigenvalues of $\mathbf{R}_n$ play an important role not only in the resolution of the Wiener-Hopf equations, where they define the numerical stability of the system of normal equations [49], but also in the convergence speed of certain adaptive algorithms of the weights of transversal filters [3]. Fortunately, it is easy to obtain simple bounds for these eigenvalues if we restrict the input signal of the transversal filter to have an absolutely summable autocorrelation function (less restrictive results can be found in [51]). In this case the power spectral density converges uniformly (and is bounded) on the interval $\omega \in [-\pi, +\pi]$.

A well know method to compute the smallest and largest eigenvalues of a symmetric matrix, in this case $\mathbf{R}_n$, is to evaluate the minimum and maximum values of the Rayleigh quotient [49]:

$$\lambda_{\min}(\mathbf{R}_n) = \min_{\mathbf{w}_n \neq \mathbf{0}} \frac{\mathbf{w}_n^{\mathrm{T}} \mathbf{R}_n \mathbf{w}_n}{\mathbf{w}_n^{\mathrm{T}} \mathbf{w}_n},$$

$$\lambda_{\max}(\mathbf{R}_n) = \max_{\mathbf{w}_n \neq \mathbf{0}} \frac{\mathbf{w}_n^{\mathrm{T}} \mathbf{R}_n \mathbf{w}_n}{\mathbf{w}_n^{\mathrm{T}} \mathbf{w}_n},$$

where $\mathbf{w}_n$ is an arbitrary nonnull vector with $n+1$ elements. In our concrete case it is easy to verify that (cf. (6) and (2))

$$\mathbf{w}_n^{\mathrm{T}} \mathbf{R}_n \mathbf{w}_n = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \left| \sum_{i=0}^{n} w_{n,i}\, e^{-\mathrm{j}\omega i} \right|^2 \Phi_{xx}(e^{\mathrm{j}\omega})\, d\omega,$$

and that (because the functions $e^{-\mathrm{j}\omega i}$ are orthonormal in the interval $[-\pi, +\pi]$)

$$\mathbf{w}_n^{\mathrm{T}} \mathbf{w}_n = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \left| \sum_{i=0}^{n} w_{n,i}\, e^{-\mathrm{j}\omega i} \right|^2 d\omega.$$

Using trivial lower and upper bounds for the power spectral density in the first of these two formulas it is easy to conclude that

$$\inf_{\omega} \Phi_{xx}(e^{\mathrm{j}\omega}) \leq \lambda_{\min}(\mathbf{R}_n) \leq \lambda_{\max}(\mathbf{R}_n) \leq \sup_{\omega} \Phi_{xx}(e^{\mathrm{j}\omega}) \tag{11}$$

where the infimum and supremum are over all values of ω in the interval $[-\pi, +\pi]$. In particular, if the power spectral density $\Phi_{xx}(e^{\mathrm{j}\omega})$ is bounded away from zero (persistence of excitation), then $\lambda_{\min}(\mathbf{R}_n) > 0$, i.e., $\mathbf{R}_n$ is non-singular. In that case the Wiener-Hopf equations have only one solution, irrespective of the value of n.

IV. THE LAGUERRE FILTER

Consider again the transversal filter of figure 3. Denote by $H(z)$ the transfer function of the (stable and causal) system that produces $y(k)$ when excited by $x(k)$. Applying the results of subsection II-D to the transversal filter it becomes clear that this filter is indirectly trying to approximate the impulse response of $H(z)$ by the first $n+1$ sequences of the canonical basis of ℓ^2. Unfortunately, these sequences are extremely localized in time. Hence, the quality of the approximation will be very poor (for small n) when the impulse response of $H(z)$ is very long (e.g., when it decays slowly to zero). Note, however, that because the canonical basis is a complete set of ℓ^2 the approximation error can be made arbitrarily small by using a sufficiently large n (cf. section II-D).

It is possible to use other complete sets of ℓ^2 to build a "transversal-like" filter. In order for the filter to be practical to use each sequence of that set should be easy to generate digitally, i.e., it should have a rational z-transform. Probably the simplest set of such sequences (besides the canonical basis) is the set of the Laguerre sequences [19], which is a complete orthonormal set of ℓ^2 [20]. The z-transforms of these sequences are given by [21]

$$L_i(z,u) = \sqrt{1-u^2}\, \frac{(z^{-1}-u)^i}{(1-uz^{-1})^{i+1}}, \qquad i \geq 0 \tag{12}$$

where u is a free (real) parameter, the Laguerre pole position, with modulus smaller that one. Note that $L_i(z,0) = z^{-i}$, i.e., the sequences of the canonical basis of ℓ^2 are a special case of the Laguerre sequences. Also interesting is the fact that for $i \geq 0$

$$L_{i+1}(z,u) = A(z,u) L_i(z,u) \tag{13}$$

with

$$A(z,u) = \frac{z^{-1}-u}{1-uz^{-1}},$$

i.e., these sequences can be generated in cascade, starting with a first order low-pass section ($L_0(z,u)$), followed by first order all-pass sections ($A(z,u)$).

Replacing the "backbone" of the transversal filter, that generates the first $n+1$ canonical sequences of ℓ^2 when excited by $\delta(k)$, by the equivalent structure that generates the first $n+1$ Laguerre sequences (when excited by $\delta(k)$) we obtain the Laguerre filter shown in figure 4. This filter structure was introduced in [22] and was studied with some detail in [24]. We will assume, unless stated otherwise, that

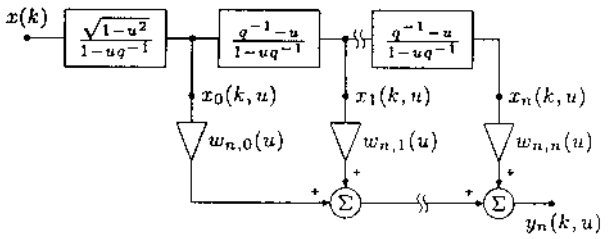


Fig. 4 - Laguerre filter of order n (with n sections). This filter is stable if and only if $|u| < 1$. For $u = 0$ the Laguerre filter degenerates into the familiar transversal filter.

for each value of u the weights of this filter are computed such that $y_n(k,u)$ is the best approximation, in the mean-square sense, to a given desired signal $y(k)$. This explains why these weights are functions of u in figure 4.

The output of an order n Laguerre filter excited by a real wide-sense stationary stochastic process $x(k)$ with zero-mean is given by

$$y_n(k,u) = \sum_{i=0}^{n} w_{n,i}(u)\, x_i(k,u)$$

where

$$x_i(k,u) = L_i(q,u)\, x(k).$$

For each value of u the optimal set of weights of this filter can be computed from the normal equations

$$\sum_{i=0}^{n} w_{n,i}(u)\, \langle x_i(k,u), x_j(k,u)\rangle = \langle y(k), x_j(k,u)\rangle \quad (14)$$

$(0 \le j \le n)$. These equations can be put in the form

$$\mathbf{R}_n(u)\, \mathbf{w}_n(u) = \mathbf{p}_n(u) \quad (15)$$

where the elements of $\mathbf{R}_n(u)$ are given by

$$r_{ij}(u) = \int_{-\pi}^{+\pi} \left(\frac{e^{j\omega} - u}{1 - ue^{j\omega}}\right)^{j-i} \frac{\Phi_{xx}(e^{j\omega})(1-u^2)\, d\omega}{2\pi\, |1 - ue^{j\omega}|^2}$$

$(0 \le i, j \le n)$, and where those of $\mathbf{p}_n(u)$ are given by

$$p_i(u) = \int_{-\pi}^{+\pi} \left(\frac{e^{j\omega} - u}{1 - ue^{j\omega}}\right)^{i} \frac{\Phi_{yx}(e^{j\omega})\, \sqrt{1-u^2}\, d\omega}{2\pi(1 - ue^{j\omega})}$$

$(0 \le i \le n)$. These expressions can be obtained easily from (3). Similarly to the transversal filter case, $r_{ij}(u)$ depends only on $|i-j|$, i.e., $\mathbf{R}_n(u)$ is a Toeplitz matrix. We will explore this fact in the next section.

It is possible to simplify considerably the expression for $r_{ij}(u)$ using the change of frequency variable $\omega \mapsto \theta$ defined by the bilinear transformation [24]

$$e^{j\theta} = \frac{e^{j\omega} - u}{1 - ue^{j\omega}}. \quad (16)$$

It is easy to verify that when ω goes from $-\pi$ to $+\pi$, θ also goes from $-\pi$ to $+\pi$ (see figure 5), that

$$e^{j\omega} = \frac{e^{j\theta} + u}{1 + ue^{j\theta}}, \quad (17)$$

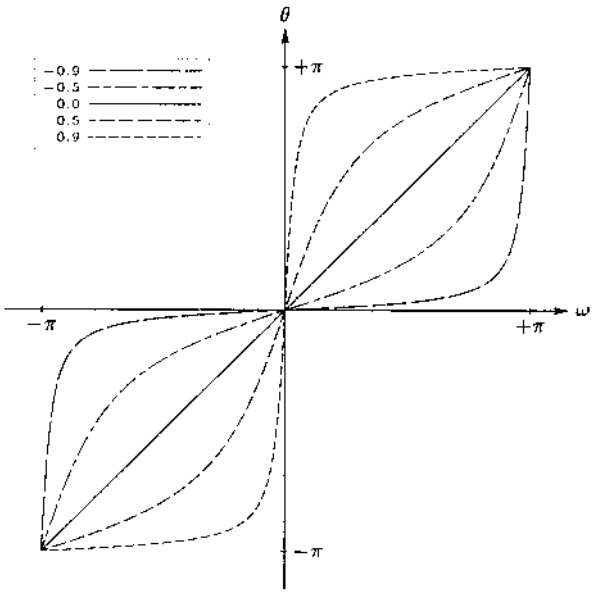


Fig. 5 - Graph of the frequency transformation $\omega \mapsto \theta$ for several values of u. Note that there exists a one to one correspondence between ω and θ. The inverse transformation $\theta \mapsto \omega$, which appears in (18), can be visualized easily by replacing u by $-u$. For $u > 0$ this inverse transformation compresses the low frequencies and for $u < 0$ it compresses the high frequencies.

and that

$$d\theta = \frac{1 - u^2}{|1 - ue^{j\omega}|^2}\, d\omega.$$

It is then trivial to verify that $r_{ij}(u)$ can also be given by [24]

$$r_{ij}(u) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} e^{j\theta(j-i)}\, \Phi_{xx}\left(\frac{e^{j\theta}+u}{1+ue^{j\theta}}\right) d\theta, \quad (18)$$

which only differs from (10) in the argument of the power spectral density of $x(k)$. This very important result can be used to determine immediately bounds for the eigenvalues of $\mathbf{R}_n(u)$ based on those for the matrix $\mathbf{R}_n$ appearing in the transversal filter case. Because (17) represents only a distortion of the frequency scale (see figure 5) it turns out that (11) is also valid for this case, i.e., the lower and upper bounds for the eigenvalues of $\mathbf{R}_n(u)$ are exactly the same as those for $\mathbf{R}_n$ and do not depend on u. Also, some results concerning the asymptotic eigenvalue distribution of $\mathbf{R}_n$ when $n \to \infty$ (see [51] or [52] for details) can be adapted with very little effort to the matrices $\mathbf{R}_n(u)$.

A. Stationarity condition of the MSE of a Laguerre filter with respect to u [30]

The variance of the error signal of a Laguerre filter, i.e., its mean-square error (MSE), is a function of u. In order to minimize this function we need to deduce its stationarity condition and then to solve it. One of the solutions of this condition will be the optimal value of u, for which the MSE attains its global minimum.

The MSE of a Laguerre filter of order n is given by

$$\xi_n(u) = \langle e_n(k,u), e_n(k,u)\rangle.$$

Assume for the moment that the weights of the Laguerre filter are arbitrary, i.e., that they do not depend on u and that

they do not need to satisfy the normal equations. Then, the partial derivative of the MSE with respect to u is given by (a prime denotes differentiation with respect to this variable)

$$\xi_n'(u, \mathbf{w}_n) = -2 \sum_{i=0}^{n} w_{n,i} \langle e_n(k,u), x_i'(k,u) \rangle \qquad (19)$$

where $\mathbf{w}_n$ is a vector whose elements are the weights of the Laguerre filter. Notice that the stationarity conditions of $\xi_n(u, \mathbf{w}_n)$ require (19) to be zero, and they also force the normal equations to be satisfied. This latter condition would have been unnecessary if we had assumed that the weights where computed from the normal equations for each value of u. (This assumption would have made the analysis of the problem much more difficult.)

Because of the remarkable formula [30]

$$L_i'(z,u) = \frac{(i+1)L_{i+1}(z,u) - i\,L_{i-1}(z,u)}{1-u^2}, \qquad (20)$$

the derivative of $x_i(k,u) = L_i(q,u)x(k)$ with respect to u is given by

$$x_i'(k,u) = \frac{(i+1)x_{i+1}(k,u) - i\,x_{i-1}(k,u)}{1-u^2}. \qquad (21)$$

Applying this formula in (19) and simplifying the result with the normal equations (14) we obtain

$$\xi_n'(u) = -2(n+1)\frac{w_{n,n}(u)\,\langle e_n(k,u), x_{n+1}(k,u)\rangle}{1-u^2}. \qquad (22)$$

We emphasize that this formula is only valid if the normal equations are satisfied. Hence, the weights are again functions of u, i.e., they are again computed from the normal equations. (In fact, this formula is the total derivative of $\xi_n(u)$.) Equating this formula to zero gives the stationarity condition of the MSE with respect to u.

The next step is to find the value of the only complicated term of (22): $\langle e_n(k,u), x_{n+1}(k,u)\rangle$. In order to do so we need to orthonormalize the signals $x_i(k,u)$.

Let $x_0^o(k,u), \ldots, x_i^o(k,u)$ be the orthonormalized signals obtained by applying the Gram-Schmidt orthogonalization algorithm (with normalization) to the signals $x_0(k,u), \ldots, x_i(k,u)$. The linear transformation performed by this algorithm can be expressed by

$$\mathbf{x}_i^o(k,u) = \mathbf{T}_i(u)\,\mathbf{x}_i(k,u) \qquad (23)$$

where

$$\mathbf{x}_i(k,u) = [\,x_0(k,u) \quad \cdots \quad x_i(k,u)\,]^{\mathrm{T}}$$

is a vector holding the original signals, and where

$$\mathbf{x}_i^o(k,u) = [\,x_0^o(k,u) \quad \cdots \quad x_i^o(k,u)\,]^{\mathrm{T}}$$

is the corresponding vector holding the orthonormalized signals. The matrix $\mathbf{T}_i(u)$ is a lower-triangular matrix. It will be nonsingular if and only if the signals $x_0(k,u), \ldots, x_i(k,u)$ are linearly independent (we will assume this condition to hold in the sequel). For example, this will happen if $x(k)$ satisfy the persistence of excitation condition.

It is possible to prove that [53]

$$\mathbf{T}_i(u)\,\mathbf{R}_i(u)\,\mathbf{T}_i^{\mathrm{T}}(u) = \mathbf{I}_i,$$

i.e., $\mathbf{T}_i(u)$ is the inverse of the first Cholesky factor of $\mathbf{R}_i(u)$ [49]. For notational convenience we will denote the element of the last line and column of $\mathbf{T}_i(u)$ by $t_i(u)$. Note that $t_i(u) = 1$ for all i if and only if $\Phi_{xx}(e^{j\omega}) = 1$ for all ω, i.e., if and only if $x(k)$ is white noise with unitary variance. Note also that $t_i(u)$ is strictly positive, and that the element in the same position on the lower-triangular matrix $\mathbf{T}_i^{-1}(u)$ is $1/t_i(u)$.

Because the signals $x_i^o(k,u)$ are obtained by a linear combination of the signals $x_j(k,u)$, $j = 0, \ldots, i$, it is clear that the output of a Laguerre filter of order n can also be given by the orthonormal expansion

$$y_n(k,u) = \sum_{i=0}^{n} c_i(u)\,x_i^o(k,u) \qquad (24)$$

with $c_i(u) = \langle y(k), x_i^o(k,u)\rangle$. Note that $c_i(u)$ does not depend on n. It is also clear that the error signal of a Laguerre filter of order $n+1$ is given by

$$e_{n+1}(k,u) = e_n(k,u) - c_{n+1}(u)\,x_{n+1}^o(k,u).$$

This formula implies that

$$\begin{aligned}\langle e_n(k,u), x_{n+1}(k,u)\rangle &= \langle e_{n+1}(k,u), x_{n+1}(k,u)\rangle \\ &+ c_{n+1}(u)\langle x_{n+1}^o(k,u), x_{n+1}(k,u)\rangle.\end{aligned}$$

The last normal equation for the Laguerre filter of order $n+1$ gives $\langle e_{n+1}(k,u), x_{n+1}(k,u)\rangle = 0$. It is also clear that

$$\begin{aligned}y_{n+1}(k,u) &= \mathbf{w}_{n+1}^{\mathrm{T}}(u)\,\mathbf{x}_{n+1}(k,u) \\ &= \mathbf{c}_{n+1}^{\mathrm{T}}(u)\,\mathbf{x}_{n+1}^o(k,u),\end{aligned}$$

with obvious definitions for the vectors $\mathbf{w}_{n+1}(u)$ and $\mathbf{c}_{n+1}(u)$. Due to (23) and to the special form of $\mathbf{T}_{n+1}(u)$ it is then easy to verify that

$$c_{n+1}(u) = w_{n+1,n+1}(u)/t_{n+1}(u). \qquad (25)$$

Due to the special form of $\mathbf{T}_{n+1}^{-1}(u)$ and to the orthonormality of the signals $x_i^o(k,u)$ it is also easy to verify that

$$\langle x_{n+1}^o(k,u), x_{n+1}(k,u)\rangle = 1/t_{n+1}(u).$$

Putting all these facts together gives

$$\langle e_n(k,u), x_{n+1}(k,u)\rangle = \frac{w_{n+1,n+1}(u)}{t_{n+1}^2(u)}.$$

Applying this formula in (22) we obtain

$$\xi_n'(u) = -\frac{2(n+1)\,w_{n,n}(u)\,w_{n+1,n+1}(u)}{(1-u^2)\,t_{n+1}^2(u)}. \qquad (26)$$

Therefore, the stationarity points of the MSE with respect to u satisfy the simple condition [30]

$$w_{n,n}(u)\,w_{n+1,n+1}(u) = 0. \qquad (27)$$

This condition is a generalization of the condition presented in [26] for the particular case where $x(k) = \delta(k)$ (which is the deterministic signal equivalent to white noise). It is interesting to verify that if $w_{n,n}(u) = 0$ then $\xi_n(u) = \xi_{n-1}(u)$, and that if $w_{n+1,n+1}(u) = 0$ then $\xi_n(u) = \xi_{n+1}(u)$. Hence, in each stationary point of $\xi_n(u)$ the graph of this function touches the graph of $\xi_{n-1}(u)$ and/or of $\xi_{n+1}(u)$. We will illustrate this phenomenon in section VI. It is important to stress that usually, *but not always*, the local minima of $\xi_n(u)$ satisfy the conditions $w_{n,n}(u) \neq 0$ and $w_{n+1,n+1}(u) = 0$, in which case $\xi_{n-1}(u) > \xi_n(u) = \xi_{n+1}(u)$.

For simple and efficient ways of solving approximately (27) we refer the reader to [54]. Basically, we approximate $w_{i,i}(u)$ by a truncated Taylor series or by a Padé approximant, and then find the zeros of that approximation. The derivatives of $w_{i,i}(u)$ required to form these approximations can be computed differentiating (15) and using (20) to simplify the result.

V. THE LAGUERRE LATTICE FILTER [55]

As promised earlier we are going to explore the Toeplitz structure of the matrix $\mathbf{R}_n(u)$. This will give rise to the lattice form of the Laguerre filter. The following line of reasoning is a simple generalization of the ideas that led to the standard lattice filter. These ideas can be found in any good book about adaptive filter theory, such as [1]–[3]. Another important work related to the material presented here is [8]. As before, we assume that the signals $x_i(k,u)$ of the Laguerre filter are linearly independent. To simplify the notation we will use the definition $r_{|i-j|}(u) \triangleq r_{ij}(u)$ when referring to the elements of the Toeplitz matrix $\mathbf{R}_n(u)$.

In order to orthogonalize the signals $x_i(k,u)$ it is useful to consider the problem of the minimization of the variance of the following signals, with the restrictions $a_{i0}(u) = b_{i0}(u) = 1$:

$$x_i^f(k,u) = \sum_{j=0}^{i} a_{ij}(u)\, x_j(k,u); \tag{28}$$

$$x_i^b(k,u) = \sum_{j=0}^{i} b_{ij}(u)\, x_{i-j}(k,u). \tag{29}$$

As explained in subsection II-C, $x_i^f(k,u)$ will be orthogonal to $x_j(k,u)$ for $0 < j \leq i$, and $x_i^b(k,u)$ will be orthogonal to $x_j(k,u)$ for $0 \leq j < i$. This implies that the signals $x_i^b(k,u)$ are the result of the Gram-Schmidt orthogonalization procedure applied to the signals $x_i(k,u)$. We will denote the standard deviation (the square root of the variance) of $x_i^f(k,u)$ by $\sigma_i^f(u)$, and that of $x_i^b(k,u)$ by $\sigma_i^b(u)$. Both of these standard deviations are strictly positive because we have assumed that the signals $x_i(k,u)$ are linearly independent.

It is simple to verify that the augmented normal equations[4] for these two problems are (in the following two equations we have omitted, for aesthetical reasons, the dependence on u of all variables)

$$\begin{bmatrix} r_0 & \cdots & r_i \\ \vdots & \ddots & \vdots \\ r_i & \cdots & r_0 \end{bmatrix} \begin{bmatrix} a_{i0} & b_{ii} \\ \vdots & \vdots \\ a_{ii} & b_{i0} \end{bmatrix} = \begin{bmatrix} (\sigma_i^f)^2 & 0 \\ \vdots & \vdots \\ 0 & (\sigma_i^b)^2 \end{bmatrix}. \tag{30}$$

Due to the symmetry of these two problems it is clear that $a_{ij}(u) = b_{ij}(u)$ for all $i \geq 0$ and for $0 \leq j \leq i$, and that $\sigma_i^f(u) = \sigma_i^b(u) \triangleq \sigma_i(u)$ also for all $i \geq 0$.

The reason why the signals $x_i^f(k,u)$ are also useful is related to the special form of (30), that implies that

$$\begin{bmatrix} r_0 & \cdots & r_{i+1} \\ \vdots & \ddots & \vdots \\ r_{i+1} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ a_{i1} & a_{ii} \\ \vdots & \vdots \\ a_{ii} & a_{i1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \sigma_i^2 & \Delta_i \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ \Delta_i & \sigma_i^2 \end{bmatrix}. \tag{31}$$

Note that $\Delta_i(u)$ and $\sigma_i(u)$ can be computed as soon as the coefficients $a_{ij}(u)$ are know.

From (31) it is very easy to obtain the order update formulae for the weights $a_{ij}(u)$ (and also for the weights $b_{ij}(u)$), which are

$$a_{i+1,j}(u) = a_{ij}(u) + k_{i+1}(u)\, a_{i,i+1-j}(u)$$

for $0 \leq j \leq i+1$, with $k_{i+1}(u) = -\Delta_i(u)/\sigma_i^2(u)$, and with $a_{i,i+1}(u) \triangleq 0$. The application of these formulae in (28) and in (29), together with (13), gives

$$x_{i+1}^f(k,u) = x_i^f(k,u) + k_{i+1}(u)\, A(q,u)\, x_i^b(k,u)$$

and

$$x_{i+1}^b(k,u) = A(q,u)\, x_i^b(k,u) + k_{i+1}(u)\, x_i^f(k,u),$$

with $x_0^f(k,u) = x_0^b(k,u) = x_0(k,u)$. These recursion equations define part of the Laguerre lattice filter. It is also easy to show that

$$k_{i+1}(u) = -\frac{\langle A(q,u) x_i^b(k,u), x_i^f(k,u) \rangle}{\sigma_i^2(u)},$$

and that

$$\sigma_{i+1}^2(u) = [1 - k_{i+1}^2(u)]\, \sigma_i^2(u).$$

From this last formula we conclude, if $\sigma_{i+1}(u) > 0$, that $|k_{i+1}(u)| < 1$. This implies that the inverse Laguerre lattice filter is stable [56]. The coefficients $k_i(u)$ are sometimes called reflection coefficients.

It is clear that the output of the Laguerre filter is given by the orthogonal expansion (compare with (24))

$$y_n(k,u) = \sum_{i=0}^{n} d_i(u)\, x_i^b(k,u)$$

where

$$d_i(u) = \frac{\langle y(k), x_i^b(k,u) \rangle}{\sigma_i^2(u)}$$

does not depend on n. This formula defines the joint-process part of the Laguerre lattice filter, shown in figure 6.

[4] The augmented normal equations are the equations (4) and (5) put together in only one equation.

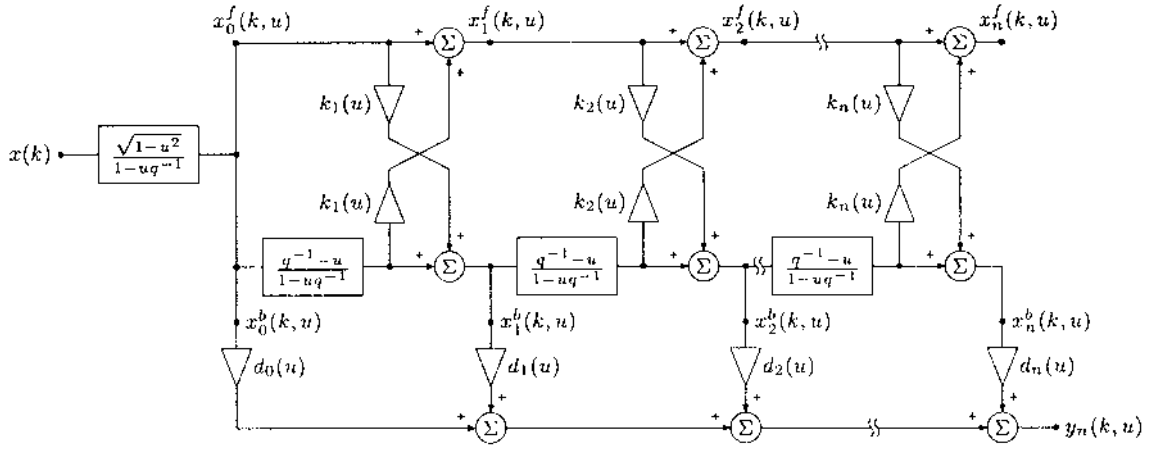


Fig. 6 - Laguerre lattice filter of order n. For $u = 0$ this is the familiar lattice filter.

A. *Stationarity condition of the MSE of a Laguerre lattice filter with respect to u [55]*

We start by normalizing the signals $x_i^b(k,u)$, obtaining the signals

$$x_i^o(k,u) = \frac{x_i^b(k,u)}{\sigma_i(u)}.$$

Using these signals the output of the Laguerre filter can also be given by (this is a repetition of (24))

$$y_n(k,u) = \sum_{i=0}^{n} c_i(u)\, x_i^o(k,u)$$

with $c_i(u) = \sigma_i(u)\, d_i(u)$, and its MSE is given by

$$\xi_n(u) = \langle y(k), y(k) \rangle - \sum_{i=0}^{n} c_i^2(u).$$

Because

$$x_i^o(k,u) = \sum_{j=0}^{i} \frac{b_{i,i-j}(u)}{\sigma_i(u)}\, x_j(k,u)$$

it is not very difficult to show, using (21), that

$$\frac{dx_i^o(k,u)}{du} = \sum_{j=0}^{i+1} \alpha_{ij}(u)\, x_j^o(k,u) \qquad (32)$$

with

$$\alpha_{i,i+1}(u) = \frac{(i+1)\,\sigma_{i+1}(u)}{(1-u^2)\,\sigma_i(u)}.$$

The exact value of the other α_{ij}'s will not be needed. It will prove useful to change the upper limit of the summation in (32) from $i+1$ to ∞. This is accomplished with the definition $\alpha_{ij}(u) \triangleq 0$ for $j > i+1$.

Differentiating the orthonormality condition

$$\langle x_i^o(k,u), x_j^o(k,u) \rangle = \delta_{ij}$$

with respect to u and using (32) it is easy to verify that for all $i, j \geq 0$

$$\alpha_{ij}(u) + \alpha_{ji}(u) = 0.$$

This formula implies that $\alpha_{ij}(u) = 0$ for $j = i$ and also for $j < i-1$. Because

$$c_i(u) = \langle y(k), x_i^o(k,u) \rangle$$

it is then clear that

$$c_i'(u) = \alpha_{i,i-1}(u)\, c_{i-1}(u) + \alpha_{i,i+1}(u)\, c_{i+1}(u). \qquad (33)$$

The derivative of the MSE with respect to u is given by

$$\xi_n'(u) = -2 \sum_{i=0}^{n} c_i(u)\, c_i'(u).$$

Using (33) and $\alpha_{i,i-1}(u) = -\alpha_{i-1,i}(u)$ this summation becomes a telescopic series (!) whose sum is

$$\xi_n'(u) = -2\alpha_{n,n+1}(u)\, c_n(u)\, c_{n+1}(u).$$

Note that this formula is in accord with (26) because $t_i(u) = 1/\sigma_i(u)$. It is then very easy to verify that

$$\xi_n'(u) = -\frac{2(n+1)\,\sigma_{n+1}^2(u)\, d_n(u)\, d_{n+1}(u)}{(1-u^2)}. \qquad (34)$$

Hence, the stationarity condition is

$$d_n(u)\, d_{n+1}(u) = 0. \qquad (35)$$

Note how easily this condition can be interpreted: the MSE of a Laguerre lattice filter has a stationary point with respect to u if and only if the last weight used to compute the output signal vanishes and/or the first unused weight vanishes.

Although (35) could also have been obtained much more easily from (27) and (25), its deduction given above is entirely based on the Laguerre lattice filter, and is interesting in its own right.

VI. AN EXAMPLE

To illustrate the approximation capabilities of Laguerre filters we used a Laguerre lattice filter with 10 sections to approximate the output of a third order elliptic low pass filter with the following transfer function

$$H(z) = \frac{0.01624(1+z^{-1})(1-1.7313z^{-1}+z^{-2})}{(1-0.8957z^{-1})(1-1.8445z^{-1}+0.9282z^{-2})},$$

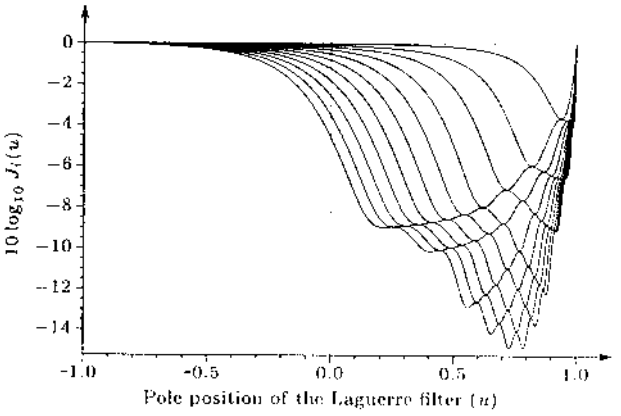


Fig. 7 - Normalized MSE (in dB) of the Laguerre lattice filters of orders from 0 (top) to 10 (bottom) as functions of u. The Laguerre filters were used to approximate a low pass system excited by colored Gaussian noise. Note that these curves touch only at their local extrema. Note also the bad performance of the transversal filter ($u = 0$) when compared to the best Laguerre filter, i.e., with optimal u, of the same order.

excited by colored Gaussian noise generated by feeding (pseudo) white Gaussian noise of unitary variance to a filter with transfer function

$$N(z) = \frac{0.5 + 1.5z^{-1}}{1 + 0.4z^{-1}}.$$

The same signal was used as input of the Laguerre lattice filter. The approximation was performed off-line using 2500 samples of the input signal, previously recorded from one realization of the (pseudo) white Gaussian process. To reduce the effects of the null initial conditions, the first 500 samples were used only to initialize the Laguerre lattice filter. The other 2000 samples were used to compute the reflection coefficients ($k_i(u)$), the joint-process weights ($d_i(u)$), and the MSE ($\xi_i(u)$) of the Laguerre lattice filters of orders from 0 to 10. (Remember that a Laguerre lattice filter of order n effectively contains all Laguerre lattice filters of lower orders.) The algorithm used to compute these coefficients is presented in the appendix.

The normalized MSE error[5] curves for the eleven Laguerre lattice filters are presented in figure 7. Note that consecutive curves touch only where they have local extrema, which is in accord with (35). Although in this example all local minima (maxima) of $J_i(u)$ are associated with the condition $d_{i+1}(u) = 0$ ($d_i(u) = 0$) this is not always the case. Figure 7 also shows that the MSE curve of a Laguerre filter usually has local minima. This is an usual characteristic of insufficient order IIR filters used in an output error configuration.

VI. CONCLUSIONS

We have seen that the transversal filter can be generalized to a filter structure, the Laguerre filter, which has one additional free parameter that controls the filter's (multiple) pole position. Setting this parameter to 0 puts the (multiple) pole of the filter at the origin, turning the Laguerre filter into a transversal filter. By adjusting properly this parameter, which controls the rate of decay to zero of the filter's impulse response, allows this filter to provide good approximations to systems with slowly decaying impulse responses.

It is easy to devise algorithms to adapt the weights of Laguerre filters similar to the LMS or the RLS developed for the transversal filter [1]–[3]. It is also easy to adapt the reflection coefficients and joint-process weights of the Laguerre lattice filters using a stochastic gradient approach similar to the one used for transversal filters [1]–[3]. Unfortunately, it appears that it is not easy to generalize the FTF and LSL fast adaptation algorithms [1]–[4] to the Laguerre filter. Finally, it is possible to extend the adaptation (using a LMS scheme) to the Laguerre pole position. In this respect, the equations (26) and (34) are useful (specially the latter).

Besides the Laguerre functions there are other complete orthonormal sets of ℓ^2 whose sequences have rational z-transforms. In this respect the Kautz functions [57] and sequences [21] are particularly useful. Replacing the Laguerre sequences by the Kautz sequences we obtain a Kautz filter, which appears to be very promising in the approximation of systems with a dominant complex pole pair [58]–[62].

APPENDIX

The following C code implements an algorithm to compute the coefficients of the orthonormal expansion (24) given the symmetric Toeplitz matrix $\mathbf{R}_n(u)$ (more properly, its first line) and the vector $\mathbf{p}_n(u)$. From these coefficients it is very easy to compute the MSE of the Laguerre lattice filters with up to n sections. Note that this algorithm is slightly different than the usual Levinson algorithm [49] that solves the system $\mathbf{R}_n \mathbf{w}_n = \mathbf{p}_n$. Here we are not interested in $\mathbf{w}_n$ but in the coefficients of the orthonormal expansion (24).

```
/*
** Modified Levinson algorithm
**
** Inputs:
**   n       --- Number of sections
**   r[0..n] --- Elements of the first line
**                 of the R Toeplitz matrix
**   p[0..n] --- Elements of the p vector
** Outputs:
**   c[0..n] --- Weights of the orthonormal
**                 expansion
** Internal variables:
**   k  --- Reflection coefficient of each
**            section
**   s2 --- Variance of orthogonal output
**            signal of each section
*/

typedef double real;
#define nMax 10

void modLevinson(int n,real *r,real *p,real *c)
{
  real k,s2,a[1 + nMax],b[1 + nMax];
  int i,j;

  a[0] = 1.0;
  s2 = r[0];
  c[0] = p[0] / sqrt(s2);
  for(i = 1;i <= n;i++)
  {
    a[i] = 0.0;
```

[5] That is, the MSE error divided by the variance of the signal being approximated: $J_i(u) = \xi_i(u)/\langle y(k), y(k)\rangle$.

```
    k = 0;
    for(j = 0;j < i;j++)
      k += r[i - j] * a[j];
    k /= s2;
    s2 *= 1.0 - k * k;
    for(j = 0;j <= i;j++)
      b[j] = a[j];
    for(j = 0;j <= i;j++)
      a[j] -= k * b[i - j];
    c[i] = 0.0;
    for(j = 0;j <= i;j++)
      c[i] += a[i - j] * p[j];
    c[i] /= sqrt(s2);
  }
}
```

This algorithm can be easily modified to generate the $d_i(a)$ coefficients of the Laguerre lattice filter. It is only necessary to replace the divisions by `sqrt(s2)` with divisions by `s2`.

REFERENCES

[1] Michael L. Honig and David G. Messerschmitt, *Adaptive Filters. Structures, Algorithms, and Applications*, Kluwer Academic Publishers, Boston, 1984.

[2] Bernard Widrow and Samuel D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, 1985.

[3] Simon Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, second edition, 1991.

[4] Ali H. Sayed and Thomas Kailath, "A state-space approach to adaptive RLS filtering", *IEEE Signal Processing Magazine*, vol. 11, no. 3, pp. 18–60, July 1994.

[5] C. Richard Johnson, Jr., "Adaptive IIR filtering: Current results and open issues", *IEEE Transactions on Information Theory*, vol. 30, no. 2, pp. 237–250, Mar. 1984.

[6] John J. Shynk, "Adaptive IIR filtering", *IEEE ASSP Magazine*, vol. 6, no. 2, pp. 4–21, Apr. 1989.

[7] T. Söderström and P. Stoica, "Some properties of the output error method", *Automatica*, vol. 18, no. 1, pp. 93–99, Jan. 1982.

[8] David G. Messerschmitt, "A class of generalized lattice filters", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 2, pp. 198–204, Apr. 1980.

[9] Y. W. Lee, "Synthesis of electric networks by means of the Fourier transforms of Laguerre's functions", *Journal of Mathematics and Physics*, vol. XI, pp. 83–113, 1933.

[10] Y. W. Lee, *Statistical Theory of Communication*, John Wiley and Sons, New York, 1960.

[11] D. G. Lampard, "A new method of determining correlation functions of stationary time series", *The Proceedings of the Institution of Electrical Engineers*, vol. 101, Part III, pp. 343–346, 1954, Monograph No. 104.

[12] J. W. Head, "Approximation to transients by means of Laguerre series", *Proceedings of the Cambridge Philosophical Society*, vol. 52, pp. 640–651, 1956.

[13] G. J. Clowes, "Choice of the time-scaling factor for linear system approximations using orthonormal Laguerre functions", *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 487–489, Oct. 1965.

[14] Kenneth Steiglitz, "Rational transform approximation via the Laguerre spectrum", *Journal of the Franklin Institute*, vol. 280, no. 5, pp. 387–394, Nov. 1965.

[15] J. J. King and T. O'Canainn, "Optimum pole positions for Laguerre-function models", *Electronics Letters*, vol. 5, no. 23, pp. 601–602, Nov. 1969.

[16] Martin Schetzen, "Power-series equivalence of some functional series with applications", *IEEE Transactions on Circuit Theory*, vol. 17, no. 3, pp. 305–313, Aug. 1970.

[17] Martin Schetzen, "Asymptotic optimum Laguerre series", *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 493–500, Sept. 1971.

[18] T. W. Parks, "Choice of time scale in Laguerre approximations using signal measurements", *IEEE Transactions on Automatic Control*, vol. 16, no. 5, pp. 511–513, Oct. 1971.

[19] Morris J. Gottlieb, "Concerning some polynomials orthogonal on a finite or enumerable set of points", *American Journal of Mathematics*, vol. 60, pp. 453–458, 1938.

[20] Gábor Szegö, *Orthogonal Polynomials*, vol. XXIII, American Mathematical Society Colloquium Publications, fourth edition, 1975.

[21] Paul W. Broome, "Discrete orthonormal sequences", *Journal of the Association for Computing Machinery*, vol. 12, no. 2, pp. 151–168, Apr. 1965.

[22] R. E. King and P. N. Paraskevopoulos, "Digital Laguerre filters", *Circuit Theory and Applications*, vol. 5, pp. 81–91, 1977.

[23] Bruno Marone and Biagio Turchiano, "Laguerre z-transfer function representation of linear discrete-time systems", *International Journal of Control*, vol. 41, no. 1, pp. 245–257, 1985.

[24] Bo Wahlberg, "System identification using Laguerre models", *IEEE Transactions on Automatic Control*, vol. 36, no. 5, pp. 551–562, May 1991.

[25] Svante Gunnarsson and Bo Wahlberg, "Some asymptotic results in recursive identification using Laguerre models", *International Journal of Adaptive Control and Signal Processing*, vol. 5, pp. 313–333, 1991.

[26] Mohammad A. Masnadi-Shirazi and N. Ahmed, "Optimal Laguerre networks for a class of discrete-time systems", *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 2104–2108, Sept. 1991.

[27] Y. Fu and G. Dumont, "An optimum time scale for discrete Laguerre network", *IEEE Transactions on Automatic Control*, vol. 38, no. 6, pp. 934–938, June 1993.

[28] Y. Fu and G. A. Dumont, "On determination of Laguerre filter pole through step or impulse response data", in *Preprints of the 12-th IFAC World Congress*, July 1993, pp. 5:303–307.

[29] Tomás Oliveira e Silva, "On the equivalence between Gamma and Laguerre filters", in *1994 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Apr. 1994, pp. IV:385–388.

[30] Tomás Oliveira e Silva, "Optimality conditions for truncated Laguerre networks", *IEEE Transactions on Signal Processing*, vol. 42, no. 9, pp. 2528–2530, Sept. 1994.

[31] Ü. Nurges, "Laguerre models in problems of approximation and identification of discrete systems", *Automation and Remote Control*, vol. 48, pp. 346–352, 1987.

[32] C. Zervos, P. R. Bélanger, and G. A. Dumont, "On PID controller tuning using orthonormal series identification", *Automatica*, vol. 24, no. 2, pp. 165–175, 1988.

[33] Guoxiang Gu, Pramod P. Khargonekar, and E. Bruce Lee, "Approximation of infinite-dimensional systems", *IEEE Transactions on Automatic Control*, vol. 34, no. 6, pp. 610–618, June 1989.

[34] Guy A. Dumont, Christos C. Zervos, and Gerry L. Pageau, "Laguerre-based adaptive control of pH in an industrial bleach plant

extraction stage", *Automatica*, vol. 26, no. 4, pp. 781–787, 1990.

[35] Pertti M. Mäkilä, "Approximation of stable systems by Laguerre filters", *Automatica*, vol. 26, no. 2, pp. 333–345, 1990.

[36] P. M. Mäkilä, "Laguerre series approximation of infinite dimensional systems", *Automatica*, vol. 26, no. 6, pp. 985–995, 1990.

[37] Jonathan R. Partington, "Approximation of delay systems by Fourier-Laguerre series", *Automatica*, vol. 27, no. 3, pp. 569–572, 1991.

[38] Osvaldo Agamennoni, Eduardo Paolini, and Alfredo Dasages, "On robust stability analysis of a control system using Laguerre series", *Automatica*, vol. 28, no. 4, pp. 815–818, 1992.

[39] P. R. Bélanger, O. Arafat, M. Gaber, S. Gendron, and D. Vurgaft Cherson, "Direct performance optimization using Laguerre models", *Automatica*, vol. 30, no. 5, pp. 883–886, May 1994.

[40] L. Wang and W. R. Cluett, "Optimal choice of time-scaling factor for linear system approximations using Laguerre models", *IEEE Transactions on Automatic Control*, vol. 39, no. 7, pp. 1463–1467, July 1994.

[41] A. C. den Brinker, "Adaptive modified Laguerre filters", *Signal Processing*, vol. 31, pp. 69–79, 1993.

[42] A. C. den Brinker, "Calculation of the local cross-correlation function on the basis of the Laguerre transform", *IEEE Transactions on Signal Processing*, vol. 41, no. 5, pp. 1980–1982, May 1993.

[43] Albertus C. den Brinker, "Laguerre-domain adaptive filters", *IEEE Transactions on Signal Processing*, vol. 42, no. 4, pp. 953–956, Apr. 1994.

[44] K. Steiglitz, "The equivalence of digital and analog signal processing", *Information and Control*, vol. 8, pp. 455–467, 1965.

[45] Alan V. Oppenheim and Ronald W. Schafer, *Digital Signal Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1975.

[46] Nicholas Young, *An Introduction to Hilbert Space*, Cambridge University Press, Cambridge, UK, 1988.

[47] Boaz Porat, *Digital Processing of Random Signals – Theory and Methods*, Prentice–Hall, 1994.

[48] Walter Rudin, *Real and Complex Analysis*, McGraw–Hill Book Company, New York, third edition, 1987.

[49] Gene H. Golub and Charles F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, second edition, 1989.

[50] Lennart Ljung, *System Identification. Theory for the user*, Prentice Hall, Englewood Cliffs, NJ 07632, 1987.

[51] Ulf Grenander and Gabor Szegö, *Toeplitz forms and their applications*, Chelsea Publishing Company, New York, second edition, 1984.

[52] Robert Mollen Gray, "On the asymptotic eigenvalue distribution of Toeplitz matrices", *IEEE Transactions on Information Theory*, vol. 18, no. 6, pp. 725–730, Nov. 1972.

[53] Philip J. Davis, *Interpolation and Approximation*, Dover Publications, Inc., New York, 1975.

[54] Tomás Oliveira e Silva, "On the determination of the optimal scale factor of truncated Laguerre networks", *Submitted to IEEE Transactions on Signal Processing*, July 1994.
URL: *ftp://inesca.inesca.pt/pub/tos/English/tspXX.ps.gz*

[55] Tomás Oliveira e Silva, "Optimality conditions for Laguerre lattice filters", *Submitted to IEEE Signal Processing Letters*, Nov. 1994.
URL: *ftp://inesca.inesca.pt/pub/tos/English/splXX.ps.gz*

[56] P. P. Vaidyanathan and Sanjit K. Mitra, "A unified structural interpretation of some well-known stability-test procedures for linear systems", *Proceedings of the IEEE*, vol. 75, no. 4, pp. 478–497, Apr. 1987.

[57] William H. Kautz, "Transient synthesis in the time domain", *IRE Transactions on Circuit Theory*, vol. 1, pp. 29–39, 1954.

[58] P. Lindskog and B. Wahlberg, "Applications of Kautz models in system identification", in *Preprints of the 12-th IFAC World Congress*, July 1993, pp. 5:309–312.

[59] Bo Wahlberg, "System identification using Kautz models", *IEEE Transactions on Automatic Control*, vol. 39, no. 6, pp. 1276–1282, June 1994.

[60] Tomás Oliveira e Silva, "Optimality conditions for truncated second order Kautz networks with two complex conjugate poles", *IEEE Transactions on Automatic Control*, Feb. 1995, (scheduled date of publication).
URL: *ftp://inesca.inesca.pt/pub/tos/English/tac95.ps.gz*

[61] Tomás Oliveira e Silva, *Sobre os filtros de Kautz e sua utilização na aproximação de sistemas lineares invariantes no tempo*, PhD thesis, Universidade de Aveiro, July 1994.

[62] Tomás Oliveira e Silva, "Kautz filters", English translation of a work written in Portuguese for the "Prémio Científico IBM 94", Sept. 1994.
URL: *ftp://inesca.inesca.pt/pub/tos/English/ibm94e.ps.gz*

Construção de Software para Cálculo Matemático

Miguel Oliveira e Silva, Francisco Vaz

Resumo - **Este artigo faz uma abordagem introdutória ao problema da construção de "software" para cálculo matemático, propondo uma técnica de estruturação diferente seguindo o paradigma orientado a objectos. Faz-se um estudo sucinto sobre a estruturação funcional usada "classicamente" nesta área, salientando as suas qualidades e defeitos. É feita uma análise e projecto seguindo uma estruturação orientada a objectos, à qual se segue uma enumeração das exigências colocadas sobre as linguagens de programação que a queiram implementar. No fim refere-se as vantagens e desvantagens desta nova estrutura. Por forma a facilitar a compreensão deste trabalho, faz-se também uma introdução aos paradigmas de construção de "software" aqui referenciados.**

Abstract - **This paper studies the problem of software construction for mathematical applications, and proposes a different structuring technique following the Object-Oriented paradigm. A short study about the "classic" functional structure used in this area is made, pointing out its qualities and drawbacks. An Object-Oriented analysis and design is performed, and from it is enumerated a set of requirements to impose on programming languages to use in implementation. At the end, the main advantages and disadvantages of this new structure are enumerated. To help to understand this work, one section is dedicated to an introduction to the usual software construction paradigms.**

I. INTRODUÇÃO

O desenvolvimento de *software* para cálculo matemático foi a primeira aplicação prática de relevo dos computadores.

Desde o início, quer pelo cariz funcional do cálculo matemático, quer pela estrutura funcional do *hardware* dos próprios computadores e das linguagens que lhe servem de *interface*, a estruturação deste *software* tem assentado no *paradigma funcional*.

Neste artigo vamos fazer um pequeno estudo sobre uma estruturação seguindo o *paradigma orientado a objectos*.

Todo o projecto desta estruturação é feito pensando em facilitar a extensão do *software* quer para novas funções matemáticas, quer para novos tipos numéricos. Nas bibliotecas e aplicações matemáticas "clássicas" é relativamente fácil (na maioria dos casos) a sua extensão com novas funções. É, no entanto, quase impossível a extensão para novos tipos de números.

Começaremos por definir e caracterizar, resumidamente, alguns paradigmas de programação. De seguida far-se-á uma análise sobre a estrutura geralmente usada no software desenvolvido para cálculo matemático. Depois iniciar-se-á o projecto da estruturação orientada a objectos fazendo uma análise sobre o cálculo matemático. Assente nessa análise, lançar-se-á as bases fundamentais para a estrutura do *software*. No fim concluir-se-á este artigo salientando as principais características desta estruturação.

Este artigo assume um conhecimento mínimo sobre engenharia de *software*, nomeadamente no respeitante aos factores de qualidade, princípios e metodologias de programação, aplicáveis ao *software* (ver [1]–[3]).

Não sendo obrigatório, algum conhecimento (básico) sobre análise e projecto orientado a objectos torna mais fácil a leitura deste artigo (ver [2], [4]–[7]).

Uma abordagem bastante mais completa, incluindo uma implementação e respectiva avaliação prática, pode ser encontrada em [1].

II. PARADIGMAS DE PROGRAMAÇÃO

A forma como se analisa, decompõe, compõe, e estrutura um qualquer problema de programação define uma certa filosofia de programação e é usualmente denominada por *paradigma de programação*. Sendo uma classificação necessáriamente subjectiva, existem diferentes noções de paradigmas de programação.

Nesta secção, iremos apresentar as noções dos paradigmas referidos neste artigo. Desta forma espera-se evitar confusões resultantes de diferentes interpretações dos paradigmas enunciados.

A. Funcional

Stroustrup [8] define-o da seguinte forma:

> "Decidir quais os procedimentos que se quer; usar os melhores algoritmos possíveis."

Neste paradigma o elemento básico de estruturação do *software* é a função ou procedimento.

Esta é, tendo em vista o funcionamento dos computadores, a forma mais "natural" e fácil de desenvolver uma linguagem de programação. A linguagem máquina das unidades de processamento segue o mesmo paradigma. Um programa em linguagem máquina é constituido por sequências de instruções, em que cada instrução define uma acção, ou procedimento, a realizar pelo processador.

Desta forma torna-se natural que as primeiras linguagens de programação de "alto nível"[1] tenham seguido a mesma filosofia, facilitando o desenvolvimento de compiladores e interpretadores (tradutores) para a linguagem máquina de cada processador.

[1] O nível duma linguagem é tanto mais alto quanto menos esforço tenha de ser feito para desenvolver aplicações. É uma medida necessariamente subjectiva e relativa entre linguagens de programação.

A grande fraqueza deste paradigma, resulta da dificuldade que tem em lidar (estruturar) com problemas complexos [1], [6].

B. Encapsulamento de Dados

Ao longo dos anos a importância foi passando dos procedimentos para a organização dos dados. Esta situação resultou da constatação de que os dados eram menos sujeitos a mudanças do que os procedimentos. Donde, se a estrutura básica do *software* assentasse nestes, ficava menos sujeita a alterações, por vezes drásticas, tão comuns na sua manutenção.

Aparece assim um novo paradigma, definido da seguinte forma por Stroustrup [8].

> "Decidir quais os módulos que se quer; partir o programa de forma a que os dados sejam escondidos nos módulos."

Um módulo aqui é identificado com um conjunto de procedimentos e funções (serviços), partilhando dados internos.

Este paradigma representa, em termos de resolução de problemas complexos, uma melhoria significativa em relação ao paradigma anterior. Os princípios e critérios de modularidade [2], [1] são cumpridos quase na totalidade.

É mesmo possível, desde que os módulos sejam tipos de dados, implementar tipos de dados abstractos (*Abstract Data Types*) [2], [1], [8].

C. Orientado a Objectos

Segundo um estudo de Lientz [9], estima-se que cerca de 70% do custo do *software* é dispendido na sua manutenção. Nesta fase, cerca de dois quintos devem-se a extensões e modificações requeridas pelos seus utilizadores.

Coloca-se assim o seguinte problema: como construir *software* por forma a facilitar a sua extensão e adaptação?

Nestes aspectos, o paradigma orientado a objectos revoluciona as metodologias de programação. Fazendo uso de um mecanismo - a *herança* - e das duas técnicas a ele associadas - o *polimorfismo* e a *ligação dinâmica* - este paradigma consegue, de uma forma admirávelmente simples, ir de encontro aos princípios da antecipação de mudanças e da incrementabilidade [2], minimizando o problema anterior.

Este novo paradigma assenta directamente sobre o paradigma de encapsulamento de dados, extendendo a sua usabilidade de duas formas. Primeiro permite, por herança, a definição de novos módulos (*Abstract Data Types*) por extensão e/ou ajustamento (*refinement*) de módulos já existentes. Por fim, permite também, por polimorfismo e ligação dinâmica, a substituição (dinâmica) de módulos sem afectar o código do cliente. Estas características permitem um uso verdadeiramente *abstracto* dos tipos de dados, onde os módulos são usados sem necessidade de qualquer conhecimento prévio da sua implementação, podendo mesmo ser usados módulos sem implementação (em tempo de execução, óbviamente terão de ser substituidos por algum módulo implementado), é necessário sim, a definição das *interfaces* ou comportamento, de cada módulo.

Na terminologia da maioria das linguagens orientadas a objectos, a especificação de cada módulo é designada por *classe*, e as entidades que em tempo de execução instanciam as classes são os *objectos*.

A herança deve ser vista como um método de *classificação*, estabelecendo uma relação *é-um* (*is-a*) entre a classe filha e a(s) classe(s) progenitora(s).

C.1 Covariância

Havendo a possibilidade de redefinir serviços herdados de uma classe progenitora, que liberdade dar aos tipos dos seus parâmetros de entrada e, se existir, ao tipo da entidade de retorno?[2]

Das várias soluções possíveis para este problema iremos referir apenas duas delas, por serem, práticamente, as únicas usadas.

A primeira, consiste em permitir que os tipos da assinatura dos serviços redefinidos possam ser alterados *contra* o sentido normal da herança. Esta é a solução conhecida por *contra-variância*.

Esta é a solução preconizada pelo *C++*. Tem a vantagem de facilitar enormemente o trabalho do compilador, pois evita "buracos" no sistema de tipos. Nunca há o risco ao usar a ligação dinâmica, de invocar um serviço com entidades cujos tipos não sejam conformes com os da sua assinatura. A sua desvantagem é ter pouca aplicação prática (o autor ainda não conhece nenhum exemplo prático que leve vantagem pela aplicação desta solução).

A segunda solução, consiste em permitir que os tipos da assinatura dos serviços redefinidos possam ser alterados *no sentido* normal da herança. Esta é a solução conhecida por *covariância*.

Esta é a solução usada pelo *EIFFEL*. A vantagem desta solução reside na sua maior proximidade com os problemas reais. Por exemplo, uma situação prática nitidamente covariante, é a seguinte:

> Os herbívoros comem plantas. As vacas *são* herbívoros. A erva *é* uma planta. As vacas comem erva mas não outras plantas.

A desvantagem da covariância reside na complexidade imposta ao compilador, pois esta solução pode gerar "buracos" no sistema de tipos da linguagem.

C.2 O Problema do Encaminhamento Múltiplo

Inerente ao próprio paradigma orientado a objectos é o envio de mensagens[3] de objectos para outros objectos. O problema que aqui se coloca tem a ver com o encaminhamento das mensagens. Já vimos que a herança fornece um mecanismo privelegiado (ligação dinâmica) para que esse encaminhamento se faça em função do tipo do objecto (em tempo de execução) para o qual queremos enviar a mensagem.

Sendo um mecanismo simples, nem sempre resulta numa aplicação simples e directa de alguns problemas reais.

Vejamos, por exemplo, a operação de soma entre dois números

$$a + b.$$

[2] A este conjunto costuma-se chamar *assinatura* do serviço.

[3] Sejam elas implementadas por procedimentos (*EIFFEL* e *C++*), ou explicitamente por mensagens (*SMALLTALK*).

Qual será o objecto ao qual devemos enviar a mensagem de soma? Ao *a*? Ao *b*? Ou a outro objecto qualquer?

Se formos rigorosos e aplicarmos a semântica matemática envolvida na soma de dois números, não pode haver efeitos colaterais (*side effects*) nem em *a* nem em *b* pela aplicação desta operação. Assim sendo a escolha deve recair sobre a terceira alternativa: a mensagem deve ser enviada para um terceiro objecto.

O problema que aqui queremos discutir é sobre qual o tipo de objectos a atribuir a esse terceiro objecto. O de *a*? O de *b*? Ou um outro tipo qualquer?

Se *a* e *b* forem do mesmo tipo a solução é simples: o tipo do terceiro objecto deve ser o mesmo.

Se *a* e *b* forem de tipos diferentes (expressão heterogénea) colocam-se três situações: ou *a* é conforme[4] com *b*; ou *b* é conforme com *a*; ou nenhuma das situações anteriores.

Na terceira hipótese, coloca-se a questão da validade da aplicação da própria operação de soma (que sentido tem somar dois tipos de números não relacionados?). Esta é uma situação que, em princípio, deve ser rejeitada pelo sistema de tipos da linguagem (em tempo de compilação ou em tempo de execução).

Para as outras duas hipóteses, se observarmos com algum cuidado, a solução também é imediata: o tipo do terceiro objecto deve ser o tipo do objecto mais genérico dos dois (o tipo do objecto "pai").

Vejamos um exemplo. Se somarmos um número real com um número inteiro, o resultado tem de ser evidentemente um número real (tipo mais genérico).

Esta solução resulta directamente de relação *is-a* imposta pela herança. O tipo de resultado da operação tem de englobar o tipo dos dois objectos aos quais se aplica a operação, donde tem de ser o tipo mais genérico.

C.3 O Problema das Expressões Heterogéneas

Intimamente ligado ao problema anterior existe o problema da implementação dos serviços elementares com pelo menos um argumento, ao qual é necessário ter acesso directo ao seu estado interno.

Não sendo esta uma situação muito frequente, aparece inevitavelmente na implementação dos vários operadores aritméticos, como sejam o operador da soma e da multiplicação.

Voltando ao exemplo da subsecção anterior, vamos supor que queremos implementar o operador soma entre dois números reais. Como é evidente, o serviço relativo ao operador soma terá de ser um serviço elementar, pois não é possível fazer a soma de dois números reais sem ter acesso à sua representação interna.

Se esta situação não traz nenhum problema quando os dois números corresponderem a objectos que sejam instâncias de uma mesma classe, o mesmo já não se passa se um deles pertencer a uma classe descendente diferente, por exemplo, a uma classe de números inteiros. Nesta situação, como implementar este serviço sem obrigar a que essa implementação tenha conhecimento de todas as classes descendentes da classe dos números reais?

[4] Descendente.

A solução só pode ser uma, tem de ser possível converter o estado interno dos objectos das classes descendentes para o estado interno dessa classe, no caso, para a classe dos números reais. Aqui a solução ideal talvez passasse por um mecanismo de coerção (*casting*) da linguagem de programação, definível para cada classe, aplicado automaticamente pelo compilador sempre que necessário. Infelizmente, nenhuma das linguagens de programação conhecidas pelo autor implementa este mecanismo, nem oferece qualquer solução para este problema (*C++*, *EIFFEL*, *SMALLTALK*, etc.). Nestas linguagens, ou se limita o uso de expressões matemáticas a expressões homogéneas (só com objectos de um só tipo), ou se implementam as classes descendentes mantendo o estado interno da classe pai, alterando-o (sempre que possível) paralelamente ao novo estado da classe filha (pode-se dizer que, nesta situação, a coerção é feita sistematicamente em cada alteração do estado interno da classe descendente).

III. ESTRUTURAÇÃO "CLÁSSICA"

Em geral, dois tipos de soluções têm sido usadas para facilitar o desenvolvimento do *software* para grupos específicos de problemas: *especialização* de linguagens de programação; e o desenvolvimento de *bibliotecas* de *software* em linguagens de programação de aplicação geral, para este domínio de problemas.

A primeira solução, faz com que as entidades manuseáveis da linguagem de programação se aproximem das entidades e termos usados nessa área de aplicação, do que resulta uma maior simplicidade e compreensibilidade do *software*. Esta solução facilita o desenvolvimento de ambientes e ferramentas altamente especializados para a resolução deste tipo de problemas, fazendo com que, em geral, se obtenham tempos de desenvolvimento baixos.

Como exemplos de aplicações seguindo esta solução na área do cálculo matemático temos: o MATLAB e o MATHEMATICA.

Comum à maioria destas aplicações é o facto de assentarem em interpretadores das suas linguagens, do que resulta, em princípio, uma perda do desempenho.

Outro aspecto comum a estas aplicações (pelo menos as mencionadas) é o facto de assentarem numa estrutura funcional, onde os módulos do sistema são funções e dados pré-definidos (reais, complexos, etc.). Daqui resulta de imediato uma coesão forte [2] entre os dados e as funções que operam sobre eles.

Estas características fazem com que este tipo de aplicações sejam ideais para *prototipagem*.

A segunda solução, além de em princípio levar a um melhor desempenho, tem outras vantagens importantes: são mais facilmente integráveis em aplicações maiores (compostas não só por cálculo matemático) e, em princípio, são mais facilmente extensíveis para novos problemas não previstos nos sistemas especializados.

Como exemplo de uma biblioteca seguindo esta solução temos a *NAG*, desenvolvida para a linguagem *FORTRAN*. Inúmeras outras bibliotecas existem para outras linguagens como sejam o *C* e o *C++*. A *NAG* surge no entanto como

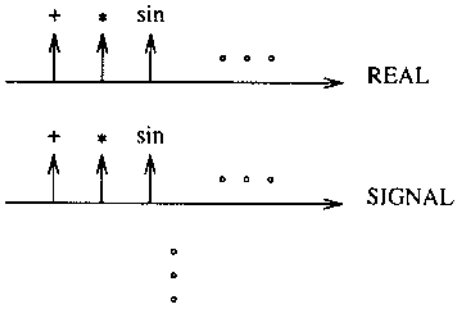


Fig. 1 - Estruturação funcional.

referência essencial pelo seu uso generalizado nesta área.

Estas características fazem com que estes sistemas sejam mais indicados para desenvolvimento de *aplicações em grande escala*, ou quando o desempenho é um factor importante.

Fazendo um apanhado sobre algumas das bibliotecas e aplicações existentes para cálculo matemático, constata-se que a estruturação base que estas usam também assenta no paradigma funcional.

Os problemas deste tipo de estrutura, nesta área, são essencialmente dois. Em primeiro lugar, como já foi referido, há uma coesão forte entre os tipos numéricos e as funções que os operam. Assim, por exemplo, o uso da função *seno* não pode ser visto como o cálculo do seno de um número real, mas sim como o cálculo do seno de uma (única) implementação particular de números reais. Esta característica dificulta enormemente a extensão e a adaptação da biblioteca a novos números.

O segundo problema assenta na não classificação dos vários tipos de números e funções. Por exemplo, um número real *é* um número complexo, assim como um número inteiro *é* um número real, estas relações estabelecem claramente uma hierarquia entre estes tipos de números, no entanto essa hierarquia não é, de forma alguma, implementada nesta estrutura. Se na aparência este problema não parece ser particularmente importante, na prática pode ter implicações profundas. Por exemplo, pelo facto de um número real ser um complexo, é perfeitamente válido fazer a sua soma com um número complexo, obtendo-se sempre como resultado um outro número complexo. Pelo contrário, a soma de um real com um complexo, em geral não tem como resultado um número real. Não havendo uma hierarquização destes tipos numéricos, como garantir estas regras?

Classicamente, estas regras de implementação de expressões heterogéneas, são definidas de uma forma *AD HOC*. Desta forma torna-se muito difícil extender estas regras a novos tipos de números. Outra consequência negativa desta não classificação, é a menor compreensibilidade do *software*, o que pode afectar a sua fiabilidade.

A grande vantagem deste tipo de estrutura (pelo menos nas bibliotecas de *software*) é o seu excelente desempenho. Essa eficiência deve-se, óbviamente, à relação muito directa feita por este *software* com os recursos disponibilizados pelo *hardware* dos computadores.

IV. CÁLCULO MATEMÁTICO

Um dos pilares duma boa estruturação orientada a objectos consiste em assentar a estrutura de classes do *software* nas abstracções essenciais da área do problema que se está a resolver. Impõe-se assim, um estudo e caracterização das entidades e conceitos com que se lida no Cálculo Matemático em geral.

Comum a todo o cálculo numérico é a necessidade de lidar com entidades representando quantidades e entidades que calculam, objectiva e rigorosamente, novas quantidades através de regras, leis e axiomas formais.

As entidades que representam quantidades são designadas pelo conceito abstracto de *número*. As entidades que transformam ou calculam números são designadas por *funções* ou *operadores*. Associadas a estas entidades existe um conjunto de *propriedades* que caracterizam o comportamento dessas entidades (a sua semântica).

A. Números

Comecemos pelos números. O que é que os caracteriza e define? Antes de mais, cada tipo de números implementa uma forma particular de representar "quantidades". Por exemplo, para representar quantidades discretas e unidimensionais usam-se os números inteiros, para quantidades também unidimensionais mas contínuas usam-se os números reais, etc. Além desta característica óbvia, a cada tipo de números pode-se aplicar uma infinidade de operadores e funções, do que resultam novos números (não necessariamente do mesmo tipo). É o caso, por exemplo, dos operadores aritméticos básicos atrás referidos, das funções trignométricas, etc. Da semântica (comportamento) desses operadores ou funções é possível extrair um conjunto de propriedades sempre observáveis na aplicação dos mesmos (comutatividade da soma, etc.).

Associadas a cada tipo de números existe um conjunto de funções ou operadores básicos que permitem caracterizar as propriedades básicas de cada tipo de números. Por exemplo, é esse o caso dos operadores de soma, multiplicação, igualdade, e da relação de ordem dos números reais.

B. Funções

As funções (e operadores), servem para processar números. Apesar de estarem sempre ligados a estes, podem ter uma identidade muito para além de um determinado tipo de número. É o caso, por exemplo, do operador soma, aplicável a uma enorme variedade de tipos de números, como sejam os números inteiros, reais, complexos, etc. Essa identidade refere-se, não só à aplicação da mesma função a vários tipos de números, mas também, no caso geral, pela verificação das mesmas propriedades no seu uso. A comutatividade do operador soma, por exemplo, verifica-se quer seja soma de inteiros, complexos ou mesmo matrizes. Já o caso do operador *maior-do-que*, restringe o tipo de números a que é aplicável: estes têm de ser ordenáveis.

Na maioria dos casos a definição de novas funções é feita à custa de outras já existentes. Por exemplo, a soma de matrizes é definida à custa do produto e da soma dos seus elementos, quer estes sejam números inteiros, reais ou comple-

xos. Neste exemplo, vemos mesmo que o algoritmo para o cálculo dessa operação é o mesmo para todos esses tipos de números.

C. *Propriedades*

As propriedades servem para disciplinar o uso de funções e números uns com os outros. Funcionam assim para validar o uso da função, ou número, no contexto onde são usados. Por exemplo a propriedade *ordenável* de um determinado tipo de número serve para validar o uso da função *maior-do-que* a entidades (ou instâncias) nesse tipo de número.

É notória, a similaridade entre o uso das propriedades em matemática e o uso de um sistema de tipos em linguagens de programação. Tal como as propriedades matemáticas, a atribuição de tipos aos objectos duma linguagem orientada a objectos, tem o objectivo de validar a utilização desses objectos no contexto onde são usados. Desta constatação resulta a implementação óbvia de cada propriedade matemática numa linguagem orientada a objectos. Associar cada propriedade matemática a um tipo (classe) de objectos distinto. Desta forma, para atribuir propriedades a cada tipo de números ou função basta fazer com que esses números ou funções herdem das propriedades que os caracterizam.

Infelizmente, embora se possa sempre associar uma classe a cada tipo de propriedade, nem todas as propriedades são facilmente implementáveis por classes. Como exemplos das duas situações temos as propriedades da *ordenabilidade* e da *comutatividade*. A ordenabilidade é facilmente implementável, basta definir para a respectiva classe os serviços associáveis às relações de ordem: maior; menor; igual; menor ou igual; e maior ou igual (estas duas últimas podem ser obtidas das três anteriores). Assim só as classes que herdem da classe da ordenabilidade podem fazer uso desses serviços. O caso da comutatividade é, no entanto, completamente diferente. A validação desta propriedade não se faz pelo uso de serviços distintos, mas sim apenas pela semântica no uso de serviços já existentes. Na prática, implementar esta propriedade tornar-se-ia pesado pois obrigaria, para cada operação de soma efectuada, a efectuar a mesma operação com os argumentos trocados e a verificar se o resultado era igual. Teríamos assim um *overhead* excessivo por cada operação efectuada, o que tornaria o *software* necessariamente bastante ineficiente.

Em rigor, este problema é também transportável para a ordenabilidade (e em geral para qualquer outra propriedade matemática), pois o facto de termos as operações das relações de ordem definidas não nos garante que funcionem adequadamente, conforme o sentido semântico matemático que a elas associamos.

Aqui apresentam-se duas alternativas. Ou se tenta garantir ao máximo a observância estrita de todas as propriedades matemáticas usadas, o que obrigaria à execução de inúmeros testes por cada uso de funções e números. Ou se tenta limitar a verificação das propriedades aos casos que não ponham em risco o desempenho do *software*.

A escolha recai obviamente pela segunda alternativa, pois o objectivo do *software* é o seu uso na prática, e não o garantir a correcção absoluta na utilização das entidades matemáticas.

D. *Representação de Números*

Um dos problemas graves que se coloca na implementação de números em computadores, é o problema da sua representação ter de ser feita por um número finito de estados. Esta situação leva a que possa haver efeitos colaterais "inesperados" na aplicação de funções a esses números. Esta situação pode gerar perdas de precisão e saturação no processamento de números, afectando a fiabilidade de todo o *software*.

Se pouco ou nada se pode fazer em termos de implementar números sem limites máximos ou mínimos e de precisão infinita, pode-se, no entanto, não obrigar ao uso de uma única representação para cada tipo de números usados. Ou seja, arranjar uma estrutura do *software*, em que se possam implementar novas funções de processamento numérico sem que estas obriguem ao uso de um único tipo de dados.

Esta situação reforça as conclusões do estudo atrás feito sobre as vantagens de haver independência entre os números e as funções que operam sobre eles.

Desta forma pode-se desenvolver destemidamente novas função de cálculo numérico, sem o risco de mais tarde elas terem de ser reimplementadas só pelo facto de a representação dos tipos numéricos usada ser insuficiente (ou mesmo boa demais, limitando o desempenho) para as exigências colocadas sobre o *software* de cálculo desenvolvido.

E. *Funções e Operadores Elementares*

Este problema da representação de números em computadores, salienta outra questão importante. Cada tipo de números em particular, terá, como é óbvio, de lidar com o tipo de representação usada para o mesmo. Isso só pode ser feito ligando fortemente um certo conjunto básico de operadores e funções a esse tipo de números, por forma a que, por exemplo, a soma de 3 com 5 dê 8 em cada tipo de números inteiros existentes.

Tem assim de haver um conjunto de funções dependentes de cada tipo numérico existente. Essas funções ditas elementares têm de fazer parte da especificação de cada tipo numérico, e é só fazendo uso delas que se pode processar números.

Os critérios de escolha entre fazer com que uma função faça parte, ou não, da ADT[5] de cada tipo numérico prendem-se essencialmente com duas situações: ou a sua implementação depende da representação interna do número, ou por questões de desempenho.

F. *Excepções Numéricas*

Qual é o comportamento que o *software* deve ter se houver uma divisão por zero, ou se a multiplicação de dois números não poder ser representada nesses tipos numéricos?

Esta é uma situação razoavelmente frequente, que resulta de, por vezes, não ser possível efectuar correctamente um

[5] *Abstract Data Type.*

qualquer cálculo numérico. As excepções matemáticas surgem sempre da impossibilidade de expressar com correcção o resultado numérico de uma qualquer função.

Podem-se identificar duas origens para esta situação. A primeira diz respeito a questões puramente matemáticas, como é o caso de se tentar usar uma função com valores que não pertencem ao seu domínio. A segunda, resulta de limitações na representação de números em computadores, podendo gerar problemas de precisão ou de saturação nos cálculos.

Não prever, ou achar-se irrelevantes, este tipo de situações é de todo inaceitável, pois além de elas serem relativamente frequentes, nem sempre exprimem erros no *software* desenvolvido. O exemplo clássico desta situação é o da inversão de matrizes. Esta operação só é válida se as matrizes não forem singulares (determinante diferente de zero), no entanto, o algoritmo que testa se uma matriz é singular é igual ao que faz a inversão. É assim preferível tentar a inversão e lidar com uma eventual excepção de singularidade *a posteriori*.

Estas duas situações levam também a que não possa haver uma única forma de lidar com excepções. Por vezes deve-se indicar ao programador um erro no *software* (situação normal). Outras vezes deve-se permitir que se efectue um processamento especial qualquer na sua existência. Pior que isso, não compete à própria função onde a excepção se deu, lidar com essa situação, pois uma mesma excepção pode corresponder a causas muito diferentes, podendo o tratamento a dar às mesmas ser completamente diverso. Por exemplo, uma inversão de uma matriz singular pode mostrar, de facto, um erro no *software*, devendo-se nesta situação terminar "suavemente"[6] a aplicação e indicar ao programador a localização do código fonte onde a excepção se deu. Esta mesma situação pode, como já foi referido, resultar de uma tentativa consciente, de inversão da matriz.

Assim, deve competir ao cliente dessa função o tratamento a dar em caso de excepções.

G. *Tipos Numéricos Básicos*

Fundamental numa qualquer biblioteca matemática é a existência dos tipos numéricos básicos, nomeadamente os números inteiros, reais e complexos.

Já vimos que uma qualquer implementação desses tipos numéricos pode levar a problemas por limitações na sua representação. Vimos também o tipo de estruturação que se deve usar para minorar esse problema. Falemos agora um pouco sobre a essência desses números e como se relacionam entre si.

Os números inteiros servem para representar quantidades discretas.

Os números reais representam quantidades contínuas, e, em geral, são implementados por um de dois tipos de representações: vírgula fixa ou vírgula flutuante.

Como o conjunto dos números reais contém o conjunto dos números inteiros, a biblioteca deve permitir o uso de um número inteiro em substituição de um número real, não devendo, em princípio, permitir a situação inversa.

Os números complexos são definidos à custa de uma quantidade abstracta designada por j, cujo valor é igual a $\sqrt{-1}$, e à custa dos números reais. Para estes números há também duas representações frequentes: cartesiana e polar.

[6] Sem mais eventos catastróficos.

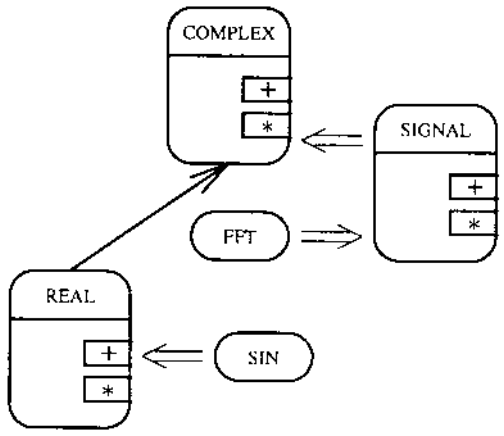


Fig. 2 - Estruturação Orientada a Objectos.

Como os números complexos se implementam à custa dos números reais, em princípio, a biblioteca só precisa de uma implementação deste tipo de números ligada a um tipo real abstracto redefinível para outros quaisquer tipos reais (incluindo os próprios números inteiros).

Estando o conjunto dos números reais contido no conjunto dos números complexos, a biblioteca deve permitir, como acontecia com os números inteiros em relação aos números reais, o uso de números reais em substituição de números complexos. Já o uso de números complexos como reais, não deve ser permitido, já que só sob certas condições (parte imaginária nula) é que um número complexo pode ser visto como real, o que tornaria, pela complexidade envolvida, muito difícil, e mesmo pouco desejável, a implementação de um tal mecanismo.

V. ESTRUTURAÇÃO ORIENTADA A OBJECTOS

Pegando na análise da secção anterior, vamos agora projectar a estrutura básica do *software*, baseada no paradigma orientado a objectos.

Uma estruturação orientada a objectos deve assentar em relações comportamentais entre os vários módulos da biblioteca. Isso é feito recorrendo a *classes* para descrever os vários módulos e ao uso da *herança* para atribuição dos comportamentos associados a cada classe.

Como vimos na secção anterior, as abstracções base do cálculo matemático são: *número*, *função* e *propriedade*. Duas delas são generalizáveis para abstracções mais simples (e uteis): uma *função* pode ser vista como um caso particular de um *sistema*, e um *número* é uma espécie de *dados* (memória).

A abstracção *propriedade* parece ser suficientemente geral. Para se conseguir que as propriedades validem o uso de números e funções faz-se uso do sistema de tipos das linguagens orientadas a objectos, ou seja, por herança.

Por forma a facilitar o armazenamento heterogéneo de quaisquer objectos da biblioteca (e não só), torna-se útil definir uma classe progenitora de todas as classes da biblioteca. Convencionou-se chamar ANY a essa classe.

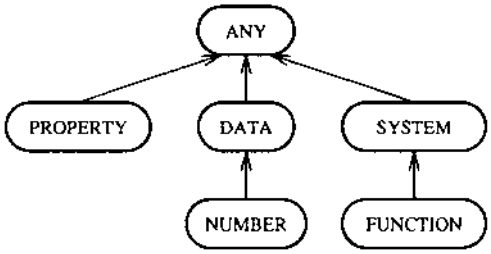


Fig. 3 - Classificação básica da biblioteca

A. *Classificação básica*

Iremos assim assentar toda o *software* em três classes base: PROPERTY, DATA e SYSTEM. Dessas classes base a classe NUMBER herda da classe DATA e a classe FUNCTION herda da classe SYSTEM.

Nesta estruturação surge uma particularidade curiosa e aparentemente paradoxal, há uma classe que abstrai o conceito de função!

Meyer [2] num capítulo sobre técnicas de projecto orientado a objectos, descreve aquilo que designa de *o grande erro* no projecto de classes:

> "O papel fundamental do projecto orientado a objectos é o de construir módulos à volta de tipos de objectos, não de funções. (...)"

Sendo esta afirmação verdadeira na quase totalidade das situações, falha neste exemplo em particular, pois o que se pretende abstrair é precisamente o conceito de função (o próprio Meyer [10] identifica uma situação análoga, a classe COMMAND pertencente ao grupo das classes de *interface* com o utilizador). Um pouco mais à frente nesse livro, ele generaliza o problema dando-lhe, na opinião do autor, o contexto adequado:

> "(..) Este erro é fácil de evitar uma vez estando consciente do risco. O remédio é, como era anteriormente, garantir que cada classe corresponde a uma abstracção de dados com sentido."

Nesta situação, sem dúvida que a abstracção de função é uma abstracção com sentido, pois é uma das entidades fundamentais com que se lida em matemática.

A validação de propriedades, por questões de desempenho e simplicidade, é feita usando o sistema de tipos da linguagem de programação (ou seja através da herança).

B. *Tipos Numéricos Básicos*

Pretende-se projectar classes para os tipos de números básicos, de tal forma que seja possível redefinir qualquer um desses tipos sem afectar o resto da biblioteca, e possibilitando que um número real substitua um complexo, e que um inteiro substitua qualquer um desses dois.

A solução é clara, tornar a classe dos números reais herdeira da classe dos números complexos, e fazer o mesmo para a classe dos números inteiros em relação à classe dos números reais. É necessário também garantir que há redefinição dos operadores da classe dos complexos pelos operadores da classe dos reais, assim como para a classe dos inteiros.

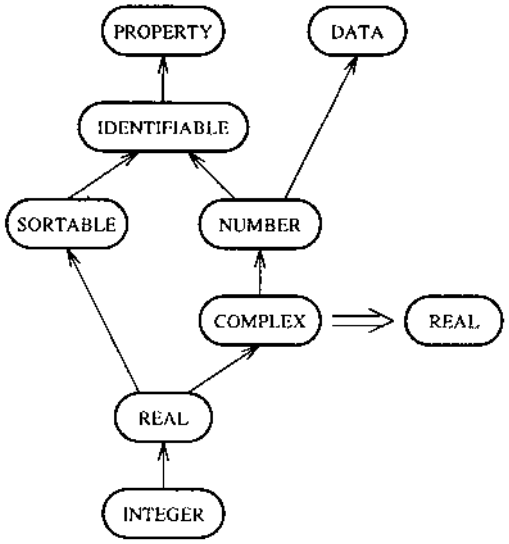


Fig. 4 - Classes numéricas básicas.

A capacidade de substituir os tipos numéricos mais genéricos por tipos numéricos mais especializados, obriga a uma redefinição covariante dos seus operadores comuns. Por exemplo, a soma de dois números complexos é um número complexo, assim como a soma de números reais resulta também num número real, logo este operador tem de ser, obrigatoriamente, redefinido de uma forma covariante.

Apesar deste processo de especialização desde os números complexos até aos números inteiros, todos os serviços dos números complexos continuam a ser aplicáveis, com resultados apropriados aos novos números, nos seus descendentes. Assim continua a ser válida uma invocação ao serviço de conjugação, por exemplo, não sendo, no caso, tomada nenhuma acção uma vez que o conjugado de um número inteiro ou real, é igual ao próprio.

C. *Expressões Aritméticas*

Como resultado das exigências colocadas na estrutura dos tipos numéricos básicos pode acontecer não ser possível usar directamente os tipos numéricos básicos postos à disposição pela linguagem de programação. Essa eventualidade, quando somada à exigência de covariância, pode inviabilizar o uso de expressões aritméticas na sua forma mais natural, ou seja usando directamente os operadores aritméticos.

Se à partida isso pode não parecer muito importante, pois o uso de expressões aritméticas não é mais do que uma forma alternativa de invocar funções numa linguagem de programação, as consequências resultantes da sua não utilização são mais sérias do que isso.

Em primeiro lugar resulta numa menor compreensibilidade do *software*, logo, temos uma menor fiabilidade do *software* que é, evidentemente, o factor de qualidade mais importante a ter em consideração num produto de *software*.

A menor compreensibilidade que esta situação traz, é bem visível nas duas implementações (em pseudo-código) que

em seguida se faz da convolução de dois sinais.

```
(1)-> from k = h.low to h.high loop
        tmp->assign(x[k]);
        tmp->rmult(h[n - k]);
        y[n]->radd(tmp);
      end;

(2)-> from k = h.low to h.high loop
        y[n] = y[n] + x[k] * h[n - k];
      end;
```

Sendo a simplicidade um dos pilares essenciais de uma estruturação orientada a objectos, seria incoerente obrigar os utilizadores do *software* a usarem uma notação tão pouco natural para implementarem expressões matemáticas.

A implementação de expressões aritméticas respeitando as exigências colocadas sobre os tipos numéricos básicos tem também os problemas decorrentes do encaminhamento múltiplo e das expressões heterogéneas.

D. Exigências sobre a Linguagem de Programação

Nesta subsecção vamos agrupar os requesitos necessários ou aconselháveis, da linguagem de programação a usar para implementação de uma biblioteca para cálculo matemático seguindo esta estruturação.

Alguns destes requesitos devem ser fornecidas pela própria linguagem de programação a utilizar, outras podem ser implementadas sobre a mesma (irá depender evidentemente, das facilidades postas à disposição pela linguagem).

- Herança, polimorfismo e ligação dinâmica.
- Herança múltipla (ver [1]).
- Sistema de tipos estático (ver [1]).
- Covariância.
- Programação por contrato (ver [1]).
- Tipos numéricos básicos implementados como classes.
- Relações entre tipos numéricos básicos como descrito na figura 4.

VI. CONCLUSÕES

Como conclusões deste pequeno estudo sobre uma estruturação orientada a objectos de *software* para cálculo matemático, pode-se dizer o seguinte:

- É óbvio que a grande vantagem prática desta estruturação é a sua extensibilidade a novos tipos numéricos. Por acréscimo, congrega também as vantagens inerentes às técnicas orientadas a objectos, embora estas só sobresaiam caso o *software* a construir seja suficientemente complexo.
- A desadequação de todas as linguagens de programação orientadas a objectos existentes. Aqui a que mais se aproxima é a linguagem *EIFFEL*, falhando na hierarquia que impõe aos tipos numéricos básicos, e a forma como lida com o encaminhamento múltiplo e as expressões heterogéneas.
- Por fim, e na sequência do ponto anterior, implementações de uma biblioteca com esta estrutura numa linguagem de programação existente, leva necessariamente a um mau desempenho. A razão principal deve-se ao desconhecimento pelo compilador dos tipos numéricos básicos, impedindo optimizações dos cálculos matemáticos usando esses tipos.

Esta estruturação foi testada numa biblioteca denominada *Calculus Object Oriented Library* (COOL), implementada sobre a linguagem *C++*, e também numa linguagem de utilização dessa biblioteca designada por *COOL Usage Language* (CUL). Ambas as ferramentas foram desenvolvidas na tese de mestrado do autor (ver [1]).

AGRADECIMENTOS

Este trabalho teve o apoio da bolsa de mestrado da JNICT no âmbito do programa CIÊNCIA: BM/1733/91-IA

REFERENCES

[1] Miguel Oliveira e Silva, "Biblioteca de software para cálculo matemático e processamento digital de sinal", Master's thesis, Departamento de Electrónica e Telecomunicações – Universidade de Aveiro, Sept. 1994.

[2] Bertrand Meyer, *Object-Oriented Software Construction*, Prentice-Hall, Englewood Cliffs, N.J., 1988.

[3] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli, *Fundamentals of Software Engineering*, Prentice-Hall, 1991.

[4] Peter Coad and Edward Yourdon, *Object Oriented Analysis*, Prentice-Hall, 1990.

[5] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.

[6] Grady Booch, *Object Oriented Design with Applications*, The Benjamin/Cummings Series in Ada and Software Engineering. The Benjamin/Cummings Publishing Company, Inc., 1991.

[7] Sally Shlaer and Stephen J. Mellor, *Object-Oriented Systems Analysis – Modeling the World in Data*, Prentice-Hall, 1988.

[8] Bjarne Stroustrup, "What is object-oriented programming?", *IEEE Software*, pp. 10–20, May 1988.

[9] B.P. Lientz and E.B. Swanson, "Software maintenance: A user/management tug of war", *Data Management*, pp. 26–30, Apr. 1979.

[10] Bertrand Meyer, *Reusable Software: The Base Object-Oriented Component Libraries*, The Object-Oriented Series. Prentice-Hall, 1994.

User Interface for a Decision Support System Based on Factor Analysis

Carlos Rui Carvalhal[1], Beatriz Sousa Santos[1], Carlos Ferreira[2], José Alberto Rafael[1]
[1]Departamento de Electrónica e Telecomunicações, Universidade de Aveiro
[2]Departamento de Matemática, Universidade de Aveiro

Resumo - **Neste artigo descreve-se o interface de utilizador de um Sistema de Apoio à Decisão baseado em Análises Factoriais, nomeadamente: Componentes Principais, Discriminante, Correspondências e Quadros de Distâncias. O sistema foi desenvolvido para Windows usandoVisual Basic TM 3.0.**

Abstract- **A Decision Support System based on Factor Analysis was designed to offer the user the possibility to perform easily four types of Factor Analysis: Principal Component, Discriminant, Correspondence and Distance. We will describe the user interface of such a system, developed for Windows using Visual Basic TM 3.0.**

I. INTRODUCTION

The purpose of Data Analysis is to distinguish, in large data sets, what is essential from what is accessory. It can be used whenever it is important to simplify and reduce the complexity of a problem and it is founded on the trade-off between loosing information and gaining significance.

The system developed can be used to perform Factor Analysis [1] producing synthetic representations of large numerical data sets. Thus, it is possible to represent the information using a reduced set of parameters enhancing relevant features hidden in the raw data set, and offering to the user the possibility to control, along the analysis, the progression of the error (between the raw data set and its representation).

The DSSBFA *(Decision Support System Based on Factor Analysis)* offers the user an interactive decision support working environment based on four different Factor Analysis, meant for the analysis of four types of information:

- *Principal Component Analysis* -PCA, to study quantitative data sets, all of similar nature
- *Correspondence Analysis* -CrA, to study quantitative data sets, of different nature
- *Discriminant Analysis* -DcA, to study the relationships among a qualitative variable and several quantitative variables
- *Distance Analysis* -DtA, to study distance data sets.

The success of such a system depends heavily on the type of interaction and presentation of the results. That is why we have used this problem as a practical exercise of a course on Human Computer Interface during the academic year of 93/94. The goal was to develop a user interface as friendly as possible and this should be accomplished by a group of 7 under-graduated students co-ordinated by a MSc Student, having a mathematician as consultant.

The system was developed for a Windows platform using Visual Basic TM 3.0 [2] and it has an open architecture allowing an easy integration of new methods, auxiliary procedures or new ways of displaying results.

In the next sections we will describe briefly the user's profile as well as the system architecture. We will go into further detail on the user interface.

II. USER'S PROFILE AND TASK CHARACTERISTICS

A clear ideia of the user and task characteristics is fundamental for the design of a user interface [2]. The target user population of this system consists of mathematicians working mainly in enterprises, but also possibly in research environments. A simple study of this population allows us to conclude that the prioritary users of this system will have the following profile:

- Low to medium computer literacy;
- Low to medium motivation;
- Low to medium experience with system;
- Little or no training;
- Discretionary use;
- Low application experience;
- Moderate task experience;
- Structured task.

These characteristics of the user and task lead us to conclude that this interface should be, fundamentally, easy to learn and remember and also that the most adequated dialog techniques were menus, fill-in-forms and direct manipulation.

Our goal was to develop a system that allows the average user (with the above described profile) to perform a simple analysis and produce results using data introduced by her/himself. This should be accomplished after spending only a reasonable amount of time learning how to use the system.

As no formal training was considered necessary, the only training available will be the *User's Guide* and the *On-Line Help*.

III. THE DECISION SUPPORT SYSTEM BASED ON FACTOR ANALYSIS

According the system specification it should allow the user to:

— introduce (edit) new (existing) data;
— perform four types of factor analysis;
— display the results in numerical or graphical form;
— manage the corresponding files;
— get On-Line Help.

The DSSBFA can be viewed as having two separate blocks: the User Interface (UI) and the code corresponding to the Factor Analysis. These two blocks communicate via a set of files containing data and results in standard formats so that the system can display data and results produced by other already existing applications that implement Factor Analysis.

As stated before, the UI of the DSSBFA had to be essentially easy to learn and remember, but should also offer some attractive features to more experienced users. To meet these requirements the inexperienced user is offered the possibility of entering all the information needed to perform the intended analysis through dialogue boxes that prompt her/him to input the data set (variable names, individual names and numerical data) as well as other needed information. An On-Line Help is also supplied. To support more skilled users, that know exactly what kind of information the analysis needs, or want to use and/or modify already existing data, the system offers the possibility of entering or editing all the information using a "spread-sheet like" editor. After performing the analysis, the results are displayed in a numerical or graphical form, interactively specified by the user.

IV. USER'S INTERFACE

Implementing a user interface is a difficult task, there is no standard procedure, but several authors agree on the fact that some fundamental principles and guidelines are to be observed in the design of a user interface. This principles are: user and task compatibility, familiarity and simplicity, consistency, minimizing errors, easy error recovery [3,4].

Due to the user's profile and task characteristics identifyed previously, several dialogue styles were used simultaneously: direct manipulation, menus, fill-in-forms and function keys. The functionality is presented in a quasi hierarchical structure (fig.1 shows the first level of this hierarchy).

The most significant part of this functionality is presented to the user through the options Analysis, Edit and Results. Fig. 2 shows a simplified flow chart of the interface.

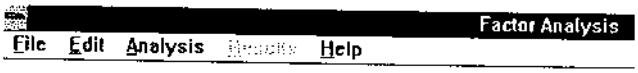


Fig.1.- Window corresponding to main menu

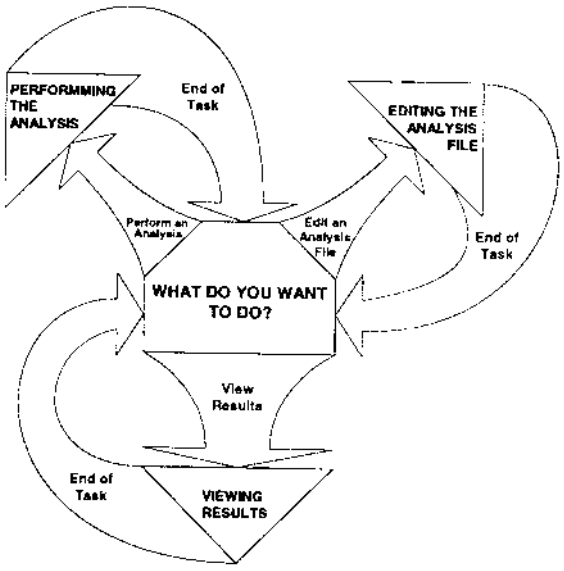


Fig.2.- Simplified Flow Chart of the User's Interface

1. Analysis

This is the first choice for the inexperienced user. It gives access to a guided (though, not flexible) process of inputing all the information needed to perform the analysis, through a set of dialog boxes. Fig. 3 shows a simplified flow chart of this option.

As three of the four types (PCA, CrA and DcA) of analysis need the same type of information, the dialogue is similar. Distance Analysis requires a simpler dialogue. Fig. 4 shows a simplified flow chart of the dialogue corresponding to PCA, CrA and DcA.

Figs 5 to 9 show the sequence of dialogue boxes as they appear to the user and illustrate, with an example, the interactive process that helps the user to input the parameters of the analysis and data set.

After the introduction of general information (identification of the analysis) regarding the data to introduce, the dialog box of the fig.6 allows the user to input specific information, namely parameters concerning the data set, type of metric to use and possible supplementary data).

After this procedure, the system will prompt the user to input the names of the variables (columns), individuals (rows) and the data matrix as shown in figs 7, 8 and 9.

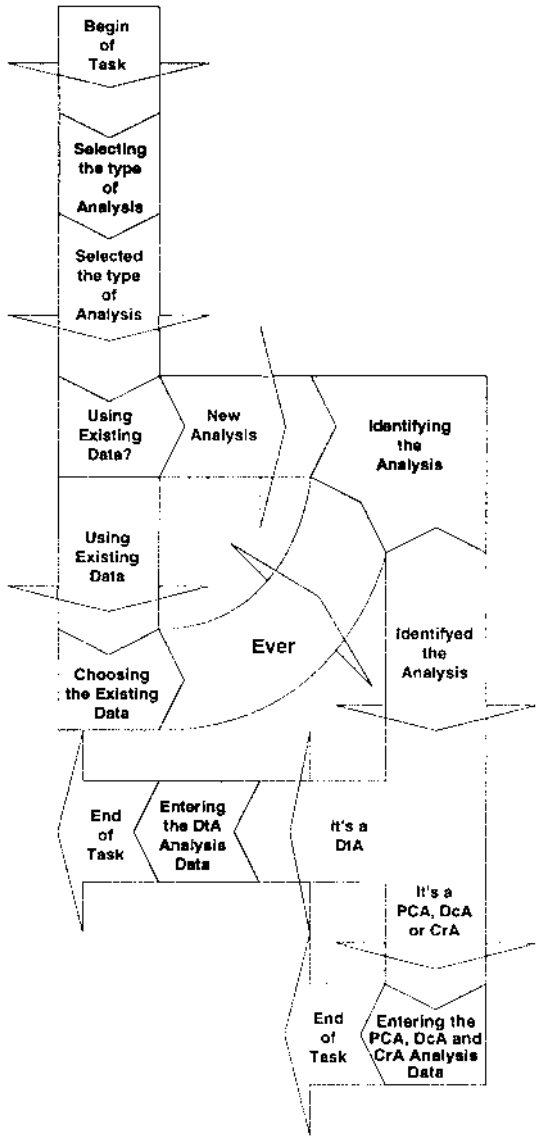


Fig.3 - Simplified flow chart of "Analysis"

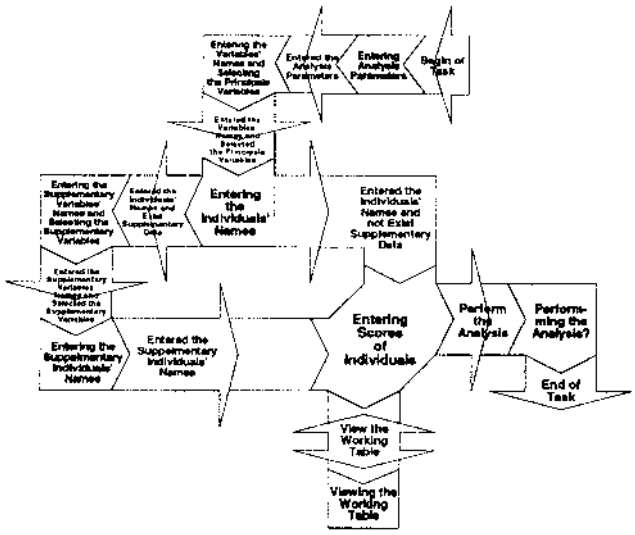


Fig.4- Simplified data flow chart of the process of entering data for PCA, CrA and DcA

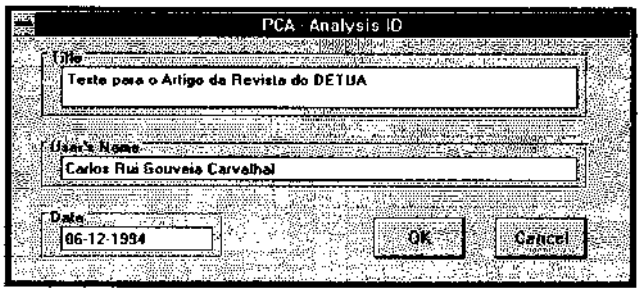


Fig.5.- PCA/DcA/CrA/DtA- Analysis-ID Window

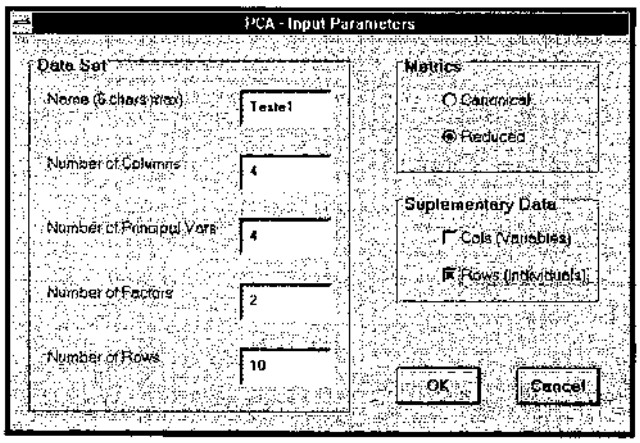


Fig.6.- PCA/DcA/CrA- Input Parameters Window.

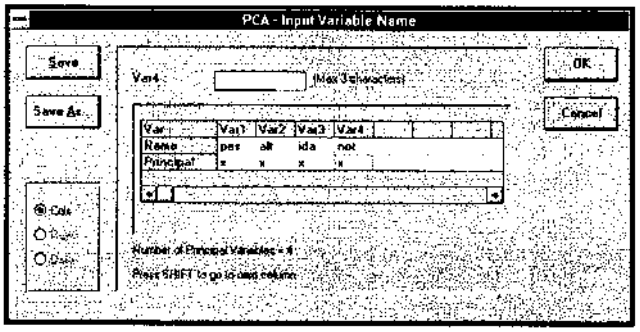


Fig.7.- PCA/DcA/CrA- Input Variables' Names Window.

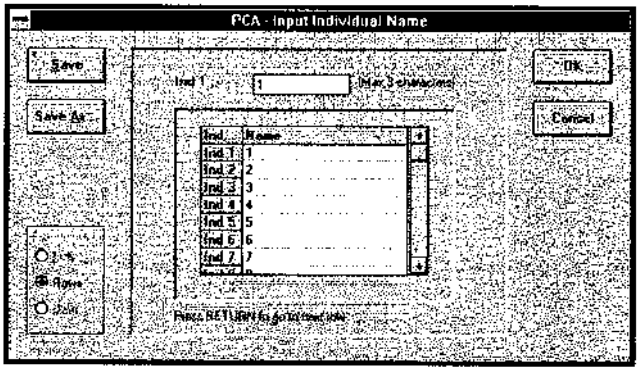


Fig.8- PCA/DcA/CrA- Input Individuals' Names Window.

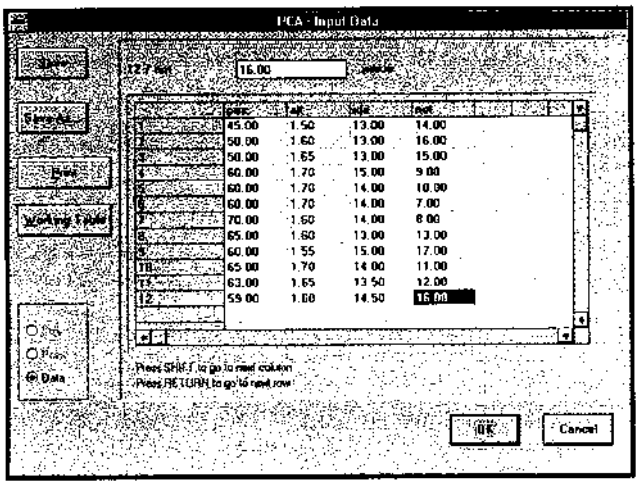


Fig 9.- PCA/DcA/CrA- Input The Data Window

Fig. 10 shows a simplified data flow chart of the dialogue corresponding to Distance Analysis (DtA), and figs 11-13 show the sequence of dialogue boxes as they appear to the user.

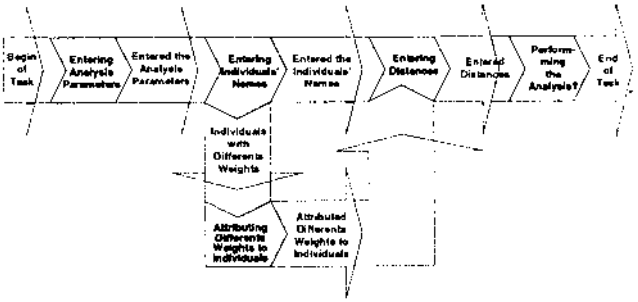


Fig. 10 - The "Entering the DtA Analysis data" Architecture

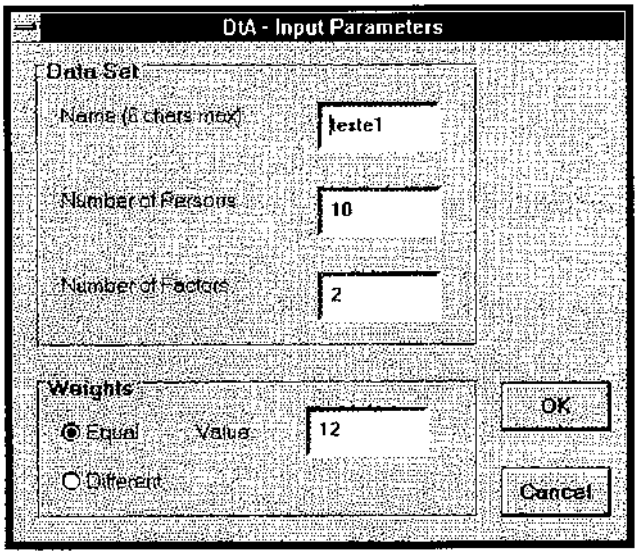


Fig.11.- DtA- Input Parameters Window

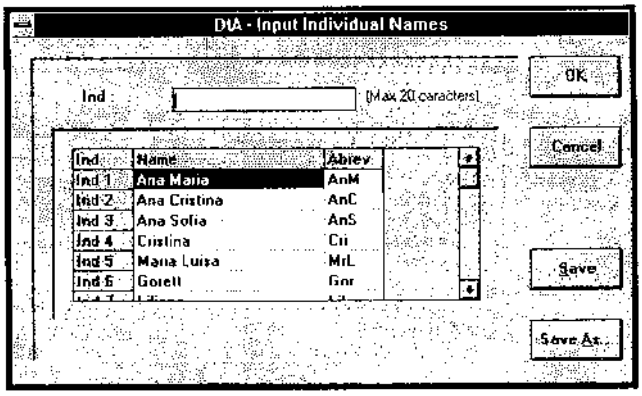


Fig.12- The PCA/DcA/CrA- Input Individuals' Names Window

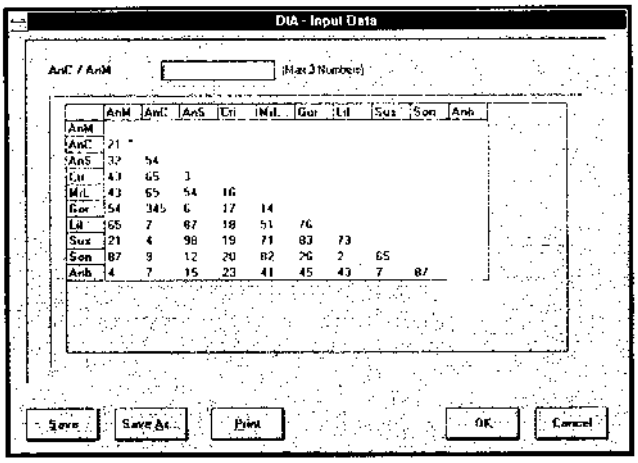


Fig 13.- DtA- Input Distances Window

2. *Edit*

To input new data or edit existing data, a "spread sheet like editor" is offered. This editor may substitute dialogue boxes shown in figs 7-9 since it can input or edit information concerning the data set, variable names or individuals names. It corresponds to a more flexible way of inputting data, more adequate to an experienced user.

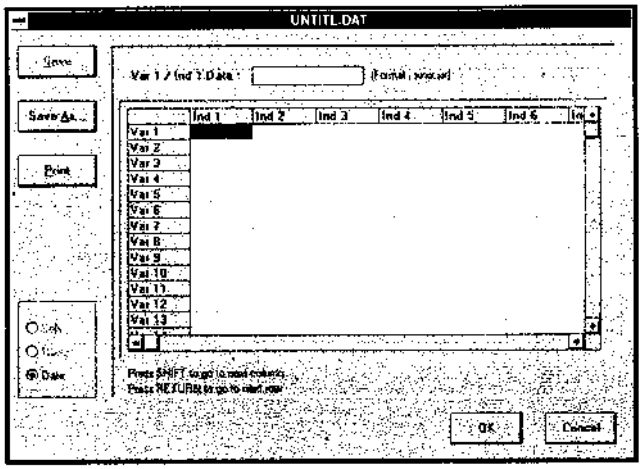


Fig 14 - Editor Window - can be used to input or edit the names of variables and individuals as well as the data set.

3. *Results*

Fig. 15 presents the flow chart corresponding to the Results option. This option allows the user to visualize the results in two different formats: numerically or graphically, as a projection in a 2D space which axis are chosen by the user in the Customize Window. Figs 16 to 18 show some results concerning the same example, used to illustrate the process of inputting data.

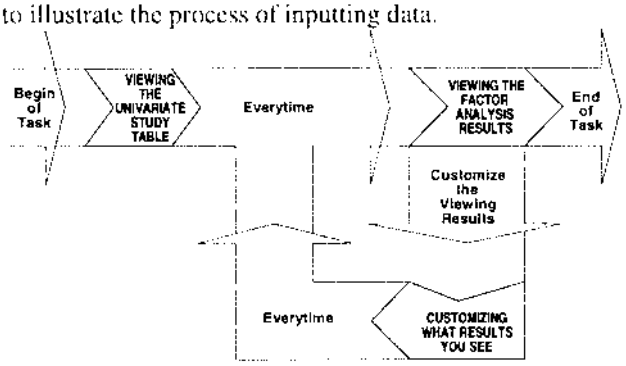


Fig.15 - The "Viewing Results" Architecture.

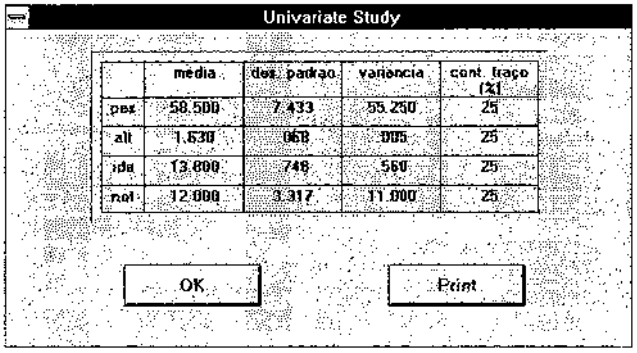


Fig.16 - The Univariate Study Table Window.

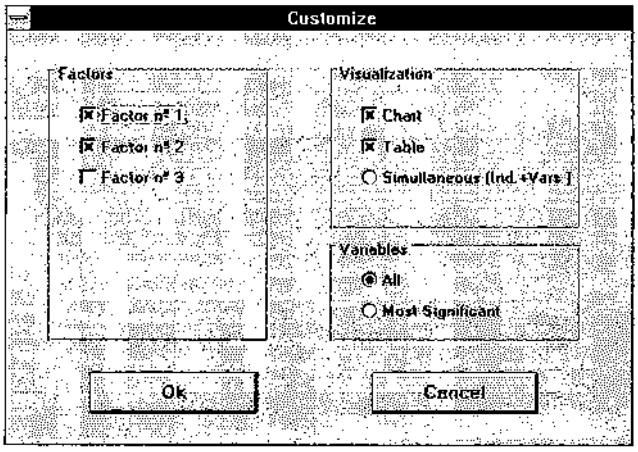


Fig.17 - Customize Window.

This example refers to the study of a group of students according to four characteristics: weight, height, age and mark. On reducing from four to two dimensions, only 20% of the information (variance) is lost.

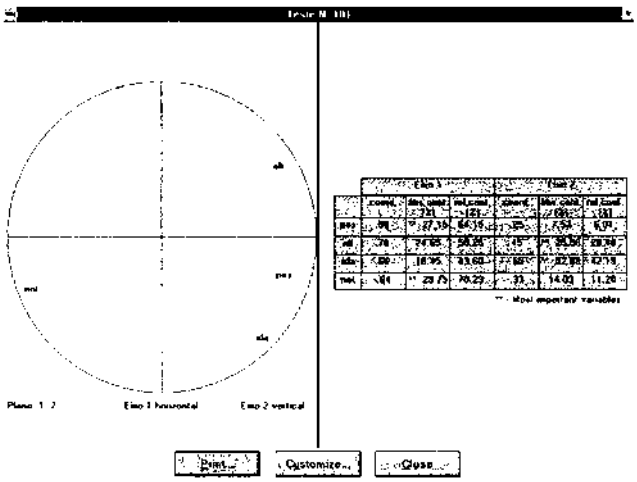


Fig.18 - Results Window.

VI. DISCUSSION

This work had two main goals: to develop an interface to a DSSBFA (Decision Support System Based on Factor Analysis) and simultaneously to illustrate the full procedure of designing, implementing and testing a user interface. A significant part of these goals, the study of the user and task characteristics, the design and the implementation of the user interface, was accomplished. A set of fundamental principles and guidelines for human-computer interface design was applied and we feel that the actual interface respects, in essence, this principles and guidelines. All this work was followed by a mathematician whose participation proved, as expected, to be indispensable.

Due to time constraints, it was not possible to perform the full test of the interface. This process should involve the design and application of tests to a set of users considered to be representative of the target population, as well as using the obtained information to improve the current interface. Since the design, implementation and test of a user interface is an iterative process, we expect to proceed this work during the courses on Human Computer Interfaces in the years to come.

ACKNOWLEDGMENT

This work has been partially financed by JNICT - Junta Nacional de Investigação Científica e Tecnológica - grant BM1459.

REFERENCES

[1] Foucart T. *Analyse Factorielle* , Masson , 1985
[2] Microsoft Visual Basic - Language Reference - Programming System for Windows, 1991
[3] Deborah Meyhew, *Principles and Guidelines in Software User Interface Design*, Prentice Hall, 1992
[4] Foley J.D., van Dam A., Feiner S.K., Hughes J.F., Computer Graphics - Principles and Practice, Addison-Wesley Publishing Company, 2nd. edition, 1991

Modelo Dinâmico Neuronal para a Percepção Categorial da Fala

Estela Bicho, Gregor Schoner, Francisco Vaz

Resumo- **O fenómeno da percepção categorial tem desempenhado um importante papel na teoria da percepção da fala. Uma das razões provem do facto da percepção categorial se mostrar como o fenómeno básico e fundamental envolvido no interface entre o sinal analógico sensorial (sinal acústico) e a representação discreta e simbólica da linguagem. Outro fenómeno que se manisfesta na percepção de sons de fala é a adaptação selectiva. Esta emerge sob a forma de saltos na fronteira categorial. Para reproduzir e explicar a categorização de sons de fala num *continuum* de vozeamento propomos um modelo dinâmico neuronal. Mostramos que este modelo reproduz os padrões típicos observados na percepção categorial, bem como efeitos de histerese e efeitos de adaptação. Comparamos as predições do modelo com os resultados experimentais. Deste estudo concluimos que a percepção categorial pode ser compreendida como o resultado de um processo de competição dentro de uma representação neuronal da informação sensorial.**

Abstract- **The phenomenon of categorical perception has played an enormous role in the theory of speech perception. One reason is that categorical perception is at the interface between the analog sensory signal and the discrete and simbolic nature of language. Another phenomenon involved in speech perception is selective adaptation. Adaptation effects appear with respect to the location of the categorical boundary along a speech *continuum*. To reproduce and account the categorization of speech sounds within a voicing *continuum* we propose a dynamic neural model. We show that this model is able to reproduce the tipical patterns observed in experiments of categorical perception, histeresis and adaptation effects. We compare the model predictions to experiments with subjects. From this study we conclude that categorical perception may be understood as resulting from competition within a neural representation of sensory information.**

I. INTRODUÇÃO

A. Definição do fenómeno e paradigma clássico

O fenómeno da percepção categorial tem desempenhado um importante papel na teoria da percepção da fala. Uma das razões provem do facto da percepção categorial se mostrar como o fenómeno básico e fundamental envolvido no interface entre o sinal analógico sensorial (sinal acústico) e a representação discreta e simbólica da linguagem.

Falando genéricamente, diz-se que a percepção categorial ocorre quando a capacidade de discriminação entre membros de uma mesma categoria é muito fraca, enquanto que a capacidade de discriminação de elementos pertencentes a categorias diferentes é muito boa [1].

Embora a investigação da percepção categorial seja, em princípio, uma área muito vasta permitindo uma certa variedade de métodos, ao longo dos anos tem sido identificada com um paradigma laboratorial particular. O paradigma clássico da percepção categorial envolve duas tarefas, uma de identificação e a outra de discriminação [2]. Na tarefa de identificação de sons, logo após a apresentação do estímulo, o sujeito tem que responder qual o som percebido. Os estímulos que pertencem a um *continuum* físico "dividido" normalmente em duas categorias, representadas inequivocamente pelos estímulos extremos do *continuum*, são apresentados repetidamente numa ordem aleatória para serem classificados numa ou noutra categoria. Com base nos resultados desta tarefa é possível verificar para cada estímulo a percentagem de respostas de uma determinada categoria e definir a função de identificação. Na tarefa de discriminação de sons usa-se tipicamente o paradigma ABX. Neste paradigma para cada tríade de estímulos ABX, em que os dois primeiros estímulos são diferentes, embora possam vir da mesma categoria fonémica, o ouvinte tem que indicar com qual é que o terceiro estímulo é idêntico. Os resultados são em seguida representados numa função de discriminação que mostra a relação entre a posição no *continuum* e a percentagem de respostas correctas para as tríades.

Como referência experimental recente podemos relatar o trabalho de Castro [3]. Neste trabalho foi usado o paradigma experimental clássico da percepção categorial para testar um *continuum* de vozeamento, baseado na fala natural portuguesa, entre as sílabas /ba/ e /pa/, constituido por estímulos que vão desde /pa/ com 0 ms de pré-vozeamento até /pa/ com 70 ms de pré-vozeamento (ou -70ms de VOT), a intervalos de 7 ms (ou -7 ms de VOT). Os resultados obtidos para um dos sujeitos intervenientes nesta experiência são apresentados na figura 1. Os resultados demonstram claramente a existência de percepção categorial para o *continuum* de vozeamento. Como vemos, a discriminação de sons de fala pertencentes a duas categorias diferentes, tais como /ba/ e /pa/, é relativamente fácil, enquanto que a discriminação entre dois /pa/ de um mesmo orador é quase inexistente. Por

outro lado, diferentes elocuções /pa/ são categorizadas identicamente. Mas, apesar disto as fronteiras entre categorias são flexíveis, ajustando-se com factores tais como contexto fonético, ritmo da fala, experiência linguística, entre outros.

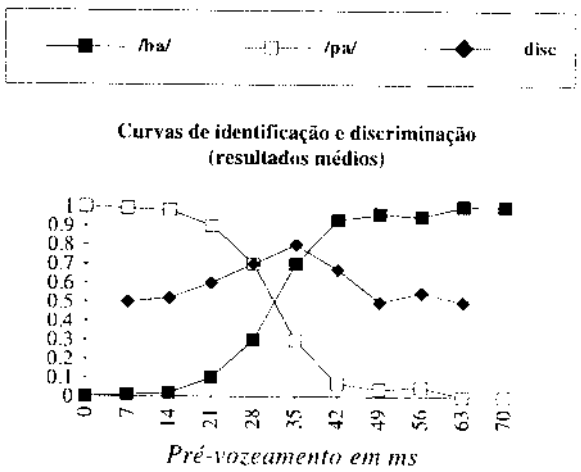


Fig. 1 Curvas de identificação e de discriminação (resultados médios). Em abcissa os estímulos de [pa] com 0 ms de pré-vozeamento a [pa] com 70 ms de pré-vozeamento, a intervalos de 7 ms. . Cada ponto da curva de discriminação diz respeito à comparação entre os estímulos imediatamente anterior e subsequente [Castro (1993)].

B. Adaptação Selectiva

Um outro fenómeno envolvido na percepção da fala é a adaptação selectiva. Estes efeitos de adaptação aparecem em relação à localização da fronteira fonémica ao longo do *continuum* de fala. Quando os sujeitos ouvem repetidamente um dos esímulos do *continuum*, que são percebidos como exemplares claros das categorias que representam, o resultado é um salto da fronteira na direcção da categoria perceptual áquela a que o estímulo adaptador pertence [2]. Os efeitos da adaptação selectiva são de origem auditiva e devem-se às relações espectrais entre os estímulos de teste e o adaptador [4].

C. Histerese

Em outras áreas da percepção, a reorganização discreta dos perceptos, à medida que os parâmetros do estímulo são variados continuamente, tem sido relacionada com cooperatividade. Este efeito é caracterizado por regimes multi-estáveis e uma espécie de memória perceptual designada por histerese [5]. Por exemplo, o trabalho em fusão binocular e em percepção visual de movimento revelou efeitos de histerese [6], em que a organização de um percepto depende não só de estímulo corrente, mas também da prévia organização perceptual. Uma questão importante que se levanta é até que ponto a cooperatividade pode desempenhar um papel importante na percepção categorial da fala. Para investigar o efeito de cooperatividade, Castro e seus colaboradores [7], têm vindo a explorar a existência de histerese na percepção fonémica. O fenómeno de histerese manifesta-se quando numa tarefa de identificação de sons, a série de estímulos consiste numa lista em que o valor do estímulo acústico é apresentado desde um extremo do *continuum* até ao outro, e, se esta série de estímulos é apresentada numa experiência em ordem ascendente e noutra na ordem inversa, então resulta que os pontos no *continuum* onde ocorre uma transição perceptual depende da direcção de apresentação. Em cada caso, a transição perceptual é "atrasada", ou seja a fronteira categorial desloca-se no sentido de apresentação da série, tal que é obtida uma região de sobreposição. Nesta região do *continuum* o percepto formado depende da direcção em que a série é apresentada. A manifestação deste fenómeno vem demonstrar que a emergência de um percepto pode depender não só do estímulo corrente mas também do percepto anterior.

D. A motivação da teoria dinâmica não linear

As evidências experimentais mostram que a formação perceptual de padrões (ou de perceptos) e transições entre perceptos exibem um comportamento dinâmico. Em particular, os conceitos de estabilidade e cooperatividade são essenciais para um bom entendimento do modo como são formados perceptos coerentes. Estabilidade e cooperatividade são também conceitos centrais da teoria dinâmica não linear e da teoria da formação de padrões. Consequentemente, há muitas ideias da dinâmica que poderiam ser desenvolvidas para a percepção. A teoria dinâmica revela-se ainda uma bordagem adequada para lidar com os efeitos da adaptação selectiva. Isto acontece, porque o processo de adaptação selectiva e o processo de organização perceptual propriamente dito ocorrem em níveis diferentes e as suas escala temporais são muito diferentes, o que permite dasacoplar com relativa facilidade estes dois processos dentro da teoria dinâmica.

E. Hipóteses de aplicações do modelo

Numa perspectiva bastante ambiciosa e a longo termo, poderíamos explorar a hipótese de integrar o modelo dinâmico neuronal, que aqui apresentamos, num sistema de reconhecimento automático de fala, ao nível da categorização elementar dos sons de fala.

Um sistema de reconhecimento automático de fala é bastante complexo e deve envolver vários níveis de processamento, desde a segmentação e categorização fonémica (porque a fala pode ser interpretada como uma sequência de fonemas) até à interpretação semântica. A filosofia do modelo que propomos poderia ser usada para implementar o nível de segmentação do sinal de fala numa sequência de fonemas. À partida perspectivamos as seguintes vantagens: O problema da invariância poderia ser parcialmente reduzido, pois as fronteiras perceptuais (ou categoriais) do modelo são flexíveis. Summerfield [8] descobriu que quando o ritmo da fala aumenta, as diferenças entre as consoantes vozeadas e não-vozeadas

diminuem em relação ao VOT, e observou que concomitantemente a localização da fronteira perceptual que marca a distinção entre as consoantes vozeadas e não vozeadas desloca-se na direcção de valores de VOT menores. Adicionalmente, resolve-se também o problema para vários oradores. Isto porque, devido aos saltos nas fronteiras sob efeitos de adaptação o nível de categorização fonémica seria capaz de se "adaptar" a um orador particular. Esta propriedade poderia ser também considerada para sistemas de reconhecimento de orador. Relativamente aos efeitos de contexto, estes poderiam ser também já tratados a este nível como uma espécie de memória perceptual.

II. MODELO

A. Estrutura conceptual

O modelo que propomos consiste em três níveis de representação como é ilustrado na figura 2 e que passamos a descrever.

a) Nível de VOT

O nível de entrada (ou de VOT) é o nível de representação mais baixo. Neste, é feita uma representação discreta dos estímulos acústicos através de um conjunto de detectores acústicos equidistantes colocados ao longo do eixo de VOT. Cada um destes detectores acústicos sinaliza a presença de um estímulo acústico com um valor de VOT particular.

b) Nível auditivo

O nível intermédio, ou auditivo, é onde ocorrem os efeitos de adaptação. O nível de representação do nível auditivo consiste em três camadas de neurónios diferentes, como é ilustrado na figura2. Cada neurónio da camada Z, camada de entrada do nível auditivo, é especializado numa parte particular do eixo de VOT e assume-se que há sobreposição. Estes neurónios são excitados proporcionalmente à sua sensibilidade ao valor do estímulo acústico, sinalizado por um dos detectores do nível de VOT. Isto conduz em geral à situação em que mais do que um neurónio da camada Z está activo. Desta maneira é necessário um processo de normalização. Este processo é realizado na camada Y, em que todos os neurónios estão desligados excepto o neurónio correspondente ao maior z_i. Quando dois (ou mais) neurónios da camada Z tem o mesmo valor de excitação e este é o máximo valor então na camada Y dois (ou mais) neurónios ficam ligados. Os neurónios y_i possuem um limiar mínimo de activação porque na ausência do estímulo acústico não queremos que eles fiquem ligados devido a quaisquer pequenas flutuações nos neurónios de entrada.

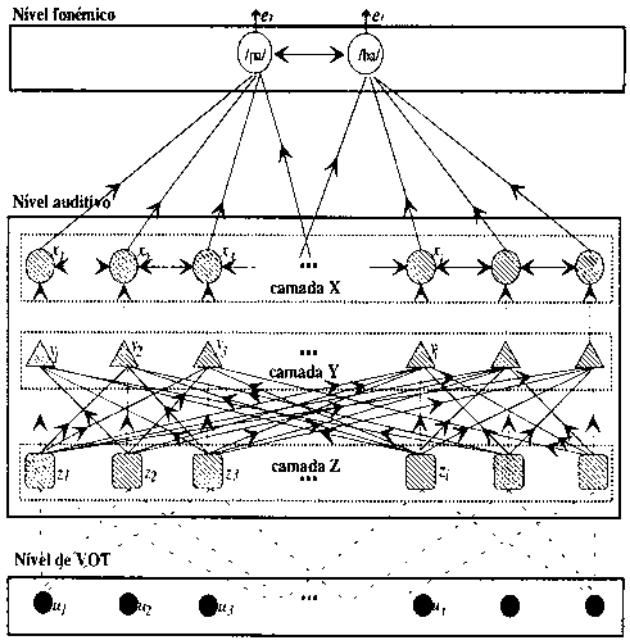


Fig. 2 Níveis de representação para o modelo dinâmico neuronal para a percepção categorial e adaptação selectiva

Consideramos que a saída do nível auditivo define um espaço contínuo (caracterizado por um parâmetro a). Mas, este espaço é amostrado através de um conjunto de detectores equidistantes ao longo do eixo a, a figura 3 ilustra este processo.Os neurónios x_i representam estes detectores e o índice i rotula a_i. $x_i=\pm 1$ representa uma detecção no nível auditivo do valor a_i e $x_i=0$ representa a ausência de detecção deste valor.

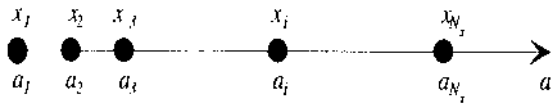


Fig. 3 Amostragem do espaço definido pelo parâmetro a

Os neurónios x_i constituem a última camada do nível auditivo e formam um campo dinâmico de neurónios que competem globalmente. Esta competição conduz à activação dominante de um único neurónio da camada X, este é o neurónio que na situação corrente melhor representa o estímulo acústico. O papel desta camada é particularmente importante na situação em que mais do que um neurónio da camada Y está ligado. Esta situação irá ocorrer em consequência dos efeitos de adaptação.

Como podemos ver, o nível auditivo é basicamente um filtro complicado que na ausência de adaptação iria conduzir a uma relação unívoca entre os valores do estímulo acústico (VOT), sinalizados por u_i e a_i. Na figura 4a) ilustra-se o mapeamento dos valores de VOT em valores de a na ausência de adaptação.

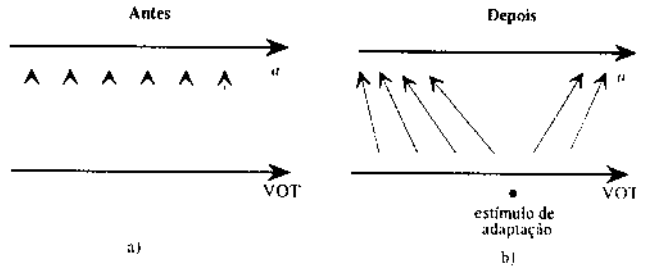


Fig 4 Em a) ilustra-se o mapeamento dos valores de VOT em valores de a na ausência de efeitos de adaptação. E em b) ilustra-se o mesmo mapeamento após a adaptação

A adaptação é introduzida através de uma distorção sistemática deste mapa, tal que, após alguns períodos de detecção de um valor particular a_i, o seu valor de VOT correspondente (estímulo de adaptação) e valores de VOT vizinhos são mapeados em valores de a que estão mais afastados do valor de a_i do que antes da adaptação, ver figura4.b). Uma maneira simples de realizar isto consiste em reduzir a sensibilidade dos neurónios da camada Z que estão activos, isto é, enfraquecer a força do seu acoplamento ao nível de VOT. Esta variação da sensibilidade é dinâmica se assumirmos que é gradual durante a adaptação e durante a recuperação. Um aspecto importante a ter em mente é que para que os efeitos da adaptação sejam sentidos ao longo das simulações, a recuperação da sensibilidade tem que ser muito mais lenta do que a sua diminuição.

c) Nível fonémico

O nível de representação mais elevado é o nível fonémico, onde a saída das operações realizadas no nível auditivo são organizadas em termos de um percepto fonémico. Este nível consiste em dois neurónios dinâmicos que recebem estimulação do nível auditivo. Os neurónios competem entre si e a sua mútua activação exclusiva representa um estado perceptual, que conduz a qualquer uma das duas respostas categoriais.

B. Formalização matemática

a) Nível de VOT

Os detectores do nível de VOT são simplesmente células do tipo ligadas/desligadas que sinalizam a presença de um valor particular do estímulo acústico:

$$u_i = \begin{cases} 1 \text{ se estímulo} = \text{VOT}_i \\ 0 \text{ se estímulo} \neq \text{VOT}_i \end{cases} \quad (1)$$

$u_i = 1$ sinaliza a presença de um estímulo acústico com valor VOT_i. Neste procedimento está implícito uma amostragem do eixo de VOT pelas células u_i. O índice i rotula o valor de VOT VOT_i.

b) Nível auditivo

Cada neurónio z_i é sensível a uma zona particular do eixo de VOT e não responde equivalentemente a todos os valores compreendidos dentro do seu campo receptivo. A função de sensibilidade de um neurónio z_i é dada por

$$v_i = \begin{cases} V_{0i}\left(1 - \dfrac{2(CRF_i - VOT)}{RF}\right) \text{ se } CRF_i - \dfrac{RF}{2} < VOT < CRF_i \\ V_{0i}\left(1 + \dfrac{2(CRF_i - VOT)}{RF}\right) \text{ se } CRF_i < VOT < CRF + \dfrac{RF}{2} \\ 0 \qquad \text{caso contrário} \end{cases} \quad (2)$$

Onde,

V_{0i} representa o valor máximo de sensibilidade

RF é o campo receptivo destes neurónios

CRF_i é o centro do campo receptivo do neurónio z_i

Os neurónios z_i são excitados pelas células do nível de VOT do seguinte modo

$$z_i = v_{il} u_l \quad \text{para} \quad i = 1, 2, \ldots N_z \ \wedge \ l = 1, 2, \ldots, N_u \quad (3)$$

onde v_{il} representa a sensibilidade do neurónio z_i ao valor do estímulo acústico representado por u_l.

Na figura 5 é ilustrado o padrão de excitação, na ausência de adaptação, gerado nos neurónios da camada Z quando a célula u_6 do nível de VOT está ligada.

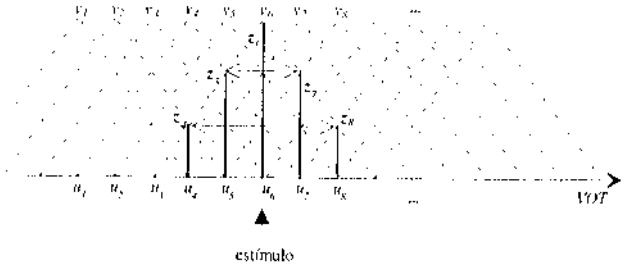


Fig 5 Padrão de excitação, na ausência de adaptação, gerado nos neurónios da camada Z quando a célula u_6 do nível de VOT está ligada.

Os neurónios da camada Y são excitados do seguinte modo:

$$y_i = \begin{cases} 1 \text{ se } z_i \text{ é o máximo valor} \wedge \text{é superior ao limiar } l_y \\ 0 \text{ caso contrário} \end{cases} \quad (4)$$

onde l_y representa o valor do limiar mínimo de activação destes neurónios. Na figura 5 é ilustrada a função realizada por esta camada.

A camada de saída do nível auditivo consiste num campo dinâmico não-linear de neurónios, identificados por x_i, que competem globalmente. A dinâmica do seu comportamento é descrita por um sistema de equações diferença, que na forma matricial é dada por:

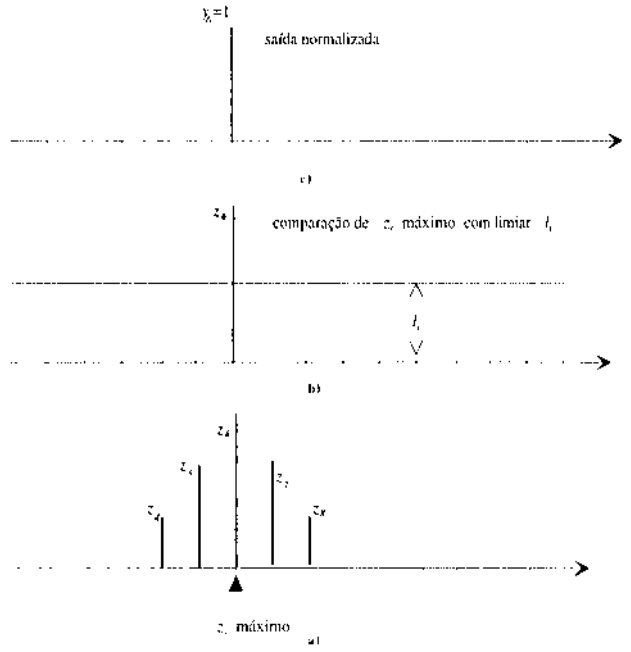


Fig 6 a) Selecção do neurónio z_i com a máxima excitação. b) Comparação de z_i máximo com o limiar l_y. Ilustração da função realizada pela camada Y. c) saída da camada Y do nível auditivo

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \mathbf{h}(\mathbf{y}, \mathbf{x}(n)) \qquad [-1,1] \bmod 2 \qquad (5)$$

$\mathbf{x}(n+1)$ e $\mathbf{x}(n)$ são vectores coluna, de dimensão N_x que representam o estado dos neurónios da camada X nos instantes $n+1$ e n respectivamente; $\mathbf{h}(\mathbf{y}, \mathbf{x}(n))$ é o campo de vectores total e $\mathbf{y}$ é um vector coluna que representa a entrada da camada X. $\mathbf{y}$ e $\mathbf{h}(\mathbf{y}, \mathbf{x}(n))$ são também vectores de dimensão N_x. Nós restringimos cada variável x_i ao espaço de fase circular [-1,1]mod2 de modo a ter soluções limitadas. Por conseguinte, apenas pontos fixos no intervalo [-1,1] são importantes.

O campo de vectores $\mathbf{h}(\mathbf{y}, \mathbf{x}(n))$ foi definido de modo a capturar as especificações do problema através de forças componentes que definem atractores ou repulsores do sistema dinâmico. No projecto de tais forças componentes fizemos uso da linguagem da teoria qualitativa de sistemas dinâmicos [9, 10, 11]. As contribuições das várias forças componentes para o campo de vectores total são tratadas aditivamente. As contribuições individuais são definidas de modo a gerar o diagrama de fase desejado, caracterizado por soluções invariantes e respectivas estabilidades, na ausência de todas as outras contribuições [12].

O campo de vectores $\mathbf{h}(\mathbf{y}, \mathbf{x}(n))$ é a soma de três forças componentes:

$$\mathbf{h}(\mathbf{y}, \mathbf{x}(n)) = \mathbf{h}_{\text{info}}(\mathbf{y}, \mathbf{x}(n)) + \mathbf{h}_{\text{comp}}(\mathbf{x}(n)) + \mathbf{h}_{\text{stoch}}(n) \qquad (6)$$

onde $\mathbf{h}_{\text{info}}(\mathbf{y}, \mathbf{x}(n))$ representa a informação que transita da camada Y para a camada X, dentro do nível auditivo. $\mathbf{h}_{\text{comp}}(\mathbf{x}(n))$ é a componente que determina a competição entre os neurónios da camada X. Finalmente, $\mathbf{h}_{\text{stoch}}(\mathbf{x}(n))$ representa a natureza estocástica dos neurónios x_i. Para o neurónio i estas várias componentes são dadas por:

$$h_{\text{info},i}\left(y_i, x_i(n)\right) = \varepsilon(y_i)\left(x_i(n) - x_i^3(n)\right) \qquad (7)$$

$$h_{\text{comp},i}\left(\mathbf{x}(n)\right) = \gamma_x \sum_{j=1(j\neq i)}^{N_x} x_j^2(n)\left(x_i^3(n) - x_i(n)\right) \qquad (8)$$

$$h_{\text{stoc},i}(n) = \sqrt{Q_x}\,\xi_x(n) \qquad (9)$$

onde $\varepsilon(y_i)$ é a função de entrada do neurónio x_i e tem a forma

$$\varepsilon(y_i) = y_i - l_\varepsilon \qquad (10)$$

em que l_ε tem que ser escolhido de modo a que tenhamos $\varepsilon(y_i)>0$ se y_i estiver ligado e $\varepsilon(y_i)< 0$ se y_i estiver desligado. γ_x é o parâmetro do modelo que determina a força da componente de competição. Q_x é a magnitude da natureza estocástica dos neurónios da camada X. E, por fim $\xi_x(n)$ representa ruído branco gaussiano. A introdução desta componente permite a fuga a pontos fixos instáveis.

Os neurónios x_i que tem uma função de entrada negativa, $\varepsilon(y_i)< 0$, ficam inibidos e os que possuem uma entrada positiva ficam activados. Mas no final, devido ao processo de competição apenas um destes neurónios vai ficar ligado. São dois os factores que determinam qual dos neurónios, que têm uma função de entrada positiva, vai ganhar a competição. O primeiro é a excitação inicial dos neurónios, $x_i(0)$, e o segundo é a sua natureza estocástica.

Os fenómenos de adaptação vão ocorrer na camada Z. Como já vimos uma forma de lidar com a adaptação consiste em enfraquecer a sensibilidade de todos os neurónios z_i que têm o estímulo de entrada dentro do seu campo receptivo. Isto é conseguido fazendo o parâmetro V_{0k} dinâmico e diferente para cada neurónio z_i. A dinâmica de V_{0k} pode muito simplesmente ser dada pela seguinte equação diferença:

$$V_{0k}(n+1) = V_{0k}(n) - \alpha'_{\text{adap}}\left(V_{0k}(n) - V_{0\min}\right) - \alpha'_{\text{recup}}\left(V_{0k}(n) - V_{0\max}\right) \qquad (11)$$

onde

α'_{adap} representa a taxa efectiva de decaimento de V_{0k}.

α'_{recup} é a taxa efectiva de recuperação de V_{0k}.

α'_{adap} e α'_{recup} são dados pelo produto de duas componentes:

$$\alpha'_{\text{adap}} = z_i^2 \alpha_{\text{adap}} \qquad (12)$$

$$\alpha'_{\text{recup}} = \left(1 - z_i^2\right)\alpha_{\text{recup}} \qquad (13)$$

em que α_{adap} e α_{recup} são parâmetros que determinam a taxa de decaimento e recuperação de V_{0k} respectivamente. Como vemos, para a adaptação o valor de z_i também é importante, porque um dado adaptador não provoca a

mesma adaptação em todos os neurónios z_i . Os que sofrem os maiores efeitos de adaptação são aqueles que num dado momento são mais sensíveis ao estímulo adaptador e por conseguinte os que ficam mais excitados.

Na figura 7 ilustramos o perfil das funções de sensibilidade após a adaptação com um estímulo de valor VOT_4, o padrão de excitação gerado nos neurónios da camada Z em resultado dos efeitos de adaptação e no topo da figura podemos observar que o neurónio da camada Y que agora fica ligado é o y_8. É interessante comparar o padrão de excitação gerado nos neurónios da camada Z na ausência e na presença de feitos de adaptação.

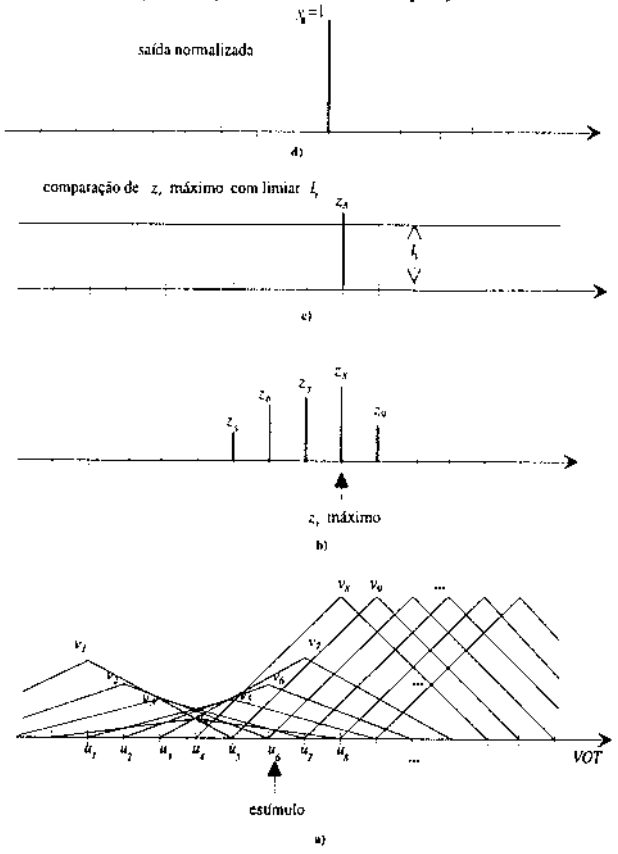


Fig 7 a) Funções de sensibilidade após a adaptação com VOT_4. b) padrão de excitação gerado sob os efeitos de adaptação nos neurónios da camada Z quando a célula u_6 do nível de *VOT* está ligada e saída da camada Y. c) Comparação de z_i máximo com limiar l_y. d) Saída da camada Y.

c) Nível fonémico

A saída de cada neurónio é identificada pela variável perceptual e_i, onde i é um índice que rotula fonemas. Um código que achámos conveniente no projecto da dinâmica do sistema é o seguinte: quando o sistema relaxa para um ponto fixo de e_i , perto de +1 ou -1 então o estado perceptivo é o fonema indexado por i (neurónio i ligado) e quando o sistema relaxa para um ponto fixo de e_i perto de zero então o fonema correspondente não é "percebido" (neurónio i desligado). Este comportamento é modelado por equações dinâmicas para as variáveis perceptuais que têm a seguinte forma:

$$e_i(n+1) = e_i(n) + g_i(\mathbf{e}(n), \mathbf{x}) , \quad e_i \in [-1,1] \bmod 2 \qquad i = 1,2 \quad (14)$$

onde $\mathbf{e}(n)$ exprime a dependência do campo de vectores total g_i em relação aos valores das variáveis perceptuais no instante n, e $\mathbf{x}$ é o vector de saída do nível auditivo. Este sistema de equações dinâmicas começa com uma condição inicial $\mathbf{e}(0)$, que nas simulações é escolhida de modo a representar a história perceptual prévia. Nós restringimos e_i ao espaço de fase circular [-1,1]mod2 de modo a ter soluções limitadas. Como consequência, apenas pontos fixos no intervalo [-1,1] são importantes.

O campo de vectores $g_i(\mathbf{e}(n), \mathbf{x})$ é a soma de três forças componentes:

$$g_i(\mathbf{e}(n), \mathbf{x}) = g_{\text{info},i}(e_i(n), \mathbf{x}) + g_{\text{comp}i}(\mathbf{e}(n)) + g_{\text{sinaltr}}(n) , \quad i = 1,2 \quad (15)$$

onde:

$g_{\text{info},i}(e_i(n), \mathbf{x})$ representa a contribuição da informação auditiva na dinâmica do nível fonémico e tem a forma:

$$g_{\text{info},i}(e_i(n), \mathbf{x}) = \beta_i(\mathbf{x})\left(e_i^3(n) - e_i(n)\right) , \qquad i = 1,2 \quad (16)$$

onde $\beta_i(\mathbf{x})$ é a função de entrada do neurónio fonémico i e define o seu grau de activação em relação ao valor corrente detectado no nível auditivo, i.e do parâmetro a. A função de acoplamento entre os níveis auditivo e fonémico tem a seguinte forma

$$\beta_i(\mathbf{x}) = \sum_{k=1}^{N_x} w_{ik} \left| x_k \right| - t_e \quad \text{para } i = 1,2 \qquad (17)$$

Nesta equação

t_e é um limiar que representa o potencial em repouso dos neurónios do nível fonémico. O valor deste limiar determina o tempo de relaxação do sistema para o estado em que os dois neurónios fonémicos ficam desligados, o que deve acontecer na ausência de estímulo acústico.

w_{ik} é a intensidade da ligação entre o neurónio e_i do nível fonémico e o neurónio x_k do nível auditivo. O seu valor define a sensibilidade do neurónio fonémico e_i ao valor a_k.

A função de sensibilidade dos neurónios fonémicos aos valores do parâmetro auditvo a é definida do seguinte modo:

$$w = \begin{cases} w_0 + \dfrac{w_0 - h}{a_{\text{sup}} - a_{c1}}\left(a - a_{\text{sup}}\right) \text{ se } i = 1 \\ w_0 - \dfrac{w_0 - h}{a_{c2} - a_{\text{inf}}}\left(a - a_{\text{inf}}\right) \text{ se } i = 2 \end{cases} \qquad (18)$$

onde a_{sup} e a_{inf} representam os limites superior e inferior do espaço definido por a respectivamente. a_{c1} e a_{c2} são os pontos onde vão ocorrer as fronteiras de estabilidade ao longo do *continuum* e o parâmetro h, como iremos ver na próxima secção, é o responsável pelo posicionamento das fronteiras de estabilidade ao longo do *continuum*. Este ser

escolhido de modo a que estas ocorram em a_{c1} e a_{c2}. $g_{comp,i}(\mathbf{e}(n))$ representa a dinâmica intrínseca que consiste na competição entre os dois neurónios. Esta componente tem aforma:

$$g_{comp,i}(\mathbf{e}(n)) = \gamma_o \gamma_{ij} e_j^2(n)\left(e_j^3(n) - e_i(n)\right), \qquad i,j = 1,2 \wedge i \neq j \tag{19}$$

onde γ_o representa a magnitude da dinâmica de competição e $\gamma_{ij} > 0$ reflecte o grau de competição entre neurónios. As ligações entre os neurónios são puramente inibitórias. Por exemplo, isto significa que a activação do neurónio fonémico 1 (i.e. /ba/), pode inibir a activação do neurónio fonémico 2 (i.e. /pa/).

Nós admitimos uma natureza estocástica para a saída dos neurónios que é modelada por uma força estocástica gaussiana:

$$g_{stoch,i}(n) = \sqrt{Q}\xi_i(n), \qquad i = 1,2 \tag{20}$$

onde $\xi_i(n)$ é ruído branco gaussiano que não está correlacionado entre passos subsequentes e $Q>0$ é a força do ruído. A inclusão desta força estocástica permite explicar a variabilidade no desempenho perceptual, transições espontâneas entre perceptos e fugas de perceptos instáveis, i.e., pontos fixos instáveis.

C. Estudo analítico

A seguir apresentamos os resultados dos estudos analíticos realizados para o campo dinâmico não linear de neurónios que constitui a camada X, para a dinâmica da adaptação selectiva que ocorre na camada Z do nível auditivo e para o nível fonémico. Nestes estudos aplicámos os métodos *standard* da teoria de sistemas dinâmicos. Primeiro determinámos os pontos fixos e em seguida avaliámos a sua estabilidade em relação aos vários parâmetros das equações.

a) Estudo analítico para a camada X *do nível auditivo*

A componente determinística para o campo de vectores total é dada pela soma das componentes de informação e de competição. Por conseguinte temos

$$h_{det,i}(y_i, x_i(n)) = h_{info,i}(y_i, x_i(n)) + h_{comp,i}(\mathbf{x}(n)) =$$
$$= \left(-\varepsilon(y_i) + \gamma_i - \sum_{j=1, j\neq i}^{N_x} x_j^3(n)\right)\left(x_i^3(n) - x_i(n)\right) [-1,1] \bmod 2 \text{ para } i = 1,\ldots,N_x \wedge i \neq j \tag{21}$$

Esta equação define pontos fixos em $\mathbf{x}_{fp}^{(0)} = (0,0,\ldots,0)_{N_x}$ (todos os elementos são nulos), $\mathbf{x}_{fp}^{(k)} = (0,\ldots,0,\pm 1,0,\ldots,0)_{N_x}$ (todos os elementos são nulos excepto o késimo que é ± 1) e $\mathbf{x}_{fp}^{*}$ (qualquer combinação de 0's e 1's e diferentes dos dois tipos anteriores). Segundo a teoria linear de estabilidade temos que:

- Uma condição necessária e suficiente para que $\mathbf{x}_{fp}^{(0)} = (0,0,\ldots,0)_{N_x}$ seja um estado estável é

$$-2 < \varepsilon(y_i) < 0 \quad \forall \quad i = 1,2,\ldots,N_x \tag{22}$$

Esta condição de estabilidade é satisfeita quando os neurónios da camada X não recebem qualquer estimulação dos neurónios na camada Y, situação em que temos $y_i = 0 \ \forall i = 1,\ldots,N_x$. Isto acontece na ausência de estímulo acústico.

- Uma condição necessária para que $\mathbf{x}_{fp}^{(k)} = (0,\ldots,0,\pm 1,0,\ldots,0)_{N_x}$, detecção com sucesso do valor a_k, seja um estado estável é

$$\gamma_\varepsilon - 2 < \varepsilon(y_i) < \gamma_x \quad \forall i = 1,2,\ldots,N_x \wedge i \neq k \quad \wedge \quad 0 < \varepsilon(y_k) < 1 \tag{23}$$

- Uma condição suficiente para que $\mathbf{x}_{fp}^{*}$ (vector com qualquer combinação de 0's e 1's e diferentes dos dois tipos anteriores) sejam pontos fixos instáveis é

$$\varepsilon(y_i) < \gamma \tag{24}$$

b) Estudo analítico para a dinâmica da adaptação selectiva

A equação que descreve a dinâmica dos fenómenos de adaptação que ocorrem na camada Z do nível auditivo é definida pela expressão (11). Quando o neurónio z_i contém o estímulo acústico dentro do seu campo receptivo a dinâmica do seu parâmetro V_{oi} reduz-se a

$$V_{oi}(n+1) = V_{oi}(n) - \alpha_{adap} z_i^2 \left(V_{oi}(n) - V_{o\min}\right) \tag{25}$$

Esta equação tem como ponto fixo $V_{oi} = V_{omin}$ e este é estável se $0 < \alpha_{adap} < 2$. Nesta situação a sensibilidade do neurónio z_i enfraquece. Quando pelo contrário, o estímulo acústico não caí dentro do seu campo receptivo a dinâmica do parâmetro V_{oi} é descrita por

$$V_{oi}(n+1) = V_{oi}(n) - \alpha_{recup}\left(1 - z_i^2\right)\left(V_{oi}(n) - V_{o\max}\right) \tag{26}$$

Esta equação tem como ponto fixo $V_{oi} = V_{omax}$ e este é um atractor se $0 < \alpha_{recup} < 2$. Agora, assiste-se a uma recuperação da sensibilidade do neurónio z_i aos estímulos que caem dentro do seu campo receptivo.

c) Estudo analítico para o nível fonémico

A componente determinística do campo de vectores total é a soma das componentes de informação e competição. Desta maneira temos

$$g_{det,i}(\mathbf{x}, \mathbf{e}(n)) = g_{info,i}(\mathbf{x}, \mathbf{e}(n)) + g_{comp,i}(\mathbf{e}(n)) =$$
$$= \left(\beta_i(\mathbf{x}) + \gamma_0 \gamma_{ij} e_j^2(n)\right)\left(e_i^3(n) - e_i(n)\right) \ [-1,1] \bmod 2 \text{ para } i = 1,2 \wedge i \neq j \tag{27}$$

Esta equação define pontos fixos em $\mathbf{e}_{fp} = [0 \quad 0]^T$, $[0 \quad \pm 1]^T$, $[\pm 1 \quad 0]^T$, $[\pm 1 \quad \pm 1]^T$ para todos os valores de a_k detectados no nível auditivo e $\left[\pm\sqrt{\frac{-\beta_2\left(\mathbf{x}_{fp}^{(k)}\right)}{\gamma_0\gamma_{21}}} \quad \pm\sqrt{\frac{-\beta_1\left(\mathbf{x}_{fp}^{(k)}\right)}{\gamma_0\gamma_{12}}}\right]^T$ se $a_k \in]a_{c1}, a_{c2}[$.

Segundo a teoria linear de estabilidade temos que:

- Uma condição necessária para $\mathbf{e}_{fp} = [0 \quad 0]^T$ ser um atractor é

$$0 < t_e < 2 \tag{28}$$

- Uma condição necessária para $\mathbf{e}_{fp} = [0 \quad \pm 1]^T$ ser um possível estado estável é

$$a_{c1} > \frac{(h - t_e - \gamma_0\gamma_{12})a_{max} + (1 + t_e - h)a_{min} + (w_0 - t_e - 1)a_{c2}}{w_0 - t_e - \gamma_0\gamma_{12}}$$
$$\vee$$
$$a_{c1} > \frac{(h - t_e - \gamma_0\gamma_{12} - 2)a_{max} + (t_e - h)a_{min} + (w_0 - t_e)a_{c2}}{2 + w_0 - t_e - \gamma_0\gamma_{12}} \tag{29}$$

- Uma condição necessária para $\mathbf{e}_{fp} = [\pm 1 \quad 0]^T$ ser um possível estado estável é

$$a_{c1} > \frac{(h - t_e - 1)a_{max} + (\gamma_0\gamma_{21} + t_e - h)a_{min} + (w_0 - t_e - \gamma_0\gamma_{21})a_{c2}}{h - t_e - 1}$$
$$\vee$$
$$a_{c1} > \frac{(h - t_e)a_{max} + (\gamma_0\gamma_{21} + t_e - h - 2)a_{min} + (2 + w_0 - t_e - \gamma_0\gamma_{21})a_{c2}}{w_0 - t_e} \tag{30}$$

- Uma condição suficiente para que $\mathbf{e}_{fp} = [\pm 1 \quad \pm 1]^T$ sejam repulsores é

$$a_{c1} > \frac{(h - t_e - \gamma_0\gamma_{12} - 1)a_{max} + (1 + \gamma_0\gamma_{21} + t_e - h)a_{min} + (w_0 - t_e - \gamma_0\gamma_{21} - h)a_{c2}}{w_0 - t_e - \gamma_0\gamma_{12} - 1}$$
$$\vee$$
$$a_{c1} > \frac{(h - t_e - \gamma_0\gamma_{12})a_{max} + (\gamma_0\gamma_{21} + t_e - h)a_{min} + (w_0 - t_e - \gamma_0\gamma_{21})a_{c2}}{w_0 - t_e - \gamma_0\gamma_{12}} \tag{31}$$

- Os pontos fixos $\left[\pm\sqrt{\frac{-\beta_2\left(\mathbf{x}_{fp}^{(k)}\right)}{\gamma_0\gamma_{21}}} \quad \pm\sqrt{\frac{-\beta_1\left(\mathbf{x}_{fp}^{(k)}\right)}{\gamma_0\gamma_{12}}}\right]^T$ existem apenas se

$$a_k \in \left]\frac{(t_e - h)a_{max} + (w_0 - t_e)a_{c1}}{w_0 - h}, \frac{(t_e - h)a_{min} + (w_0 - t_e)a_{c2}}{w_0 - h}\right[$$

e no nosso espaço de fase são repulsores.

D. Parâmetros do modelo

a) Estabelecimento dos parâmetros para o nível fonémico

Os parâmetros do modelo a ser fixados são: a_{inf}, a_{sup}, a_{c1}, a_{c2}, γ_0, γ_{12}, γ_{21}, w_0, t_e, h, $Q \in N_{1\pi}$. Os critérios que se seguem são úteis para a escolha de um conjunto básico de valores:

- O VOT deve ser transposto para o parâmetro a do modelo de modo a que num extremo da escala o sistema é monoestável, no outro extremo é também monoestável e no meio o sistema é biestável, ou seja
  - regimes monoestáveis para $a \in [a_{inf}, a_{c1}]$ e $a \in [a_{c2}, a_{sup}]$ e
  - regime biestável para $a \in [a_{c1}, a_{c2}]$

- A escolha deve ser tal que implique de algum modo uma simetria. Podemos escolher por exemplo, $a_{inf} = -1$, $a_{sup} = 1$, $a_{c1} = -0.5$, $a_{c2} = 0.5$ e $\gamma_{12} = \gamma_{12} = 1$. A simetria imposta por esta escolha de valores, para estes parâmetros, torna a escolha dos outros parâmetros mais elegante.

- Por facilidade de implementação nós usamos dinâmica discreta no tempo. Um preço a pagar por esta facilidade é que a escala de tempo não é fixa. Isto acontece porque o tempo correspondente a uma iteração, no computador, é arbitrário. Deste modo, as constantes de relaxação no modelo não têm uma base fisiológica. A constante de tempo de relaxação, "τ_{rel}", pretendida para cada atractor deve ser relativamente pequena. Escolhemos $\tau_{rel} \approx 5$ porque este valor implica uma relação razoável entre o tempo de relaxação e o tempo de uma iteração no computador.

- Devem ocorrer transições estocásticas, isto é, queremos que ocorram transições perceptuais espontâneas devido a flutuações nas variáveis perceptuais. Isto é importante porque os estímulos ambíguos conduzem à biestabilidade e nós queremos modelar este comportamento.

- Do critério anterior determinamos que o desvio padrão (*SD*), ou variabilidade, permitida para as variáveis nos estados estacionários deve ser aproximadamente 0.1, mas pode variar um pouco. Este valor foi escolhido empiricamente. Para $SD \approx 0.1$, as transições estocásticas nas simulações aparecem com uma probabilidade adequada.

Após escolhidos os parâmetros a, a_{c1}, a_{c2}, γ_{12}, γ_{21}, o tempo de relaxação e a variabilidade em regime estacionário para as variáveis perceptuais, todos os outros parâmetros do modelo podem ser estimados. Neste processo, tomamos como base alguns argumentos que fundamentam as acções realizadas.

Argumentos:

1. Condições para regimes estáveis e instáveis:

A condição necessária para que $\begin{bmatrix}0 & \pm 1\end{bmatrix}^T$ seja um estado estável é dada por (29). Substituindo os parâmetro já estabelecidos nesta equação podemos reescreve-la da seguinte forma

$$a_{c1} > \frac{0.5w_0 - 2.5t_e + 2h - \gamma_0 - 1.5}{w_0 - t_e - \gamma_0}$$

$$\vee$$

$$a_{c1} > \frac{0.5w_0 - 2.5t_e + 2h - \gamma_0 - 2}{2 + w_0 - t_e - \gamma_0} \qquad (32)$$

A condição necessária para que $\begin{bmatrix}\pm 1 & 0\end{bmatrix}^T$ seja um estado estável é dada por (30). Substituindo os parâmetro já estabelecidos nesta equação obtemos

$$a_{c1} > \frac{0.5w_0 - 2.5t_e + 2h - 1.5\gamma_0 - 1}{h - t_e - 1}$$

$$\vee$$

$$a_{c1} > \frac{0.5w_0 - 2.5t_e + 2h - 1.5\gamma_0 + 3}{w_0 - t_e} \qquad (33)$$

A condição necessária para $\begin{bmatrix}0 & 0\end{bmatrix}^T$ ser um estado estável é dada por (6.22).

A condição suficiente para que $\begin{bmatrix}\pm 1 & \pm 1\end{bmatrix}^T$ seja um estado instável toma agora a seguinte forma

$$a_{c1} > \frac{0.5w_0 - 2.5t_e + 1.5h - 1.5\gamma_0 - 2}{w_0 - t_e - \gamma_0 - 1}$$

$$\vee$$

$$a_{c1} > \frac{0.5w_0 - 2.5t_e + 2h - 2.5\gamma_0}{w_0 - t_e - \gamma_0} \qquad (34)$$

2. Tempo de relaxação

O tempo de relaxação para um ponto fixo é dado por

$$\tau_{rel} \approx \frac{1}{|\lambda_A|} \qquad (35)$$

onde $|\lambda_A|$ representa o maior valor próprio da matriz Jacobiana $\mathbf{A}$, do campo de vectores. Como escolhemos $\tau_{rel} \approx 5$ vem que $|\lambda_A| \approx 0.2$. Isto implica que o valor próprio correspondente associado à equação do mapa dinâmico é

$$\lambda = 1 - |\lambda_A| \approx 0.8 \qquad (36)$$

3. Relaxação para $\begin{bmatrix}0 & 0\end{bmatrix}^T$

Neste ponto fixo os valores próprios de **I+A**, onde **A** é a matriz Jacobiana da equação (15) (campo de vectores para o nível fonémico) são dados por

$$\lambda_i = 1 - t_e \quad \text{para } i = 1,2 \qquad (37)$$

Agora como para este estado pretendemos $\tau_{rel} \approx 5$ das equações (35) e (36) tiramos que

$$\lambda_{max} = 0.8 \qquad (38)$$

Como $\lambda_1 = \lambda_2 = \lambda_{max}$ da equação (37) obtemos

$$t_e = 0.2 \qquad (39)$$

que satisfaz a condição (28).

4. $[\pm 1,0]^T$ deve ser estável para $a_k > a_{c1}$

Neste ponto os valores próprios de **I+A** são

$$\lambda_1 = 1 - 2\beta_1(a_k) \qquad (40)$$

$$\lambda_2 = 1 - 2\beta_2(a_k) - \gamma_0 \qquad (41)$$

Em $a_k = a_{c1}$ temos um ponto de bifurcação pois para este valor do parâmetro auditivo o estado $[\pm 1,0]^T$ torna-se instável. Se escolhermos para esta instabilidade o tipo fonte (*source*), então devemos fazer com que em $a_k = a_{c1}$ λ_1 e λ_2 atravessem simultaneamente o valor 1. Isto implica que

$$\beta_1(a_{c1}) = 0 \quad \wedge \quad \beta_1(a_{c1}) = \gamma_0 \qquad (42)$$

De seguida substituindo (17) em (42) obtemos as seguintes relações

$$h = t_e \qquad (43)$$

$$\tfrac{2}{3}w_0 - \tfrac{2}{3}h = \gamma_0 \qquad (44)$$

.A relaxação mais rápida deve ocorrer nos extremos do espaço definido pela camada X do nível auditivo. Para o estado $[\pm 1,0]^T$ a relaxação mais rápida ocorre para $a_k = a_{sup} = 1$. para este valor do parâmetro auditivo o ponto fixo $[\pm 1,0]^T$ é o único estado estável. Se escolhermos para a relaxação mais rápida $\tau_{rel} = 5$, de (35), (36) e (40) temos que

$$w_0 = t_e + 0.1 \qquad (45)$$

Finalmente os parâmetros h, w_0 e t_e podem ser determinados. Das equações (39) e (43) vem que $h = 0.2$, em seguida substituindo t_e em (45) obtemos $w_0 = 0.3$ e por fim substituindo w_0 e t_e em (44) tira-se γ_0 que pode ser sintonizado para o valor $\gamma_0 = 0.1$.

5. Ruído

A adição de ruído ao mapa dinâmico não altera muito a dinâmica do sistema. Quando há um ponto fixo estável o ruído provoca uma dispersão dos valores das variáveis perceptuais em torno do ponto fixo. A largura da distribuição dos valores das variáveis perceptuais, quando a grandeza do ruído não é muito elevada, pode ser caracterizada pelo desvio padrão que é dado por:

$$SD = \sqrt{\frac{Q\tau_{rel}}{2}} \tag{46}$$

Por convenção $\sqrt{Q}$ designa-se por pré-factor do termo de ruído (força estocástica) e é o desvio padrão do ruído. Tal como já foi mencionado, a variabilidade permitida para as variáveis perceptuais nos regimes estacionários é aproximadamente de 0.1., $SD \approx 0.1$, consequentemente obtemos a partir de (46) $Q \approx 0.004$. Deve-se salientar que a escolha destes valores não é muito rigoroza sendo possível variá-los dentro de certos limites.

6. Número de iterações

Para os valores escolhidos o tempo de relaxação dos estados mais estáveis é de aproximadamente 5 e por norma o número de iterações é escolhido a partir de

$$N_{itc} > 10\tau_{rel} \tag{47}$$

O conjunto de parâmetros $w_0 = 0.3$, $\gamma_0 = 0.1$, $t_x = 0.2$, $h = 0.2$, $\gamma_{12} = \gamma_{21} = 1$ vai ser usado como um conjunto de referência e os parâmetros Q e N_{itc} vão ser variados. Estas variações permitem investigar o papel do nível de ruído e das escalas temporais no desempenho perceptual.

b) Estabelecimento dos parâmetros para o nível auditivo

Um critério muito importante a ter em consideração ao estimar um conjunto de valores para os vários parâmetros que ocorrem no nível auditivo é que este nível deve relaxar mais rápidamente que o nível fonémico.

b1) Determinação dos parâmetros para o campo dinâmico de neurónios

Os parâmetros a ser fixados são N_x, l_r, γ_x, Q_x e N_{itex}. A seguir enumeramos alguns critérios que ajudam a escolher um conjunto básico de valores.

- Foi feita uma amostragem do espaço definido pelo parâmetro a, desde o limite inferior $a_{inf} = -1$ até $a_{sup} = 1$ a intervalos regulares de $\Delta a = 0.2$. Por conseguinte temos 11 neurónios no campo dinâmico, isto é $N_x = 11$.

- O tempo de relaxação para cada um dos atractores deve ser inferior ao tempo de relaxação do nível fonémico. No capítulo anterior fixámos que o menor tempo de relaxação para o nível fonémico era $\tau_{rel,fonémico} \approx 5$, pelo que deveremos ter $\tau_{rel,x} < 5$.

- O desvio padrão (SD), ou variabilidade, permitido para as saídas dos neurónios nos estados estacionários deve ser $SD \leq 0.01$. Este critério é importante porque as saídas dos neurónios x_i são directamente a entrada dos detectores fonémicos.

Até agora escolhemos os parâmetros N_X, a variabilidade em regime estacionário para as saídas dos neurónios x_i e impusemos limites ao tempo de relaxação. A seguir os restantes parâmetros do campo podem ser estimados. Nesta tarefa, tomamos alguns argumentos que fundamentam os vários passos.

Argumentos:

1. Condições para regimes estáveis:

A condição necessária para que $\mathbf{x}_{fp}^{(0)} = (0,0,\ldots,0)_{N_x}$ seja um estado estável é dada por (22). Isto deve acontecer quando $y_i = 0 \ \forall\, i = 1,\ldots,N_y$. Substituindo (10) em (22) obtem-se que o parâmetro l_r deve obedecer a

$$0 < l_r < 2 \tag{48}$$

A condição necessária para que $\mathbf{x}_{fp}^{(k)} = (0,\ldots,0,\pm 1,0,\ldots,0)_{N_x}$, detecção com sucesso do valor a_k, seja um estado estável é dada por (23). Esta condição deve ser satisfeita quando na camada Y existe pelo menos um neurónio y_i ligado. Se substituirmos (10) em (23) para esta situação podemos reescrever esta última equação da seguinte forma

$$-\gamma_x < l_r < 2 - \gamma_x \ \wedge \ 0 < l_r < 1 \tag{49}$$

2. Tempo de relaxação

O tempo de relaxação para um ponto fixo é dado por

$$\tau_{rel} \approx \frac{1}{|\lambda_A|} \tag{50}$$

onde λ_A representa o maior valor próprio da matriz Jacobiana, $\mathbf{A}$, do campo de vectores (6).

Os valores próprios da matriz Jacobiana do campo de vectores estão relacionados com os valores próprios da equação do mapa dinâmico pela seguinte equação

$$\lambda_i = 1 - \lambda_{A,i} \tag{51}$$

3. Relaxação para $\mathbf{x}_{fp}^{(0)} = (0,0,\ldots,0)_{N_x}$

Neste ponto fixo os valores próprios da equação do mapa dinâmico (5) são

$$\lambda_i = 1 - l_r \ \ \forall\ i = 1,\ldots,N_x \tag{52}$$

De (51) e (52) tiramos que

$$\lambda_{A,i} = l_r \ \ \forall\ i = 1,\ldots,N_x \tag{53}$$

Substituindo (53) em (50) obtemos que o tempo de relaxação para o estado em que todos os neurónios x_i estão desligados é dado por

$$\tau_{rel} = \frac{1}{l_r} \tag{54}$$

Para este estado podemos escolher, por exemplo, um tempo de relaxação de 2, o que pela equação (54) implica que $l_r = 0.5$. Este valor para o parâmetro l_r satisfaz a condição de estabilidade (48).

4. relaxação para $\mathbf{x}_{fp}^{(k)} = (0,\ldots,0,\pm1,0,\ldots,0)_{N_x}$

Para este ponto fixo os valores próprios da equação vectorial do mapa dinâmico são dados por

$$\lambda_i = \begin{cases} 1-(l_\varepsilon+\gamma_x) & \text{se } y_i = 0 \\ 1-2(1-l_\varepsilon) & \text{se } y_i = 1 \end{cases} \quad \forall\, i = 1,\ldots,N_x \tag{55}$$

Agora, a partir das equações (51) e (55) obtemos a expressão para os valores próprios da matriz Jacobiana do campo de vectores dado por (6.6)

$$\lambda_{A,i} = \begin{cases} l_\varepsilon+\gamma_x & \text{se } y_i=0 \\ 2(1-l_\varepsilon) & \text{se } y_i=1 \end{cases} \quad \forall\, i = 1,\ldots,N_x \tag{56}$$

No ponto anterior escolhemos $l_\varepsilon = 0.5$, se agora escolhermos $\gamma_x = 0.9$ as condições (47) e (49) são satisfeitas. Introduzindo estes valores em (56) obtemos que o maior valor próprio toma o valor 1.4 o que determina um tempo de relaxação que é apróximadamente 0.9.

5. Ruído

As flutuações estocásticas nas saídas dos neurónios x_i em regime estacionário devem ser suficientemente pequenas para que o desempenho do nível fonémico não seja afectado. Por exemplo, na situação em que não há estímulo acústico o estado estável do campo é aquele em que todos os neurónios tem as suas saídas a zero, mas devido à adição de ruído observam-se flutuações em torno de zero; se estas flutuações forem suficientemente elevadas podem activar o nível fonémico e induzir à detecção de um fonema. Um dos critérios que adoptamos estabelece que a máxima variabilidade permitida em regime estacionário é 0.01. Deste critério e da equação (46) tiramos então que

$$Q_x \le \frac{2\times SD_{max}^2}{\tau_{rel}} \tag{57}$$

fazendo agora $SD_{max} = 0.01$ e $\tau_{rel} = 2$ (maior tempo de relaxação) em (57) obtemos que $Q_{x,max} \le 0.0001$. Nas simulações fixamos $Q_x = 0.0001$.

6. Número de iterações

O estado que demora mais a relaxar tem um tempo de relaxação de 2, por conseguinte de (47) deduzimos que o número de iterações a escolher deve ser superior a 20. Desta maneira podemos escolher, por exemplo, $N_{iter} = 30$.

b3) Estabelecimento do limiar de activação dos neurónios y_i

O limiar de activação dos neurónios y_i é determinado pelo parâmetro l_y. O critério de escolha de um valor para este parâmetro não é muito rigorosa. As unicas condições a satisfazer são que, o valor de l_y deve ser suficientemente grande para que na ausência de estimulação acústica quaisquer flutuações nos neurónios de entrada não dê origem à detecção de um estímulo acústico, por outro lado, deve ser suficientemente pequeno para que sob os efeitos de adaptação e perante um estímulo acústico na entrada os neurónios y_i excitem os neurónios x_i. Esta ultima condição é necessária porque não queremos que o sistema fique 'surdo', pois nas experiências com sujeitos observa-se que por mais fortes que sejam os efeitos de adaptação estes ouvem sempre alguma coisa. Para o presente propósito fixamos $l_y = 0.1$.

b4) Determinação dos parâmetros para a dinâmica da adaptação selectiva

Perante um estímulo acústico a camada X do nível auditivo relaxa para o estado $\mathbf{x}_{fp}^{(k)} = (0,\ldots,0,\pm1,0,\ldots,0)_{N_x}$ com uma constante de relaxação que é de aproximadamente 0.9. E, todos os neurónios z_i que têm o estímulo acústico dentro do seu campo receptivo sofrem efeitos de adaptação. Estes consistem na diminuição do parâmetro V_{0_i} na direcção de V_{0min} como é descrito pela equação (25), com uma constante de relaxação que é dada por

$$\tau_{adap} \approx \frac{1}{\alpha_{adap}} \tag{58}$$

Nas experiências de adaptação selectiva com sujeitos verifica-se que a apresentação de um estímulo acústico uma única vez não produz quaisquer efeitos observáveis de adaptação. Desta maneira, para que os resultados do modelo coincidam com os resultados observados nas experiência com sujeitos, devemos adoptar como critério

$$\tau_{adap} > 0.9 \tag{59}$$

Podemos escolher por exemplo $\tau_{adap} \approx 3\times 0.9 = 2.7$, pelo que de (58) podemos escolher $\alpha_{adap} = 0.35$.

Para os neurónios da camada Z que não têm o estímulo acústico corrente dentro do seu campo receptivo assiste-se a uma recuperação da sensibilidade aos estímulos que pertencem ao seu campo receptivo. O fenómeno de recuperação traduz-se segundo a equação (26) num aumento do valor do parâmetro V_{0_i} na direcção de V_{0max} com uma constante de tempo que é dada por

$$\tau_{recup} \approx \frac{1}{\alpha_{recup}} \tag{60}$$

Um aspecto importante a ter em mente é que para que os efeitos de adaptação selectiva sejam sentidos ao longo das simulações, a recuperação de V_{0_i} tem que ser muito mais lenta que a sua diminuição. Por conseguinte, devemos ter a seguinte condição

$$\tau_{recup} >> \tau_{adap} \tag{61}$$

Se escolhermos, por exemplo α_{recup}=0.004, de (60) obtemos τ_{recup}≈200 que satisfaz a condição (61).

Para finalizar a escolha de um conjunto de valores para os parâmetros do nível auditivo falta ainda fixar os valores dos parâmetros V_{0max} e V_{0min}. Para estes parâmetros podemos escolhar os valores 1 e 0.15 respectivamente. A escolha de V_{0min}=0.15 > l_y evita que o sistema fique 'surdo' a um estímulo acústico por mais fortes que sejam os efeitos de adaptação.

III. ENSAIOS EXPERIMENTAIS

A. Avaliação do modelo em relação às propriedades experimentais observadas para a percepção categorial

Para as simulações construimos um pequeno simulador em matlab. A ideia que nós temos em mente é usar este simulador como os psicólogos usam os seus sujeitos quando efectuam experiências da percepção categorial.

A.1 Simulação das tarefas de identificação e predicção da discriminação

Método

As tarefas envolvidas na simulação da experiência psicológica de identificação são:

i) Preparar uma lista pseudo-aleatória de valores para o parâmetro a (representando a lista experimental de estímulos acústicos). Pseudo-aleatória significa que a ordem é aparentemente aleatória, mas cada valor diferente do parâmetro aparece o mesmo número de vezes. O uso de um número de vezes igual para cada valor do parâmetro a evita fenómenos de polarização. Para o presente propósito usámos onze valores diferentes para o parametro a que vão desde um extremo do regime ao outro. Cada um destes valores é repetido dez vezes.

ii) Corre-se o simulador sobre esta lista de estímulos. Em cada caso a condição inicial para o estímulo seguinte é a condição final da execução anterior (história perceptual).

iii) Para cada valor do parâmetro a toma-se nota do estado final. A condição lógica que classifica se o estado final é o reconhecimento do fonema 1 ou fonema 2 baseia-se na regra do máximo já descrita. Todos os resultados são contados.

iv) Dos resultados obtidos faz-se a curva de identificação, que mostra o número de vezes que cada estímulo foi reconhecido como fonema 1 (/ba/) ou fonema 2 (/pa/).

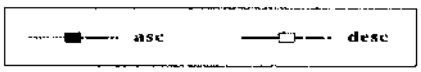
Esta experiência foi feita para vários valores dos parâmetros N_{ite} e Q.

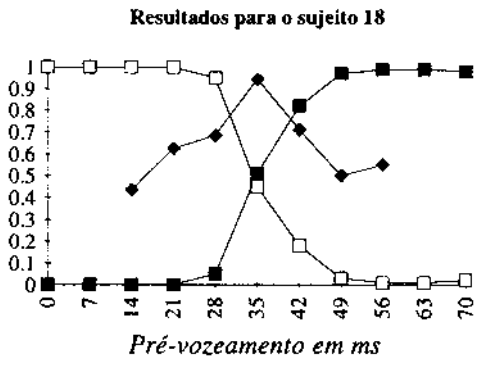
As curvas de discriminação são preditas a partir dos resultados da tarefa de identificação usando a teoria de que a discriminação é puramente baseada na categorização, isto é, assumimos o pressuposto extremo de que os sujeitos só conseguem discriminar sons de fala a que atribuem categorias fonémicas diferentes [13].

Resultados e discussão

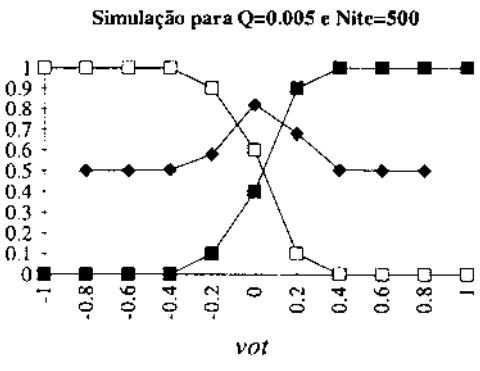
Um dos resultados das simulações da tarefa de identificação e curvas de discriminação preditas para dois passos é apresentado gráficamente na figura 8a). Na figura 8b) é apresentado gráficamente resultados análogos obtidos em experiências com um dos sujeitos. Na apresentação dos resultados cada ponto na curva de discriminação é relativo à comparação entre o estímulo imediatamente antes e depois dele.

Nesta figura observa-se que há uma partição bem definida dos estímulos em duas categorias distintas, com a fronteira categorial (ou fonémica), zona em que as duas categorias são equiprováveis, muito perto de $a = 0$ que corresponde ao ponto médio do *continuum*. Note-se a excelente concordância com os resultados obtidos com o sujeito, e em particular a localização da fronteira fonémica que para este ocorre muito perto de 35 ms de pré-vozeamento, ponto médio do *continuum* tal como acontece nas simulações.



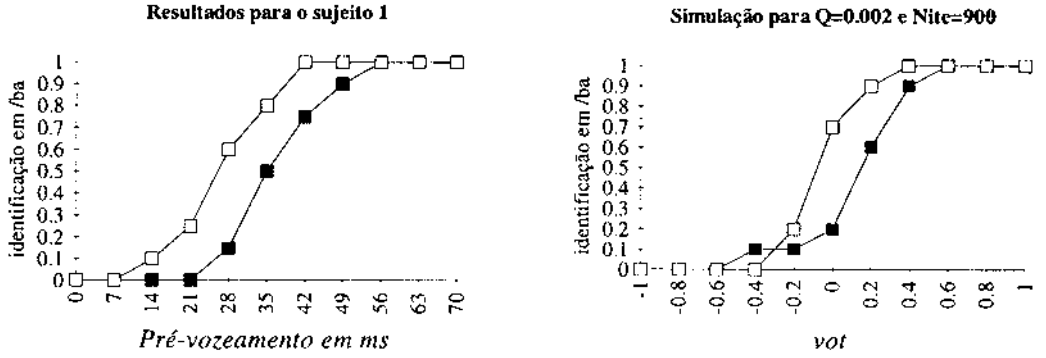


a)



b)

Fig. 8 Curvas de identificação e de discriminação. Em a) Resultados para o sujeito 18. Em b) resultados obtidos por simulação



a) b)

Fig. 9 Padrões de histerese. Em a) padrão de histerese para o sujeito 1. Em b) padrão de histerese obtido por simulação

Das curvas de discriminação emerge que estas exibem valores de 0.5 para estímulos dentro de cada regime monoestável (onde apenas é possível um percepto), e tem picos dentro do regime biestável, onde os dois perceptos são possíveis. Isto está de acordo com o facto de que com diferenças acústicas similares, a discriminação de sons entre membros da mesma categoria fonémica é quase nula (nível do acaso) enquanto que a discriminação é relativamente mais perspicaz entre estímulos que residem em lados opostos da fronteira fonémica. É o que acontece, por exemplo, para a comparação entre -0.2 e 0.2 no modelo eque equivale à comparação entre -28ms e -42ms de VOT nas experiências com os sujeitos.

A.2 Simulações de histerese

Método

As experiências de histerese são uma variante simples da experiência clássica da percepção categorial. A principal diferença é que usamos não uma lista pseudo-aleatória de estímulos mas sim uma lista ordenada. São consideradas duas séries de estímulos, uma ascendente que consiste numa série de estímulos que vão desde o extremo inferior até ao extremo superior do *continuum*, e outra ascendente que é a mesma série mas ordenada de forma inversa. Cada uma destas séries é constituída por onze valores diferentes para o parâmetro *vot*, e cada série é repetida dez vezes.

Resultados e discussão

Na figura 9 são apresentados os resultados experimentais obtidos para um dos sujeitos intervenientes nestas experiências, e é ilustrado um dos resultados da simulação. A figuras 9b) mostra que, como era esperado, a apresentação das séries de estímulos induzem o fenómeno de histerese pois o ponto em que é observado uma transição de categoria é diferente para a ordem ascendente e para a ordem descendente, em cada caso a transição perceptual é 'atrasada' em cada série tal que é obtida uma região de sobreposição. Esta região é chamada a magnitude da histerese. Em termos da dinâmica do modelo isto pode ser compreendido do seguinte modo: À medida que uma série é apresentada sequencialmente no tempo desde um extremo a outro, acontece que o percepto inicial persiste no regime biestável até que o parâmetro *a* atinge a fronteira mais distante deste regime onde o percepto perde a estabilidade. Se a série de estímulos é apresentada na ordem inversa acontece o oposto. Consequentemente os pontos na série em que ocorre a transição perceptual depende da direcção em que a série é apresentada.

B. Avaliação do modelo em relação aos efeitos da adaptação

Os efeitos de adaptação na percepção de sons de fala aparecem em relação à localização da fronteira categorial ao longo do *continuum* de estímulos. Para estudar a adaptação no modelo realizámos uma simualação da experiência de contraste, que como iremos ver produz efeitos idênticos ao paradigma da adaptação selectiva.

Simulação da experiência de contraste

Tipicamente, no paradigma da adaptação selectiva os sujeitos são expostos a uma longa sequência de adaptação, que consiste no estímulo adaptador repetido várias vezes. No fim deste período, é apresentada uma série aleatória de estímulos para serem classificados. O resultado é um salto na fronteira fonémica na direcção do estímulo adaptador.

A experiência de contraste consiste num simples procedimento em que a longa sequência de adaptação é substituída por um único estímulo, designado por estímulo de contexto, seguido por um único estímulo de teste. Para os estímulos de contexto, tomam-se estímulos que são considerados pelos sujeitos exemplares claros das categorias que representam, enquanto que os estímulos de teste são seleccionados na vizinhança da fronteira categorial para que tenham um carácter bastante ambíguo. Neste tipo de experiência os sujeitos revelam uma grande tendência para identificarem o item de teste na categoria oposta ao do estímulo de contexto. Um resultado que é semelhante ao efeito típico da adaptação selectiva. Para Diehl e seus colaboradores [14] esta semelhança sugere que os efeitos de adaptação e contraste podem ser

produzidos pelos mesmos processos. Estes autores apresentam alguns estudos que apoiam esta hipótese.

Método

Experiência A

Para a simulação da experiência de contraste foram realizados os seguintes passos [14]:

i) Para estímulos de contexto seleccionaram-se dois estímulos com valores de *VOT* -0.4 e 0.4. O estímulo -0.4 é principalmente identificado como fonema 2, enquanto que 0.4 é essencialmente classificado como fonema 1.

ii) Escolheu-se para estímulo de teste um valor junto à fronteira categorial, i.e. o valor 0, que possui um carácter muito ambíguo. Teóricamente este estímulo tem uma probabilidade de 50 % de ser classificado em qualquer uma das duas categorias.

iii) Sem os efeitos de adaptação activos, cada combinação dos estímulos de contexto e de teste foi apresentada 10 vezes ao modelo para serem classificados. As condições iniciais

para os neurónios do nível fonémico foram variadas aleatóriamente ao longo de todo o intervalo.

iv) Repetiu-se o passo anterior mas agora com a dinâmica da adaptação activa no modelo.

v) Registaram-se os resultados obtidos.

Experiência B

Repetiu-se essencialmente a experiência A mas agora para todos os restantes estímulos do *continuum*, e o passo iii) foi substituído pelo paradigma aleatório da percepção categorial estando incorporados no modelo efeitos de adaptação.

Resultados e discussão

Os resultados obtidos na experiência A são mostrados na tabela 1. E na figura 10 são apresentados graficamente os resultados da experiência B.

Estímulo de contexto		
VOT →	**-0.4**	**0.4**
Sem adaptação	60%	60%
Com adaptação	100%	0%

Tabela 1 Percentagem de vezes que o estímulo 0 é identificado como fonema 1, para dois estímulos de contexto e para o modelo sem e com efeitos de adaptação. Simulações para Q=0.005 e N_{ite}=300

Cada valor na tabela 1 representa a percentagem de vezes que o estímulo de teste, VOT=0, é identificado na categoria fonémica 1, para cada um dos estímulos de contexto e para o modelo sem e com efeitos de adaptação. Nesta tabela podemos ver que o estímulo de teste, na situação em que o sistema não exibe efeitos de adaptação, foi classificado 60% das vezes na classe fonémica 1, independentemente do estímulo de contexto. Teóricamente este estímulo de teste tem uma probabilidade de 50% de ser classificado em qualquer uma das duas categorias, acontece porém, que devido ao facto dos tempos de computação serem elevados, este estímulo foi apresentado ao modelo apenas 10 vezes, das quais 6 vezes foi classificado na categoria fonémica 1 e as restantes 4 vezes na categoria fonémica 2. Como vemos temos uma amostragem pequena que se traduz num desvio aparentemente grande dos 50%. Quando o sistema incorpora efeitos de adaptação este padrão de resultados é completamente alterado, o que revela que o efeito do estímulo de contexto na adaptação é muito significativo. Quando o sistema tem a sua dinâmica de adaptação activa e o estímulo de contexto é *VOT*=-0.4, observa-se que o estímulo de teste é sempre classificado na categoria fonémica 1, independentemente das condições iniciais para o nível fonémico. Este resultado é o reflexo de um salto na fronteira fonémica na direcção do estímulo de contexto, que é o responsável pelos efeitos de adaptação. Analogamente, quando o estímulo de contexto é *VOT*=0.4, o estímulo de teste passa a ser inequivocamente "percebido" como fonema 2, em consequência dos efeitos de adaptação que provocam uma mudança na posição da fronteira fonémica na direcção do extremo superior do *continuum*, i.e. na direcção do estímulo de contexto.

Estes resultados, obtidos por simulação, estão em perfeita concordância com os padrões de comportamento observados em experiências com sujeitos e relatados em [14].

Vejamos agora os resultados da experiência de contraste para os restantes estímulos do *continuum* que são ilustrados graficamente na figura 10.

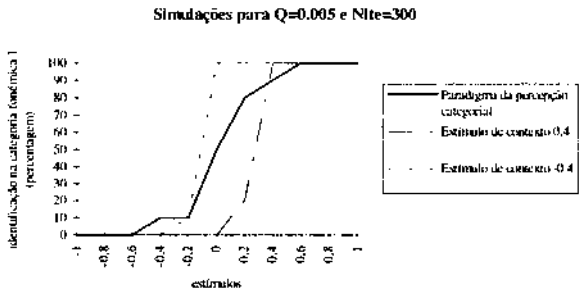


Fig.10 Curvas de identificação na categoria fonémica 1, obtidas por simulação da experiência de contraste para todos os estímulos ao longo do continuum e por simulação do paradigma da percepção categorial.

Conforme podemos observar na figura10, para o paradigma aleatório da percepção categorial o modelo exibe o padrão clássico esperado para este fenómeno. Nas experiências de contraste, o modelo revela uma elevada tendência para identificar os estímulos de teste na

categoria oposta ao do estímulo de contexto. Em particular, observa-se que o estímulo de contexto *VOT*=-0.4, produz um deslocamento na fronteira na direcção do extremo menos vozeado do *continuum*, enquanto que o estímulo de contexto *VOT*=0.4, produz o efeito contrário. Um resultado que é igual ao efeito típico da adaptação selectiva.

VI. CONCLUSÃO

Neste artigo apresentámos um modelo dinâmico neuronal para a categorização de sons de fala num *continuum* de vozeamento. Mostrámos que no paradigma clássico de identificação emerge o padrão típico da percepção categorial. Quando os estímulos são apresentados em séries ascendentes ou descendentes resulta histerese. Esta é uma forma de memória perceptual que não foi construída explicitamente. Ocorrem saltos na fronteira categorial devido a efeitos de adaptação, que são de natureza inteiramente auditiva. No modelo é também modelada a situação em que não há nada para perceber, i.e. a ausência de um estado perceptivo, o que ocorre na ausência de qualquer estímulo acústico. Comparámos as predições do modelo com os resultados experimentais. Deste estudo podemos concluir, que a percepção categorial da fala pode ser compreendida como o resultado de um processo de competição dentro de uma representação neuronal da informação sensorial. Nesta perspectiva a discretização resultante do fenómeno da percepção categorial é o resultado da formação de estados dinâmicamente estáveis. Estes estados não são imunes aos processos perceptuais subjacentes, pois dependem da história recente do sistema.

AGRADECIMENTOS

Agradecemos à Doutora São Luís Castro, da fac. de Psicologia e Ciências da Educação, a colaboração dispensada tanto na parte teórica como experimental do fenómeno que aqui estudamos.

Este trabalho foi realizado com o apoio da bolsa de mestrado da JNICT no âmbito do programa CIÊNCIA: BM/2780/92-IA.

REFERÊNCIAS

[1] Juscyzk (1986), review article by Peter Juscyzk from Handbook of Perception and Human Performance (Boff, K,Kaufman, L & Thomas, J(eds), 1986).

[2] Repp B. H. (1984), "Categorical Perception: issues, methods, findings", in N.J. Lass (Ed). Speech and Language: Advances in Basic Research and Practice, vol. 10, p. 243-335.

[3] Castro S.L. (1993), "Percepção categorial num continuum de vozeamento em fala natural portuguesa", in EPLP'93 1º Encontro de Processamento da Língua Portuguesa Escrita e Falada, 15-18.

[4] Sawusch J. R., Jusczyk P. (1981), " Adaptation and constrast in the perception of voicing", Journal of Experimental Psychology: Human Perception and Performance 2, p.408 421.

[5] Williams D., Phillips G., Sekuler R. (1986)," Hysteresis in the perception of motion direction as evidenve for neural cooperativity", Nature 324: 253-255 , 1986.

[6] Chang J. J., Julesz B.(1984), "Cooperative phenomena in apparent movement: perception of random-dot cinematograms", Vision Research 24:1781-1788, 1984.

[7] Castro S.L., Barbosa M.F., Bicho E., Schöner G. (1994), "Cooperativity in the categorical perception of speech: experiment and modelling", European Society of cognitive psychology, Lisboa 1994.

[8] Summerfield A.Q.(1975),"Aerodynamic versus mechanics in the control of voicing onset in consonant-vowell syllables", in Speech Perception (no.4),Department od Psychology, Queens University of Belfast, 1975.

[9] Braun M. (1978), "Differencial equations and their applications", Springer Verlag, New York, 1978.

[10] Perko L. (1991), "Differential Equations and Dynamical Systems", Springer Verlag, Berlin.

[11] Stein D.(1989), "Lectures in the sciences of complexity", Ed Daniel Stein, Department of physics, University of Arizona.

[12] Schoner G., Dose M.(1992),"A dynamical system approach to task level system integration used to plan and control autonomous vehicle motion", Robotics and Autonomous Systems 10(1992),p253-267-Elsevier.

[13] Liberman A., Harris K., Hoffman H., Griffith B. (1957), "The discrimination of speech sounds within and across phoneme boudaries", Journal of Experimental Psychology, Vol. 54, No. 5, 1957

[14] Diehl R. L., Kluender K.R., Parker E.M.(1985), "Are selective adaptation and contrast effetes really distinct?", Journal of Experimental Psychology: Human Perception and Performance 11, p.209-220.

Aplicação de Métodos Estruturados na Especificação e Implementação de um Sistema de Comunicação em Tempo Real

José P. Santos, Fernando M. S. Ramos

Resumo- **Em 1990 a Telecom Portugal contratou o INESC (Instituto de Engenharia de Sistemas e Computadores) para desenvolver o protótipo de um Sistema de Televigilância suportado pela RDIS (Rede Digital com Integração de Serviços) no âmbito do apoio às actividades de investigação e desenvolvimento promovidas pela Telecom Portugal relacionadas com a introdução em Portugal da RDIS, projecto que foi concluído em 1994. Este artigo inclui uma breve descrição e discussão sobre a aplicação de técnicas estruturadas de desenvolvimento e especificação de sistemas de informação ao sistema implementado, sendo apresentada a análise e o projecto estruturado do software para processamento em tempo real desenvolvido para o componente mais complexo do sistema a Estação Central de Televigilância.**

Abstract- **In 1990 INESC (Institute of System Engineering and Computers) was contracted by PT-Portugal Telecom - the Portuguese telecom operator - to develop the prototype of a telesurveilling system supported on ISDN (Integrated Services Digital Network), in the context of the R&D activities promoted by PT concerning the introduction of ISDN in Portugal, project that was concluded in 1994. This paper includes a brief survey and discussion on the structured techniques evaluated for the system specification, and presents and discusses the analysis and design processes of the software developed to support the real time functionalities of the system (images alarms,...) focused on the most complex of its components, the Central Telesurveilling Station**

1. DESCRIÇÃO DO SISTEMA IMPLEMENTADO

O Sistema de Televigilância [1] tem como objectivo permitir a visualização numa estação central de imagens obtidas por câmaras de vídeo localizadas em pontos remotos.

A arquitectura do sistema desenvolvido baseia-se em dois tipos de estações: Estação Central (EC) e Estação Remota (ER); ambas as estações baseiam-se em plataformas PC compatível com placas de comunicação RDIS (2B+D) e placas de processamento de imagem.

A comunicação entre a EC e a(s) ER(s) baseia-se no canal B (64Kbps), no modo de comutação de circuitos, para transferir imagens e mensagens de controlo do sistema, enquanto no canal D (16Kbps) são transferidas mensagens sobre os alarmes usando a capacidade UUS - User to User Signalling disponibilizada pela RDIS.

Este sistema permite transferir dois tipos de imagens entre a ER e a EC: imagens em movimento a preto e branco com 144x180 pixel com uma taxa de transferência de oito imagens por segundo, e imagens a cores com 288x360 pixel com uma taxa de transferência máxima de 1 imagem por segundo.

O estabelecimento de ligação entre uma ER e a EC pode ser da iniciativa de qualquer uma delas: a iniciativa é da ER quando um dos seus sensores (intrusão, fogo,...) fôr accionado, e é da EC quando o seu utilizador pretender configurar ou obter imagens de uma ER.

Mesmo quando a ligação entre a EC e uma ER está estabelecida a EC tem a capacidade de receber e processar indicações de alarmes que ocorram nessa ou noutra ER: nesse caso o utilizador é informado dessa(s) ocorrência(s) podendo optar por atender mais tarde ou atender imediatamente esse alarme passando nessa altura a obter imagens dessa ER.

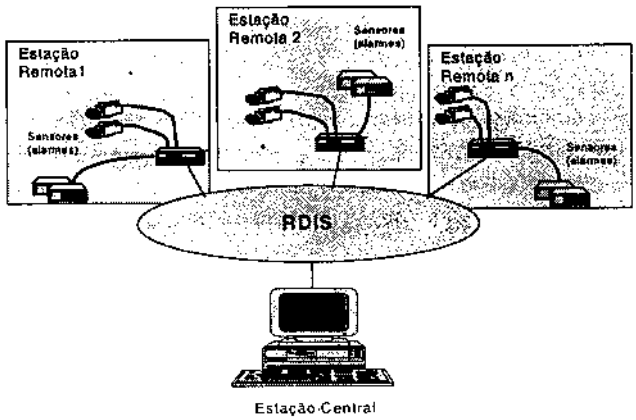


Fig. 1 - Sistema de Televigilância

A partir das funcionalidades do sistema torna-se claro que diversos tipos de ocorrências devem ser tratadas em tempo real pela EC, nomeadamente comunicações bidireccionais (envio de comandos para a(s) ER(s)e recepção de confirmações, alarmes e imagens), processamento dos alarmes, reconstrução de imagens e visualização, bem como interacções com o utilizador. Pelas mesmas razões a ER tem de ser capaz de processar diversas ocorrências em tempo real, nomeadamente comunicações bidireccionais (transmissão de alarmes e

imagens e recepção de comandos) e aquisição e compressão de imagens.

Este artigo descreve e discute o processo de análise, projecto estruturado e implementação da EC.

II. MÉTODOS ESTRUTURADOS PARA DESENVOLVIMENTO DE SOFTWARE

O uso de métodos formais de análise e projecto estruturado aplicados ao desenvolvimento de software é necessário para permitir um desenvolvimento estruturado e uma descrição rigorosa e completa do software a desenvolver.

A análise do sistema inclui a definição do seu interface com o exterior (nomeadamente a lista de todas as ocorrências provenientes do exterior que o sistema tem de tratar incluindo o formato dos dados trocados), os dados internos e dados armazenados no sistema e ainda as operações a efectuar sobre esses dados.

Para a especificação deste sistema foi feito um estudo comparativo de três métodos de análise, com o objectivo de seleccionar aquele que melhor se coadunava com o sistema a desenvolver: DeMarco [2], Gane-Sarson [3] and Yourdon [4].

DeMarco e Gane-Sarson propõem que a primeira etapa da análise de um novo sistema a desenvolver seja a criação do seu Modelo Lógico, isto é, a descrição dos dados e operações a efectuar, recorrendo aos:

Diagrama de Fluxo de Dados (DFD): ferramenta gráfica usada para mostrar os fluxos de dados, armazéns e operações a efectuar sobre os dados (processos); estes diagramas são construidos usando quatro tipos de entidades: fluxos de dados, processos, armazéns, e entidades externas.

Dicionário de Dados (DD): é uma lista organizada com a descrição (formato dos dados, gama de valores que podem assumir,...) dos fluxos de dados, armazéns e entidades externas usadas nos DFDs, permitindo uma melhor compreensão dos mesmos.

Especificação de Processos (EP): como o nome indica a especificação dos processos descreve as operações que cada processo deve efectuar sobre os dados que recebe; para esta especificação e consoante o seu nível de abstracção e tipo de processo podem ser usadas diferentes formalismos, desde o inglês estruturado até aos diagramas de transição de estados.

Para a definição dos armazéns DeMarco propõe o uso de diagramas de estrutura de dados enquanto Gane-Sarson propõe o uso das dependências funcionais com redução à 3FN-Terceira Forma Normal, sendo esta última preferível por permitir um maior nível de organização e consistência, mas obrigando ainda na análise a definir todos os dados necessários a cada armazém.

Na etapa seguinte tanto DeMarco com Gane-Sarson propõem o desenvolvimento do novo Modelo Físico, que inclui a definição do interface homem-máquina e a identificação de vários modelos físicos possíveis; a escolha do modelo a utilizar deverá ser feita com base no critério custo*benefício e avaliando o esforço requerido para o seu desenvolvimento e implementação.

Os métodos propostos por DeMarco e Gane-Sarson são habitualmente referidos como Métodos Clássicos de Análise Estruturada. Estes métodos consomem muito tempo, sobretudo quando o sistema em análise é uma evolução de um sistema já existente, dado que isso obriga também à compreensão e documentação do sistema existente. Por outro lado, estes métodos não incluem uma forma prática de descrição de sistemas em tempo real, e não possuem ferramentas que permitam, com um grau de abstracção adequado, a definição estruturada dos armazéns.

Nos anos 80 autores como McMenamim & Palmer [5], Ward & Meller [6] e em especial Yourdon [4], introduziram novos conceitos que ajudam a ultrapassar as desvantagens atrás enunciadas dos Métodos Clássicos de Análise Estruturada. Nesta nova abordagem, proposta por Yourdon e conhecida como Análise Estruturada Moderna, o processo de análise de um novo sistema começa pela definição do Modelo Essencial, o qual é constituído pelo Modelo Ambiente e pelo Modelo de Comportamento.

No Modelo Ambiente é definido o interface entre o sistema a desenvolver e o universo envolvente, e ainda a lista de acontecimentos, incluindo os dados associados, que o sistema tem de tratar.

No Modelo de Comportamento é descrito o comportamento interno do sistema com base na lista de acontecimentos, sendo desenvolvido um DFD de nível intermédio para cada um desses acontecimentos (*event partitioning*), ao contrário da abordagem top-down usada pelos outros métodos. As ferramentas utilizadas são os DFDs, os DDs, as EPs e uma nova ferramenta:

Diagrama de Entidade-Relacionamento (DER): diagrama que permite a criação de armazéns normalizados e consistentes com base nas entidades do sistema e no relacionamento entre os seus dados associados.

Por outro lado Yourdon propõe para a especificação dos processos o uso de Diagramas de Transição de Estados e das *Pre-post Conditions*:

Diagramas de Transição de Estados (DTE): diagramas que identificam os diferentes estados em que (neste contexto) um processo se pode encontrar e com base nesse estado as operações a exercer sobre os fluxos de dados. A estes processos chamam-se processos de controlo.

Pre-post Conditions (PPC): definem, com um grande nível de abstracção, as operações a efectuar sobre os fluxos de dados por um processo.

Os DTEs fornecem à análise uma ferramenta poderosa para a descrição dos processos de controlo que são essenciais para a especificação de sistemas de tempo real. Os DER permitem a definição dos armazéns com um

elevado nível de abstracção e organização sendo por isso uma importante ferramenta da análise.

No Modelo Ambiente para permitir uma separação clara entre o ambiente envolvente e o sistema em análise é definido o diagrama de contexto, que permite definir claramente qual a parte (o todo ou uma parte) do sistema vai ser objecto de especificação.

No desenvolvimento do modelo de comportamento Yourdon propõe a abordagem *Event Partioning* já referida, o que facilita a análise do sistema dado que orienta a especificação com base nos acontecimentos.

Para a passagem da análise para o projecto estrututrado Yourdon propõe a criação dos seguintes modelos: O Modelo de Implementação do Sistema, que inclui o Modelo do Processador e o Modelo da Tarefa, e ainda o Modelo de Implementação do Programa.

O Modelo do Processador consiste na definição da melhor estratégia de alocação das especificações funcionais criadas através da análise pelos vários processadores/máquinas disponíveis para a implementação do sistema. Por sua vez o Modelo da Tarefa define para cada processador/máquina a(s) tarefa(s) a desempenhar. Cada tarefa fica então com um conjunto de DFDs, armazéns, etc. alocados, sendo então necessário converter esse modelo assíncrono num modelo síncrono, dado a implementação de uma tarefa (em UNIX ou DOS) ser efectuada de uma forma síncrona. Com este objectivo Yourdon propõe a criação do Modelo de Implementação do Programa usando para isso uma ferramenta gráfica conhecida por Diagrama de Estrutura [7].

III. ESTUDO DE UM CASO: APLICAÇÃO DE MÉTODOS ESTRUTURADOS PARA O DESENVOLVIMENTO DE UM SISTEMA DE TELEVIGILÂNCIA SUPORTADO PELA RDIS

A figura 2 apresenta o Modelo Ambiente do sistema em análise, cuja descrição funcional foi resumidamente apresentada na secção I:

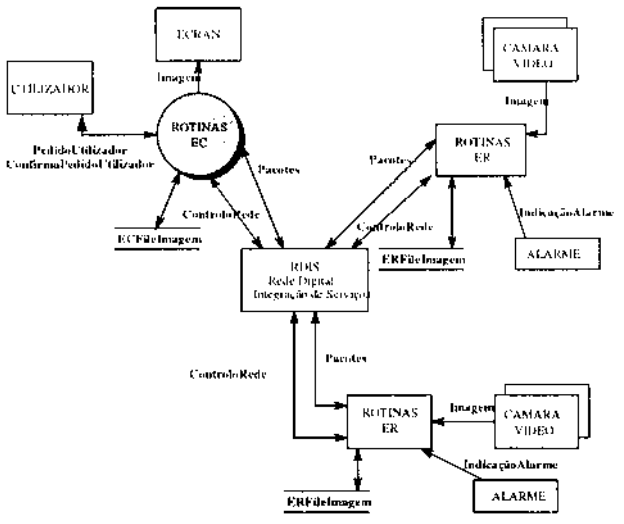


Fig. 2 - Modelo Ambiente do Sistema de Televigilância suportado pela RDIS.

As imagens obtidas a partir de câmaras de vídeo são adquiridas e comprimidas através de placas de aquisição e compressão de imagem em cada ER, após o que são transmitidas para a EC através da placa RDIS, o que envolve o envio de um ou mais pacotes de dados. Se a ligação não se encontrar ainda estabelecida entre a ER e a EC, a ER guarda localmente imagens, para uma eventual visualização posterior, no armazém ERFileImagem.

Na EC as imagens são recebidas da rede através da placa RDIS, e visualizadas após terem sido descomprimidas. O utilizador pode adicionalmente pedir para gravar as imagens na EC, o que será feito no armazém ECFileImagem.

A EC e as ERs podem também trocar outro tipo de informação, tal como comandos de configuração e indicação de alarmes.

Neste artigo apenas é abordada a análise e o projecto da EC, o que está expresso no Diagrama de Contexto apresentado na figura 3.

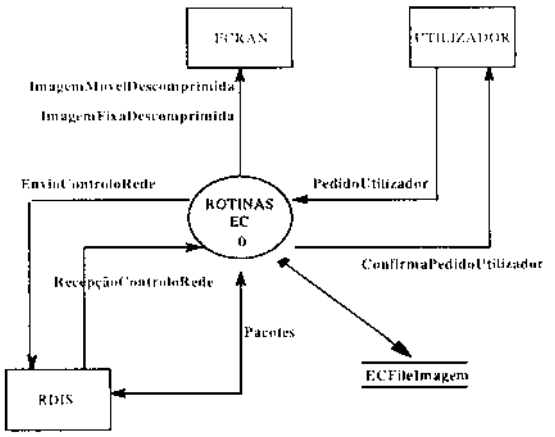


Fig. 3 - Diagrama de Contexto da EC.

Os fluxos de dados recebidos e as correspondentes respostas são descritos no DD. As mensagens recebidas provenientes do utilizador ou da ER não são mais que os acontecimentos que o sistema tem de tratar.

Os acontecimentos gerados pelo utilizador podem ser de quatro tipos diferentes: pedido de configuração da EC, pedido de configuração de uma ER, operação de uma ER, e visualização de imagens na EC. Os acontecimentos recebidos da rede vindos da ER podem ser de dois tipos: indicação de alarme numa ER e recepção de imagens.

Do Modelo Ambiente também faz parte a lista de todos os acontecimentos que o sistema tem de tratar.

O Modelo de Comportamento descreve o comportamento interno do sistema em análise baseando-se na lista de acontecimentos; para cada acontecimento é criado um DFD de nível intermédio (*Event Partitioning*) para permitir definir a forma como cada acontecimento (fluxo de dados) deve ser processado e/ou armazenado. Com este objectivo são utilizadas ferramentas como os DFD, DD e EP. A figura 4 apresenta o DFD de mais alto nível correspondente ao Diagrama de Contexto, que foi contruído com base nos DFDs de nível intermédio.

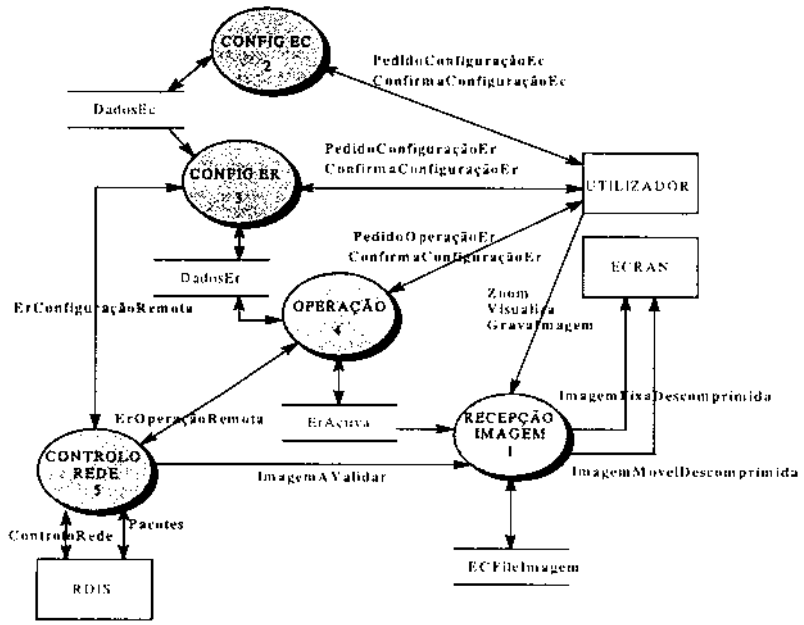


Fig. 4 - DFD 0 da EC.

A figura 5 apresenta exemplos de entradas no DD relacionadas com o DFD0. O DD inclui entradas semelhantes para todas as estruturas de dados existentes no sistema.

ImagemAValidar: receção de uma imagem vinda da ER.
ImagemAValidar= 5555

+ NumeroCâmara	= {byte}1 = 0x01-0x08
+ NumeroFragImagem	= {byte}1 = 0x00-0xff
+ Imagem	=1{byte}26000

Fig. 5 - Um exemplo de uma entrada do DD.

As técnicas usadas na especificação dos processos incluem as *Pré-post Conditions*, Inglês estruturado e Diagramas de Transição de Estados.

A figura 6 mostra como exemplo a especificação do processo 1 - *Recepção Imagem* apresentado no DFD0 (figura 4).

```
BEGIN
IF GravaImagemFixa
        Copia a imagem do ficheiro (default.jpg) para a ficheiro
        indicado
ELSE
BEGIN
    IFImagemAValidar WITH NúmeroCâmara=NumeroCâmaraActiva
        IF NumeroCâmaraActiva= 1 or 2
            Transfer imagem fixa comprimida para a placa de imagem
            Guarda imagem fixa comprimida no ficheiro (default.jpg)
        ELSE (* 2 < NumeroCâmaraActiva < 9 *)
            Transfer imagem móvel comprimida para a placa de
            imagem
        ENDIF

        ENDIF

        IF Show
            Abre file indicada
            Transfer imagem fixa comprimida para a placa de imagem
        ENDIF

        Inicia a descompressão da imagem
        DO WHILE (descompressão em progresso)
        ENDDO
        Transfere imagem da placa de imagem para a memoria
        Visualiza imagem
END
END
```

Fig. 6 - Exemplo da especificação de um processo usando Inglês estruturado.

Depois do processo de análise ter sido concluido foi iniciado o projecto estruturado. Como era suposto a EC ser implementada numa única máquina com um único processador e com base no sistema operativo MS-DOS, não há necessidade de criar o Modelo de Implementação do Sistema, dado todas as especificações criadas na análise serem alocadas a uma única tarefa numa única máquina/processador, sendo apenas necessário criar o Modelo de Implementação do Programa. A figura 8 apresenta uma versão simplificada do Diagrama de Estrutura inicial do Modelo de Implementação do Programa já com um conjunto de refinamentos necessários para evitar problemas de recursividade indirecta e para garantir o processamento adequado dos acontecimentos assíncronos que ocorrem em sistemas de processamento em tempo real.

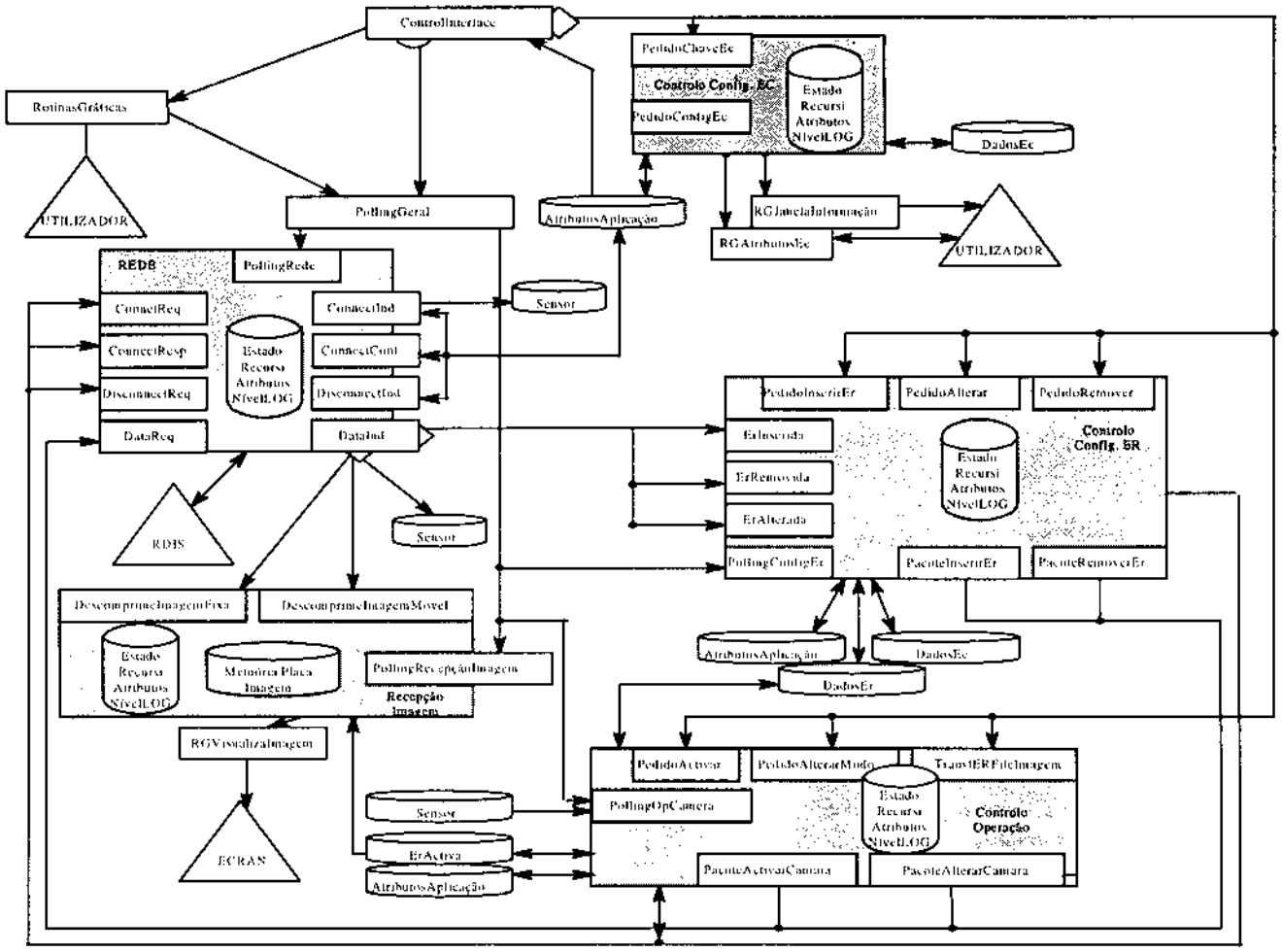


Fig. 7 -Diagrama de Estrutura inicial do Modelo de Implementação do Programa da EC.

Nesta figura surgem novos módulos, sub-módulos e armazéns adicionais que não foram definidos no Diagrama de Contexto: este facto deve-se à necessidade de construir um modelo síncrono necessário à implementação, com base no modelo assíncrono da análise.

Assim os diagramas das figuras 7 e 8 já reflectem os seguintes aspectos:

1 -A separação entre os três tipos base de rotinas, rotinas de *Interface*, rotinas de *Comunicação* e rotinas de *Processamento* (estas últimas relativas aos três módulos de controlo e ao módulo de recepção de imagem), para permitir um desenvolvimento em paralelo da aplicação por uma equipa de programadores, facilitando a integração e os testes finais, uma vez que é possível realizar testes individuais. A própria integração é gradual uma vez que cada módulo quando atinge alguma maturidade é passado aos outros programadores. Por outro a separação entre os três facilita a reutilização de módulos.

2- Para se obter uma melhor organização do software, cada tipo de rotinas deve ser implementada no seu directório respectivo. Cada um destes módulos, ao ser implementado, origina diversas rotinas de alto nível e um conjunto de rotinas auxiliares, eventualmente de processamento intermédio. As rotinas de mais alto nível que servem de interface com o exterior devem ser todas agrupadas num único ficheiro de interface, para que esse módulo possa ser usado de uma forma clara e simples pelos restantes, sendo as rotinas auxiliares agrupadas noutros ficheiros nesse mesmo directório. Assim cada módulo principal deve oferecer para o exterior um conjunto específico de rotinas (ou métodos se o módulo for implementados à custa de um objecto) onde cada rotina corresponda a um pedido a processar e onde os seus parâmetros sejam, de preferência, parâmetros elementares em vez de ponteiros para estruturas, dado que isso implicaria o conhecimento prévio dessas estruturas por parte da aplicação cliente. Este aspecto de organização está reflectido nas zonas a sombreado.

3- A relação entre as rotinas e as variáveis que estas manipulam, deve ser clara para quem examinar o software desenvolvido.

4- Como algumas acções levam tempo a serem concluídas e a execução do programa não deve ficar suspensa durante esse período (porque podem existir outros pedidos à espera de serem processados), é

necessário que cada módulo, implementado ou não à custa de máquinas de estado eventualmente só com um estado (fig. 8), possua um conjunto de variáveis (atributos) que lhe permita continuar a processar novos pedidos não deixando, no entanto, de possuir informação suficiente para processar os anteriores logo que possível.

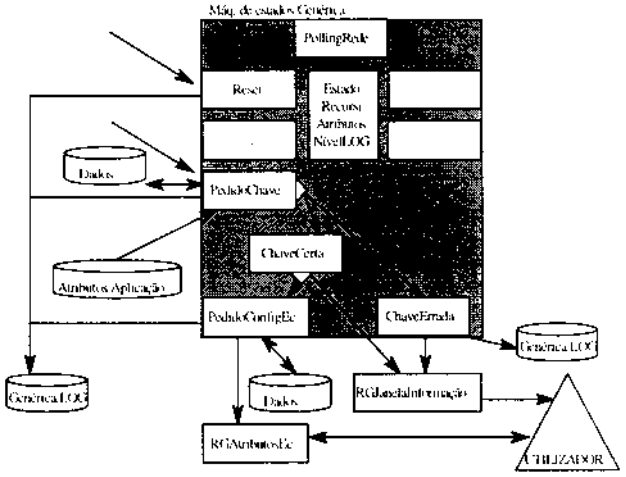


Fig. 8 - Exemplo da arquitectura de uma máquina de estados Genérica.

Para isso o módulo deve ainda possuir uma rotina (ex. *PollingRede*) que possa ser chamada periodicamente pela aplicação para que, quando existirem pedidos que ela não possa processar de uma forma total ou imediata por não existirem recursos disponíveis naquela altura, ou a máquina de estados não se encontrar no estado adequado para o fazer e necessitar de colocar esse pedido numa fila de espera, o pedido não deixe de ser processado. Nessa altura é necessário que o módulo possua um contador interno para que possa esperar algum tempo (< *TimeOut*) antes de tentar processar de novo aquele pedido ou simplesmente completar o seu processamento.

Cada rotina (uma rotina para cada pedido possível) deste módulo deve retornar um de três valores possíveis: o pedido não pode ser aceite, o pedido foi processado e o pedido irá ser processado. Neste último caso o pedido irá ser processado se a aplicação evocar periodicamente a rotina de *polling* da máquina de estados ou se a aplicação evocar outros procedimentos que venham a despoletar outra rotina desta máquina de estados que complete o seu processamento. A figura 8 mostra um exemplo de implementação de um módulo genérico.

5- O processamento de acontecimentos em sistemas de tempo real deve prevêr a chegada de novos acontecimentos para processar ainda antes do anterior ter sido completamente processado, nomeadamente quando existem pedidos cujo o processamento é demorado. No caso do Sistema de Televigilância, que é objecto deste trabalho, vários problemas se levantaram e a arquitectura do *software* é crítica. Os problemas que se podem colocar são de diferentes tipos, como por exemplo:

- Quando a meio do processamento de um pedido feito pelo utilizador chega da rede, um aviso de que um sensor foi accionado, ou chegam da rede imagens para descomprimir.
- Quando a EC está a descomprimir imagens vindas da rede pode chegar um aviso de que um sensor foi accionado; como este aviso não pode ser ignorado é necessário continuar a monitorar a rede e processar o alarme em paralelo com a descompressão da imagem.
- Como a rede tem de estar sempre a ser monitorada, mesmo durante a descompressão de uma imagem (fixa ou móvel) pode chegar uma nova imagem para descodificar ainda antes da anterior ter sido totalmente descodificada.

Duas hipóteses se põem na recepção dos acontecimentos: ou os dados recebidos são armazenados num *buffer* de recepção e processados posteriormente quando houver disponibilidade da aplicação, ou os dados são imediatamente processados, interrompendo-se momentaneamente o processamento anterior. Como vamos ver a seguir, ambas as hipóteses têm vantagens e desvantagens.

A primeira hipótese (processamento posterior) tem a grande desvantagem de implicar o uso de um *buffer* de recepção, com todos os atrasos, ocupação de memória e gestão de *buffers* inerentes. O tamanho do *buffer* de recepção dependerá do tempo máximo entre o processamento dos dados recebidos e da cadência de recepção de dados nesse período, podendo em casos extremos, ocorrer uma situação de *overflow*.

Apesar das desvantagens já referidas, esta hipótese tem a vantagem de evitar a recursividade indirecta que pode existir na segunda hipótese.

A segunda hipótese (processamento imediato) evita o uso do *buffer* de recepção e torna o processamento mais rápido. Nesta hipótese as rotinas de processamento não devem suspender a execução do programa enquanto esperam a conclusão do processamento mais demorado, mas devem retornar a execução ao programa chamador; neste caso deverão estar preparadas para que a aplicação as possa eventualmente reevocar, fazendo-lhe outro pedido ainda antes do anterior ter terminado (recursividade indirecta).

A recursividade indirecta pode originar situações anómalas em que o programa funciona durante algum tempo e sem razão aparente bloqueia ou provoca a reinicialização inesperada da máquina. Outra consequência pode ser a necessidade de compilar o programa com um *Stack* muito grande. Este tipo de problemas pode fazer o programador perder muito tempo a rever o código e a fazer alterações indevidas, até que o problema seja, finalmente, detectado. Este problema pode ser facilmente detectado usando uma variável estática que é inicializada a um no início da rotina que corra o risco de

ser reevocada e posta a zero no fim; como uma variável estática mantém o seu valor entre reevocações funciona como uma "flag".

Esta discussão permite concluir que a segunda hipótese pode ser preferível dependendo da complexidade da aplicação e das precauções tomadas pelo programador.

6 -Cada módulo deve possuir um ficheiro de LOG onde todos os pedidos recebidos e todas as respostas às suas acções fiquem gravadas para consulta posterior. Este ficheiro é particularmente útil durante o desenvolvimento da aplicação devido à grande quantidade de erros de execução, mas também é útil em aplicação real para que seja sempre possível detectar com exactidão erros que venham a ocorrer. A máquina de estados deve ainda permitir vários níveis de LOG como por exemplo: registo de todas os pedidos e respostas a acções, registo apenas de pedidos que não puderam ser satisfeitos ou respostas negativas a acções por si tomadas, ou ainda registo dos parâmetros que compõem cada pedido.

7- Cada módulo deve ainda ser capaz de permitir à aplicação fazer o seu próprio *reset* para que seja possível numa situação anómala ou mesmo numa situação normal voltar ao ponto de partida.

A figura 7 apresenta os aspectos mais importantes da arquitectura proposta para a implementação do sistema. Nesta figura pode-se notar a existência de cinco máquinas de estados, cada uma delas respeitando a arquitectura atrás referida: *Controlo Config. EC*, *Controlo Config. ER*, *Controlo Operação*, *Rede* e *Recepção de Imagem*.

Pode-se ainda observar a existência de várias *Rotinas Gráficas* responsáveis por criar janelas de interface com o *Utilizador* e utilizadas pelos módulos *Controlo do Interface* e pelas máquinas de estados, embora a figura não o mostre, excepto para a máquina de estados *Controlo Config. EC* (ex. *RGJanelaInformação*, *RGAtributos* e *RGVisualizaImagem*), para que estes possam receber pedidos ou possam enviar para o *Utilizador* as mensagens e imagens necessárias. As *Rotinas Gráficas* são também responsáveis por evocar o módulo de *PollingGeral* durante o período de tempo em que o *Utilizador* introduz dados, para evitar que o sistema fique sem monitorar a rede ou sem poder processar pedidos eventualmente já recebidos mas ainda não completamente processados durante esse período de tempo.

O módulo *PollingGeral* é evocado não só pelas rotinas gráficas mas também, ciclicamente, pelo módulo *Controlo do Interface* quando o sistema se encontra à espera de novos pedidos para processar. Como se pode ver na figura 7 o módulo *PollingGeral* evoca todas as rotinas de *polling* do sistema, nomeadamente a rotina de *PollingRede* que permite ao sistema receber pedidos da rede, nomeadamente o pedido para processar a ocorrência de um sensor numa ER; quando essa situação ocorre, a rotina *DataInd* ou a rotina *ConnectInd* é evocada e os atributos relativos ao sensor (ex. NomeEr, número do sensor accionado,...) são guardados no armazém *Sensor*.

Posteriormente a rotina *PollingOpCamara* é evocada pelo *PollingGeral* permitindo à máquina de estados *Controlo Operação* consultar os atributos do armazém *Sensor*, analisar o estado da aplicação (armazém *Atributos Aplicação*) e os atributos da ER presentemente activa (armazém *ErActiva*) para processar então o sensor activando (rotina *PacoteActivarCamara*) a recepção de imagens da câmara associada.

Quando a máquina de estados *Recepção Imagem* é evocada pela rotina *DataInd*, é-lhe passada uma imagem fixa ou uma imagem móvel para descodificar. Como a descodificação de uma imagem é demorada, as rotinas de recepção de imagem colocam a imagem a descodificar na memória da placa de imagem e retornam o controlo da execução do programa para os restantes módulos (em última análise para o *Controlo do Interface*) e ciclicamente o módulo *PollingGeral* evoca o *PollingRecepçãoImagem* que depois de verificar se a imagem já foi descodificada a enviará para o *Écran*.

Embora a figura não o explicite, os pedidos feitos ao módulo de rede implicam internamente o *polling* da rede para confirmar se o pedido foi aceite. Por exemplo, quando se envia um pacote para a rede (através da rotina *DataReq*) esse pacote é colocado num *buffer* de envio que só pode ser considerado disponível para acolher um novo pacote quando a confirmação de que o anterior foi enviado tiver sido recebida, implicando por isso a monitorização (*PollingRede*) da rede; no entanto a monitorização da rede permite receber novos dados ou outros acontecimentos (nomeadamente um pacote de dados para ser processado) antes da chegada da confirmação esperada. Nessa situação dois tipos de implementação são possíveis: ou se processa imediatamente os dados recebidos o que pode originar recursividade indirecta, ou são colocados num *buffer* de recepção com toda a gestão e problemas de *buffer overflow* que essa opção implica. Neste caso optou-se por processar imediatamente os dados recebidos (*DataInd*) tendo o cuidado de prever a recursividade indirecta que este tipo de implementação pode gerar. Por exemplo, se a rotina *PacoteActivarCamara* da máquina de estados *Controlo Operação* evocar a rotina *DataReq* e durante a sua execução fôr recebido um pacote com a informação sobre a ocorrência de um alarme numa ER, nessa altura a máquina de estados *Controlo Operação* volta a ser evocada para processar o sensor e as rotinas *PacoteActivarCamara* e *DataReq* poderiam voltar a ser evocadas segunda vez dado que o processamento do sensor implica enviar um pacote com a ordem para adquirir imagens da câmara associada ao sensor accionado. Para evitar essa recursividade indirecta a ocorrência de alarmes não é imediatamente processada mas a sua ocorrência é guardada no armazém *Sensor* e posteriormente processada por *PollingOpCamara*.

IV CONCLUSÕES

Da aplicação de técnicas estruturadas no desenvolvimento e especificação de um Sistema de Televigilância baseado na RDIS, cujas funcionalidades foram sumariamente descritas neste artigo, várias conclusões sobre as suas vantagens e desvantagens podem ser apresentadas: vamos, no entanto, centrar-nos nas principais.

Primeira conclusão, provavelmente óbvia mas que não é de mais salientar, é que a utilização destas técnicas é fundamental no desenvolvimento e documentação de software, dado que permite definir e documentar soluções para problemas específicos; no entanto, estas técnicas têm um custo imediato que é o tempo consumido na sua aplicação, mas que é rentabilizado por permitir uma especificação completa e rigorosa.

O maior ou menor nível de detalhe das especificações criadas na análise e projecto estruturado deve ser pesado com o tempo que demoram a efectuar, sendo necessário encontrar uma solução de compromisso caso a caso. Nesta perspectiva é conveniente usar, sempre que possível, para a especificação dos processos as *Pré-Post Conditions* dado que permitem, com rapidez e com um elevado grau de abstracção, especificar os processos. Além disso as *Pre-Post Conditions* dão ao programador a liberdade de implementar esses processos da forma mais adequada ao ambiente de implementação

Um aspecto essencial das especificações criadas consiste na definição rigorosa e com um bom nível de abstracção das principais estruturas de dados conseguida com os DER tendo-se concluído ser esta a melhor ferramenta proposta para este fim. O Dicionários de Dados é particularmente importante quando o sistema se destina a ser implementado por uma equipa de trabalho.

Apesar das vantagens referidas estas técnicas possuem algumas limitações no que diz respeito ao desenvolvimento de sistemas que tenham de processar acontecimentos em tempo real, mesmo no caso do método de análise proposto por Yourdon que se concluiu ser o que melhor se coaduna com a especificação de sistemas deste tipo. No exemplo apresentado este facto foi particularmente significativo na descrição das complexas interacções entre os processos de controlo do sistema e as várias entidades externas, que pelo facto de não estarem sincronizadas, podiam gerar novos pedidos a meio do processamento de pedidos anteriores: como esses acontecimentos não podem ser ignorados, nomeadamente no caso de serem imagens ou avisos de alarmes, podem-se originar situações de recursividade indirecta que não são facilmente detectáveis através da utilização desta técnicas. No entanto parece-nos que estas limitações da análise não são demasiado importantes, dado estes aspectos deverem ser considerados no projecto estruturado e não na análise.

No que diz respeito ao Diagrama de Estrutura criado no projecto estruturado, pareceu-nos bastante útil, fácil de criar e de manter. Estes diagramas são bastante úteis na implementação do sistema dado permitirem definir um modelo síncrono e possibilitarem, por isso, considerar com facilidade diversos aspectos, tais como: procedimentos de recuperação de erros, tratamento de excepções, problemas relacionados com limitações relativas ao ambiente de implementação bem como o problema da recursividade indirecta já referido. No entanto o projecto estruturado contém algumas limitações, nomeadamente, não induzir o programador na implementação a organizar convenientemente o código e os dados.

Tendo por base o objectivo de desenvolver o software de uma forma estruturada por forma a facilitar o seu desenvolvimento, reutilização, manutenção e compreensão, foram usadas com sucesso as seguintes estratégias:

1-Organizar as diferentes rotinas em três tipos principais: rotinas de *Interface*, rotinas de *Processamento* e rotinas de *Comunicação*.

2- Todas as rotinas de um dado tipo devem estar organizadas num só directório específico e por sua vez em cada directório as várias rotinas devem estar organizadas em vários ficheiros existindo um só ficheiro com as rotinas, de mais alto nível, que servem de interface para o exterior.

3- A relação entre as rotinas e os dados que estas manipulam, deve ser clara para quem examinar o software desenvolvido, nomeadamente os dados (variáveis) internos a cada módulo como se pode ver na figura 7.

4- Cada módulo principal deve estar organizado de forma a poder interromper temporariamente um dado processamento mais demorado, preservando toda a informação necessária ao seu recomeço. O recomeço desse ou de outros processamentos temporariamente interrompidos relativos a esse módulo deve poder ser despoletado por uma rotina de *polling* (desse módulo) que a aplicação possa evocar, periodicamente ou quando ocorrer um acontecimento especifico.

5- O processamento de acontecimentos em sistemas de tempo real deve prevêr a chegada de novo(s) acontecimento(s) para processar ainda antes de pedidos anteriores terem sido completamente processados, nomeadamente quando estes últimos têm de aguardar por confirmações vindas de entidades externas para poderem ser terminados.

6 -Cada módulo deve possuir um ficheiro de LOG onde todos os pedidos recebidos e todas as respostas às suas acções fiquem gravadas para consulta posterior.

7- Cada módulo deve ainda ser capaz de permitir à aplicação fazer o seu próprio *reset* para que seja possível numa situação anómala, ou mesmo numa situação normal, voltar ao ponto de partida.

REFERÊNCIAS

[1] J. Santos, F. Ramos, O. Santos, "Sistema de Televigilância suportado na RDIS", Revista do DETUA, Vol. 1, Nº 2, Setembro de 1994, pp 169-180.

[2] T. DeMarco, Structured Analysis and System Specification, Prentice-Hall, Englewood Cliffs, N.J., U.S.A., 1979.

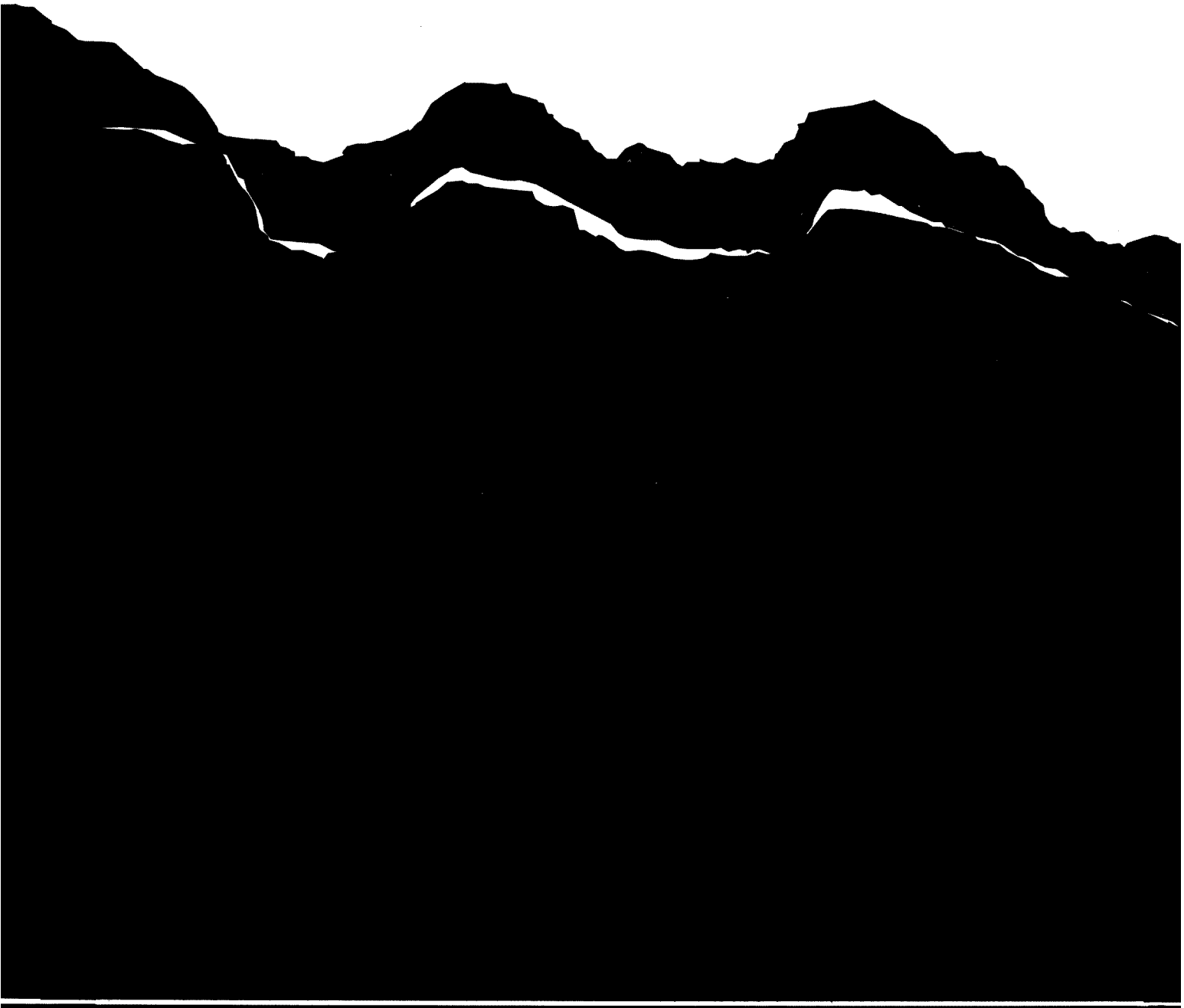
[3] C. Gane, T. Sarson, "Structured System Analysis: Tools and Techniques", Prentice-Hall, Englewood Cliffs, N.J., U.S.A., 1979.

[4] E. Yourdon, "Modern Structured Analysis", Prentice-Hall, Englewood Cliffs, N.J., U.S.A., 1989.

[5] S. McMenamim, J. Palmer, "Essential System Analysis", Yourdon Press, 1984.

[6] P. Ward, S. Meller, "Structured Developemnt for Real-Time Systems", vol. I, II and III, Yourdon Press, 1985.

[7] J. Martin, C. MacClure, "Structured Techniques: Basis for CASE", Prentice-Hall, Englewood Cliffs, N.J., U.S.A., 1988.



ARTES GRÁFICAS
SÉRGIO CABAÇO