MAURO CAVALCANTE PEQUENO

Tese apresentada ao Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do grau de Doutor em Ciências.

JOÃO MARQUES DE CARVALHO
Orientador

MÁRIO TOYOTARO HATTORI
Co-orientador

CAMPINA GRANDE - PB
DEZEMBRO - 1991

P425s Pequeno, Mauro Cavalcante.
SIPREX : um sistema especialista para processamento digital de sinais / Mauro Cavalcante Pequeno. - Campina Grande, 1991.
112 f.

Tese (Doutorado em Engenharia Elétrica) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1991.
"Orientação : Prof. Dr. João Marques de Carvalho, Prof. M.Sc. Mário Toyotaro Hattori".
Referências.

1. Processamento Digital de Sinais. 2. Processamento Digital de Imagens. 3. SIPREX. 4. Tese - Engenharia Elétrica. I. Carvalho, João Marques de. II. Hattori, Mário Toyotaro. III. Universidade Federal da Paraíba - Campina Grande (PB). IV. Título

CDU 004.932(043)

SIPREX: UM SISTEMA ESPECIALISTA PARA PROCESSAMENTO DIGITAL DE SINAIS

MAURO CAVALCANTE PEQUENO

TESE APROVADA EM 17.12.91

JOAO MARQUES DE CARVALHO, Ph.D., UFPB
Orientador

MARIO TOYOTARO HATTORI, M.Sc., UFPB
Co-orientador

HELIO WALDMAN, Ph.D., UNICAMP
Componente da Banca

LUIS ALBERTO VIEIRA DIAS, Ph.D., INPE
Componente da Banca

RUDOLPH EUGENE SEVIORA, Ph.D., UNIV. WATERLOO
Componente da Banca

BENEDITO GUIMARAES AGUIAR NETO, Dr.-Ing., UFPB
Componente da Banca

CAMPINA GRANDE - PB
DEZEMBRO - 1991

Ao meu filho Rannier

AGRADECIMENTOS

Em primeiro lugar desejo agradecer aos professores João Marques de Carvalho e Mário Toyotaro Hattori, meu orientador e co-orientador, respectivamente, pelas sugestões, pelas críticas, pela visão, pelo apoio e pelo crédito dado a mim e ao meu trabalho desde que se iniciou, no período em que os caminhos não estavam ainda bem definidos. Suas participações foram fundamentais à consecução desse trabalho.

Desejo agradecer aos participantes de nossa equipe de trabalho, Antônio Mendes da Silva Filho, Adriano Fábio Q. de Brito e Maria Lígia Perkusich que muito contribuíram para a implementação do sistema SIPREX.

Desejo agradecer a todos os professores e funcionários dos Departamentos de Engenharia Elétrica e Sistemas e Computação da UFPb pelo tratamento cordial que sempre me dispensaram. Em especial desejo agradecer ao professor Pedro Sérgio Nicolletti e a Nicolaas Ruberg pelo apoio sempre prestado.

Desejo agradecer à Universidade Federal do Ceará por investir em minha titulação, e mais particularmente aos professores do Departamento de Computação que arcaram com minha carga horária durante o período que estive afastado.

Desejo agradecer a todos os meus colegas de pós-graduação da UFPb, Campus II, pelo companheirismo e pelo excelente relacionamento.

Finalmente, desejo agradecer o apoio financeiro, indispensável à execução de um trabalho deste porte, concedido pela CAPES, CNPq e IBM do Brasil.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

SIPREX - UM SISTEMA ESPECIALISTA PARA PROCESSAMENTO DIGITAL DE SINAIS

RESUMO

SIPREX é um sistema especialista que acopla computação numérica e simbólica desenvolvido para ser usado como uma ferramenta inteligente por engenheiros de processamento digital de sinais (PDS). O sistema trabalha como um especialista em diferentes técnicas para análise e síntese de filtros digitais, decidindo na escolha dos métodos e parâmetros relevantes para uma aplicação particular. SIPREX também é capaz de direcionar o usuário, ou mesmo decidir por ele, na escolha das especificações para o problema proposto, baseado nas informações sobre áreas de aplicações de PDS contidas na base de conhecimentos. O resultado final é o projeto de um filtro que representa a melhor solução para o problema proposto.

SIPREX - AN EXPERT SYSTEM FOR DIGITAL SIGNAL PROCESSING

ABSTRACT

SIPREX is an expert system which couples symbolic and numeric computation developed to be used as an intelligent tool by digital signal processing (DSP) engineers. The system works as an expert on different analysis and synthesis techniques of digital filters, deciding on the choices of methods and trade-offs among relevant design parameters. SIPREX is also capable of directing the user, or even deciding for him, choosing design specifications for the problem at hand, based on information about application areas of DSP available in its knowledge base. The final result is a filter design which represents the best solution for the problem at hand.

CAPÍTULO 1

INTRODUÇÃO

1.1 Introdução

O conceito de sistemas especialistas (SE) surgiu na década de 70, quando pesquisadores da área de inteligência artificial (IA) decidiram abandonar a busca por máquinas com inteligência generalizada e concentrar esforços na solução de problemas práticos. Portanto, os SEs são um dos primeiros exemplos de aplicação de IA e os últimos anos têm presenciado uma verdadeira explosão de sua aplicação. SEs têm sido empregados em funções distintas, entre elas, interpretação de dados, previsão baseada em situações passadas, ferramenta de projeto, diagnósticos, controle, detecção e correção de falhas, instrução [1]. Particularmente, em aplicações de engenharia, tem se tornado cada vez mais comum o surgimento de sistemas que têm seus processos de decisão apoiados por uma base de conhecimentos, em termos da inteligência artificial.

Tipicamente um SE é baseado em um extenso cabedal de conhecimentos sobre uma área específica. Em geral, esses conhecimentos são organizados na forma de uma coleção de regras, que permitem ao sistema tirar conclusões inteligentes a partir de dados ou premissas, sendo capaz de oferecer orientação, ou tomar decisões inteligentes [2].

De particular interesse para os usuários de matemática computacional é a classe dos sistemas especialistas acoplados, os quais integram uma estrutura que utiliza computação simbólica (base de conhecimentos e motor de inferência) com algoritmos de computação numérica [3]. Um dos motivos para a crescente popularidade desse tipo de SEs deve-se ao fato de que a comunidade científica que utiliza computação numérica necessita fundamentalmente na resolução de seus problemas, de discernimento (*insight*) e precisão dos resultados. O discernimento é necessário para se obter a solução do problema e para se interpretar os resultados computados. Por outro lado, o usuário necessita muitas vezes de orientação na escolha e utilização das ferramentas disponíveis. A matemática computacional tem se preocupado com a complexidade e precisão dos algoritmos, mas não tem tido a mesma preocupação com a compreensão dos mesmos em aspectos como a aplicabilidade e a abrangência. Aos usuários tem cabido a escolha do método e ferramentas a serem utilizados na resolução de seus problemas, bem como a interpretação dos resultados obtidos. Por isso na maioria das vezes tal escolha e interpretação é falha já que, devido a diversidade de algoritmos, essa tarefa exige do usuário um grau de especialização, nem sempre presente, a fim de produzir um resultado ótimo.

Um exemplo típico da situação acima descrita é o processo de especificação e projeto de um filtro digital, onde cada etapa envolvida requer do engenheiro uma série de tomadas de decisão, como determinação das faixas de passagem e corte, tipo de filtro a ser utilizado (FIR ou IIR), características de fase, atraso de grupo, estrutura a ser usada na implementação e tamanho de palavra (precisão), dentre outras. Essas tomadas de decisão exigem do projetista conhecimento detalhado dos tipos de filtros e suas características, como também dos vários métodos existentes para síntese dos mesmos [4]. Na prática, contudo, na maioria das vezes este conhecimento não existe, limitando-se o projetista a utilizar técnicas com as quais está familiarizado, ou para as quais dispõe de pacotes de software (sub-rotinas) de fácil uso. Essa situação é compreensível, se levarmos em conta que as técnicas de projeto de filtros digitais disponíveis, principalmente no caso de filtros ótimos, utilizam ferramentas de cálculo numérico, teoria de otimização e programação linear com as quais o engenheiro comum, em geral, está familiarizado de uma maneira apenas superficial. O resultado é um filtro cujo desempenho embora aceitável, pode estar abaixo do melhor possível para aquela aplicação. Daí a vantagem de se ter um sistema capaz de armazenar o conhecimento de especialistas na área e adequá-lo a uma determinada situação. Isso é justamente a definição de sistema especialista.

Portanto, um sistema capaz de integrar o conhecimento de especialistas em processamento digital de sinais (PDS) e suas aplicações, armazenado de maneira simbólica, com complexos algoritmos numéricos constituir-se-á numa ferramenta potencialmente útil na resolução de problemas de projeto e análise de filtros digitais.

1.2 Sistemas Especialistas Acoplados

Existem problemas que não podem ser solucionados pela matemática computacional tradicional, mas, com uma abordagem inteligente e emprego de algoritmos de maneira adequada, consegue-se fazê-lo. É, por exemplo, o caso do SE de Talukdar [5], um sistema especialista para resolver sistemas de equações algébricas não-lineares, composto de três diferentes algoritmos que são gerenciados na resolução de um problema. O SE de Talukdar é capaz de solucionar sistemas de equações que não poderiam ser solucionados por nenhum dos seus algoritmos isoladamente.

Alguns sistemas que acoplam computacão simbólica e numérica foram desenvolvidos há algum tempo, como o Mycin, Prospector, Dendral, Hearsay-II e alguns sistemas para processamento de imagens, como o sistema de Sueda [6]. Esses sistemas utilizam métodos numéricos sem nenhum conhecimento prévio sobre os algoritmos e sem se preocupar com a interpretação dos resultados obtidos. Sistemas desse tipo não são considerados sistemas acoplados (*coupled systems*), pelo menos no sentido definido pelo *Workshop on Coupling Symbolic and Numerical Computing in Expert Systems*, realizado em Washington em agosto de 1985. Durante o evento

estabeleceu-se a definição, a motivação, a arquitetura e os fatores mais importantes no projeto de sistemas acoplados [3]. Dessa forma, um sistema para ser considerado um sistema acoplado é necessário, além de ser composto de uma base de conhecimentos (computação simbólica) e de algoritmos numéricos, estar munido do conhecimento dos processos numéricos envolvidos e as razões sobre as aplicações e/ou uma interpretação dos resultados obtidos nos processos numéricos. A figura 1.1, sugerida por Kitzmiller & Kowalik [7], dá uma idéia precisa de sistemas acoplados.

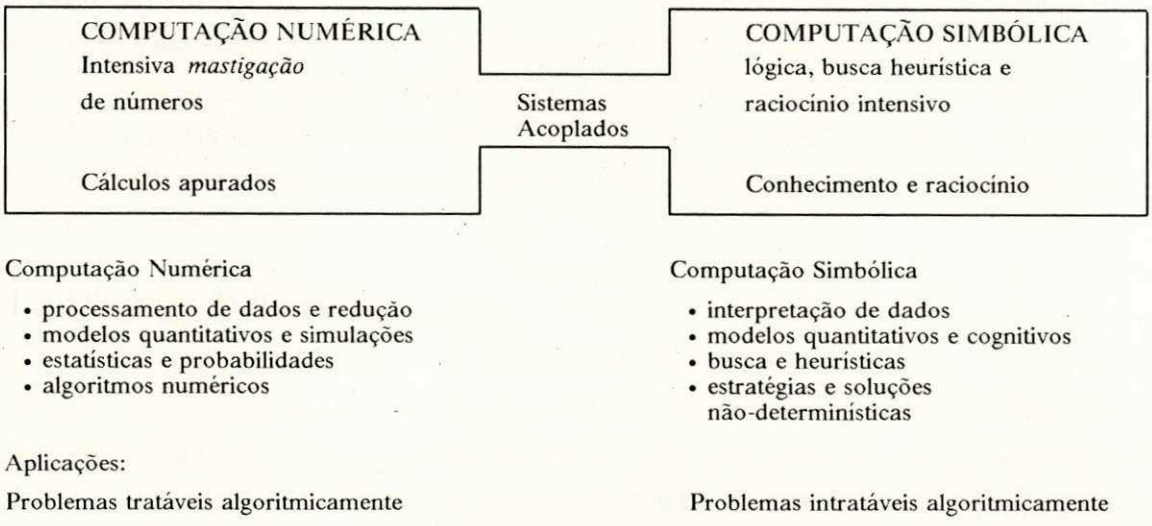

Figura 1.1: Sistemas Acoplados

Quanto à natureza, os sistemas acoplados podem ser classificados como sistemas de acoplamento superficial (*shallow coupled*) e sistemas de acoplamento profundo (*deep coupled*) [7]. Os sistemas de acoplamento superficial tratam os algoritmos como *caixas pretas*, isto é, têm pouco conhecimento sobre os processos envolvidos. Em geral, eles selecionam e orientam o uso de algoritmos numéricos e interpretam os resultados das rotinas numéricas. Um exemplo desse tipo de sistema é o EXBITAN (ver capítulo 2), um sistema capaz de solucionar problemas de obtenção de zeros de funções algébricas e transcendentais, de sistemas não-lineares, de otimização linear e não-linear, e de integração numérica. O Exbitan orienta o usuário na construção de funções (quando o problema for de obtenção de zeros de funções e de integração numérica), e na passagem dos parâmetros necessários aos algoritmos. Além disso, seleciona os algoritmos tentando manter o erro dentro de uma margem exigida pelo usuário.

Os sistemas de acoplamento profundo utilizam um conhecimento extensivo de cada processo juntamente com outras informações oriundas da experiência dos especialistas, simulação de situações, informações fornecidas pelo usuário, e, em algumas aplicações, de um aprendizado automático realizado pelo próprio sistema. O sistema especialista utiliza essa base de conhecimentos na resolução dos problemas, ativando rotinas numéricas de uma forma adequada, interpretando resultados intermediários e efetuando

um refinamento da solução pelo uso inteligente de seqüências do processo. Um exemplo de um sistema de acoplamento profundo é o sistema desenvolvido pelo North American Rockwell [8], o *Engagement Analyst's Apprendice* (EAA) que auxilia o usuário a construir simulações a partir de módulos de simulação e a avaliar os resultados obtidos.

O conhecimento das limitações e requisitos de cada processo possibilita aos sistemas de acoplamento profundo maior robustez e melhor desempenho na resolução de problemas, bem como uma seleção mais inteligente dos processos a aplicar, do que os de acoplamento superficial. Por sua vez os sistemas de acoplamento superficial são, em geral, mais simples de se usar e menos dispendiosos na sua elaboração.

A maioria dos sistemas que utilizam técnicas de acoplamento entre computação simbólica e numérica desenvolvidos até então são do tipo superficial inclusive os relacionados com processamento de sinais, onde os programas e rotinas são acoplados e usados separadamente.

1.3 Objetivo do Trabalho

O objetivo deste trabalho é a realização de estudos que possibilitem a construção de um sistema especialista acoplado (SIPREX) que possa ser utilizado como uma ferramenta de CAD por engenheiros projetistas de sistemas digitais para processamento de sinais e controle automático de processos. O SIPREX atuará como um especialista nas diversas técnicas existentes para análise e síntese de sinais e sistemas digitais, orientando o engenheiro na escolha de métodos de projeto e no estabelecimento de compromissos entre os diversos fatores que influenciam o desempenho de um sistema para processamento digital de sinais. O SIPREX deverá comunicar-se com o engenheiro usuário de uma forma interativa, recebendo deste, especificações sobre o problema a ser resolvido. A partir dessas especificações, o SIPREX terá condições de fornecer ao usuário um projeto final otimizado, obtido por um dos métodos ótimos existentes.

A não existência de metodologias formais para construção de sistemas acoplados tem sido responsável pela construção de sistemas que apresentam um desempenho aquém de suas potencialidades. Algumas dificuldades de se determinar uma metodologia para criação de sistemas acoplados foram apontadas por Jacobstein at al. [3], entre elas:

a) Duas disciplinas, inteligência artificial e matemática computacional, devem ser integradas para se ter um sistema acoplado, e cada uma delas tem seu próprio vocabulário especializado e seus paradigmas de resolução de problema.
b) Só recentemente a comunidade científica começou a se preocupar em estabelecer uma metodologia formal para desenvolvimento de sistemas acoplados.
c) Os sistemas acoplados não têm sido instrumentados para analisar os benefícios que as técnicas de IA trazem aos métodos numéricos (ou vice-versa).
d) Na maioria dos casos, os projetistas dos sistemas e aqueles responsáveis por seu desenvolvimento, são também seus usuários finais.

Como uma perspectiva futura, a comunidade científica espera o surgimento de uma metodologia formal de acoplamento, o desenvolvimento de técnicas adequadas de engenharia de software, novos formalismos para representar abstrações, extensões da lógica de 1ª ordem e incremento de sofisticações matemáticas no campo de IA. Tudo isso levará a exploração das verdadeiras potencialidades dos sistemas acoplados.

Uma tentativa recente de se estabelecer uma metodologia de um sistema acoplado empregado na automação e tratamento de sinais pode ser encontrada em Favier at al. [9].

O SIPREX deverá trazer contribuições a esses estudos, pois, além do papel de ferramenta de CAD, o SIPREX deverá contribuir para o desenvolvimento de sistemas acoplados, ou mais especificamente, de sistemas aplicados à engenharia em ambientes inteligentes. Para tanto, O SIPREX incorpora na sua estrutura diversos módulos que além de suas funções no sistema, servirão como ponto de investigação de diversos problemas:

a) O uso de interface inteligente modificará a linguagem de especificação dos problemas, minorando o esforço do usuário e, sobretudo, diminuindo as possibilidades de erros e inconsistência dos dados.
b) O uso de técnicas de IA poderá contribuir para melhorar o desempenho e aumentar o domínio dos sistemas, desde que estas técnicas sejam utilizados não apenas no sentido de classificação dos algoritmos e análise dos resultados, mas que tenham também a função de monitoramento e se incorporem à própria dinâmica dos algoritmos.
c) O desenvolvimento de técnicas de aprendizado automático contribuirá para o refinamento do sistema, sobretudo nas heurísticas necessariamente utilizadas, desde que boa parte das informações concernentes aos algoritmos de cálculos são provenientes de resultados de simulações, devido à freqüente inexistência de garantias formais para o pleno comportamento dos algoritmos face a fatores críticos, tais como a convergência.

Para se chegar ao sistema SIPREX, muitos estudos e experimentos tiveram que ser feitos. Tais esperimentos levaram à construção de dois protótipos (EXBITAN e FFTEX) com graus de complexidade e eficiência ascendentes. Estes sistemas serão descritos com detalhes nos capítulos 2 e 3, respectivamente.

Em suma, é esperado que o SIPREX represente uma solução inicial na abordagem de uma série de problemas envolvendo o uso de sistemas acoplados inteligentes para aplicações, inicialmente, na área de processamento digital de sinais e, posteriormente, na área de CAE (Computer Aided Engeneering) em geral.

1.4 Organização da Tese

Este trabalho foi organizado levando-se em conta a evolução da pesquisa desenvolvida e a natureza dos módulos que compõem o sistema SIPREX. Assim os capítulos foram dispostos da seguinte maneira:

- O capítulo 2 descreve o primeiro trabalho desenvolvido dentro desta linha de pesquisa, o EXBITAN, um protótipo de um sistema especialista acoplado, capaz de solucionar problemas de obtenção de zeros de funções algébricas e transcendentais, sistemas não-lineares, de otimização linear e não-linear, e de integração numérica.
- O capítulo 3 descreve o FFTEX, um sistema especialista para o cálculo eficiente da transformada rápida de Fourier. O FFTEX constitui-se numa evolução, em termos de concepção, em relação ao EXBITAN, pois além da escolha e ativação dos algoritmos de cálculo, incorpora uma maneira original de aprendizado automático.
- O capítulo 4 expõe a motivação para o desenvolvimento do sistema SIPREX, o assunto central deste trabalho, bem como sua arquitetura. Os módulos que compõem o sistema são descritos nos capítulos 5, 6 e 7.
- O capítulo 5 descreve as unidades responsáveis pelas tarefas de interligação no sistema, ou seja, os módulos de controle e interface.
- O capítulo 6 descreve os módulos central, de explanação e de aprendizado automático. Esses módulos são responsáveis pela computação simbólica do SIPREX.
- O capítulo 7 descreve os módulos que são responsáveis pela computação numérica do SIPREX. É constituído do módulo numérico, que implementa os algoritmos de projeto de filtros digitais, e do módulo gráfico, que expõe ao usuário a resposta em freqüência do filtro projetado.
- O capítulo 8 descreve diversos testes que foram realizados para a validação e avaliação do desempenho do sistema.
- O capítulo 9, o capítulo das conclusões, resume as principais contribuições deste trabalho à ciência, a análise comparativa com trabalhos similares realizados, e apresenta propostas para futuros trabalhos que darão prosseguimento à pesquisa realizada.

CAPÍTULO 2

EXBITAN - UM SISTEMA ESPECIALISTA PARA O CÁLCULO NUMÉRICO

2.1 Introdução

EXBITAN é um sistema especialista capaz de solucionar problemas de obtenção de zeros de funções algébricas e transcendentais, sistemas não lineares, de otimização linear e não linear, e de integração numérica. É composto de duas partes distintas acopladas entre si, uma parte simbólica que armazena as informações sobre os processos envolvidos (base de conhecimentos), e uma parte numérica, onde estão armazenados os algoritmos de cálculo numérico.

2.2 A Estrutura do EXBITAN

O EXBITAN visa auxiliar o usuário na solução de problemas numéricos, escolhendo um método adequado à resolução do problema proposto, orientando o usuário na especificação dos parâmetros necessários, para em seguida ativar a rotina que implementa o método em questão. Os resultados obtidos sofrem alguma interpretação, sobretudo no que diz respeito a ocorrências de situações patológicas.

Quando um determinado método falha na obtenção da solução de um problema, o EXBITAN, sempre que possível, ativa um outro algoritmo, até que se esgotem todas as possibilidades, e uma mensagem a esse respeito é exibida.

É composto, como já foi mencionado, de duas partes de natureza distinta, acopladas entre si, uma responsável pela computação simbólica e outra pela computação numérica.

A parte simbólica utiliza técnicas comuns à área de sistemas especialistas, ou seja, representações simbólicas, inferências simbólicas e buscas heurísticas. Consiste de um módulo de controle (*scheduler*), um módulo central, utilizando uma arquitetura em quadro negro (*blackboard*), composto de uma unidade de memória de trabalho, uma base de conhecimentos, e um motor de inferência.

O módulo de controle é o responsável pelo controle e ativação dos demais módulos, executa a ligação com a parte numérica, e ainda é o responsável pela comunicação amigável com o usuário e pela entrada e saída de dados.

A memória de trabalho é uma base de dados que contém informações específicas ao problema em resolução, os dados de entrada e os resultados intermediários obtidos de operações anteriores pelo motor de inferência.

A base de conhecimentos armazena informações genéricas sobre os problemas do domínio do sistema.

O motor de inferência executa a seleção e o casamento entre as informações contidas na memória de trabalho e na base de conhecimentos, gerando o chamado conjunto de conflito. Executa tarefas especificadas pela resolução de conflito, cuja ação pode modificar o conteúdo da memória de trabalho, enviar informações ou solicitar a ativação de rotinas numéricas.

2.3 O Sistema EXBITAN

O EXBITAN é um sistema especialista acoplado com a função de monitorar os algoritmos numéricos pertencentes à Biblioteca BITAN [10]. Assim, o EXBITAN visa auxiliar o usuário na escolha do melhor algoritmo para cada problema particular.

Possui métodos que permitem aos usuários calcular numericamente:

- Zeros de Polinômios Reais/Complexos
- Zeros de Funções Transcendentais
- Solução de Sistemas Não-Lineares
- O mínimo de uma Função Linear
- O Mínimo de uma Função Não-Linear sem Restrições sendo fornecido o gradiente da função objetivo ou não
- Integral Simples de Funções descritas por pontos ou analiticamente
- Integral Múltipla.

Pode-se descrever o EXBITAN como um sistema de controle ligado a diversos algoritmos numéricos como mostra o esquema da figura 2.1. Observa-se facilmente que o EXBITAN foi concebido de uma maneira tal que se pode suprimir, modificar e inserir novos métodos sem comprometer a estrutura do mesmo.

2.4 O Processo de Obtenção da solução

Inicialmente o EXBITAN exibe um menu contendo as diversas classes de problemas capaz de resolver, para logo a seguir perguntar ao usuário a que classe pertence o problema que deseja resolver.

Escolhida uma das opções o motor de inferência do sistema seleciona entre as regras que compõem a base de conhecimentos, as pertinentes ao processo escolhido, concluindo com a seleção de um determinado método. O módulo de controle então, parte para a ativação do módulo de interação com o usuário através do qual este deve fornecer os dados necessários à resolução do problema, sendo em seguida, ativada a rotina adequada para a solução numérica.

Os resultados obtidos são armazenados num arquivo (SAIDA) que por sua vez é solicitado pelo módulo de controle do sistema. Se o problema for resolvido com sucesso,

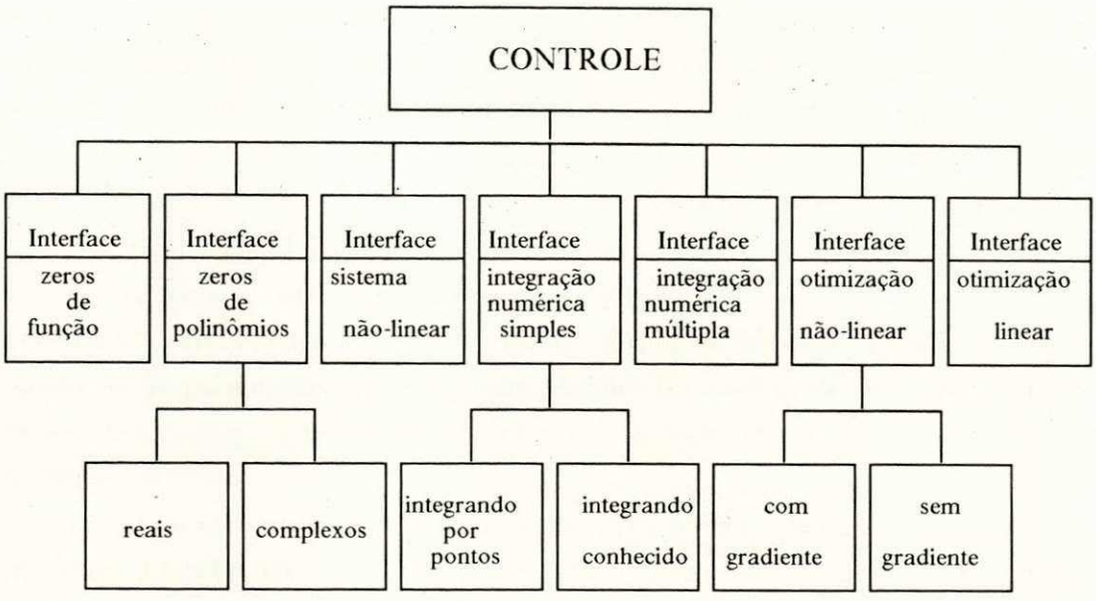

Figura 2.1: A Estrutura do EXBITAN

o resultado será fornecido ao usuário e o EXBITAN oferece ao usuário a opção por um novo problema ou parar o processamento. Se não houver sucesso em obter resultados, e se o método tiver mais de uma opção algorítmica, como no caso de polinômios, o EXBITAN tenta um outro algoritmo comprovadamente mais robusto, mas que não é ativado de início por ser menos eficiente em termos de tempo/espaço que o primeiro; o primeiro embora não tenha a mesma abrangência do segundo, é capaz de resolver a maioria dos problemas.

Se, no caso de insucesso, o sistema não tiver mais opções, uma mensagem de impossibilidade de resolução daquele problema será exibida.

Nem sempre um método pode ser ativado diretamente. Alguns métodos como os de integração numérica, resolução de sistemas não-lineares, zeros de função e os de otimização, requerem que o usuário forneça em primeiro lugar, a função utilizada no algoritmo de resolução. Neste caso, o EXBITAN instrui o usuário de como proceder para implementar esta função. Feita a implementação, de acordo com as instruções, o Sistema, após ser novamente carregado, ativa o método escolhido.

2.5 Implementação

As informações contidas na base de conhecimentos, são aquelas relativas ao domínio e ao comportamento dos algoritmos. Estão armazenadas sob a forma de regras de produção, sendo agrupadas de uma maneira tal que se pode facilmente identificar as pertinentes a uma classe específica de problemas. Assim: as regras 1, 1A, 1B,..., referem-se a problemas de cálculo de zeros de polinômios; as regras 2, 2A, 2B,..., referem-se a zero de funções transcendentais; as regras 3, 3A, 3B,..., referem-se a

integração numérica simples; as regras 4, 4A, 4B,..., referem-se a solução de sistemas não-lineares; as regras 5, 5A, 5B,..., referem-se a otimização não-linear sem restrições; as regras 6, 6A, 6B,..., referem-se a integração numérica múltipla; as regras 7, 7A, 7B,..., referem-se a otimização linear. As regras que são utilizadas por mais de um método estão agrupadas no final. São as regras Z1, Z2 e Z3. Esta maneira de agrupar as regras facilita a atualização do sistema.

Os programas que implementam os métodos de resolução estão escritos em FORTRAN, compilados e armazenados na forma de arquivos objetos (nome.OBJ), exceto os de polinômios ede integração simples de funções descritas por pontos que estão armazenados na forma executável pois não requerem do usuário o fornecimento de nenhuma rotina.

Os dados necessários à utilização de um método são solicitados ao usuário por um programa FORTRAN que é específico para um determinado método. Fornecidos os dados, as rotinas são ativadas, armazenando os resultados obtidos num arquivo chamado SAIDA. O EXBITAN executa então uma análise desses resultados e, de acordo com esta análise, a solução é transmitida ao usuário ou ativa-se um novo método, se possível, caso a solução não seja julgada satisfatória.

2.6 Conclusão

O objetivo deste trabalho foi mostrar a possibilidade e as vantagens de se construir e utilizar um sistema especialista numérico. A possibilidade ficou comprovada com a concretização do EXBITAN, sobretudo nos tópicos referentes a zeros de polinômios e otimização não-linear em que os atributos de um sistema especialista são mais sentidos.

Quanto às vantagens, deve-se levar em consideração que a grande clientela de rotinas numéricas básicas são pesquisadores que não dominam necessariamente a área, como engenheiros, físicos, estatísticos, etc., e que por melhor que sejam as documentações das bibliotecas numéricas existentes, esses especialistas relutam em utilizá-las e quando o fazem nem sempre estão aptos a fazer uma escolha dos métodos adequados. Então, o projeto de um sistema especialista com aplicação ao cálculo numérico básico como o EXBITAN, é válido e acredita-se que venha a se tornar uma ferramenta útil aos profissionais das ciências e engenharia.

A maneira como o projeto foi concebido deixou clara a possibilidade de ampliação do mesmo. Assim, é possível a incorporação de outros algoritmos referentes aos tópicos existentes para tornar o Sistema mais poderoso, útil e confiável. É possível também a inclusão de outros tópicos como Interpolação, Ajustamento de Curvas, Equações Diferenciais, etc., que contribuirão para o enriquecimento do Sistema.

CAPÍTULO 3

FFTEX - UM SISTEMA ESPECIALISTA PARA O CÁLCULO DE TRANSFORMADAS RÁPIDAS DE FOURIER

3.1 Introdução

A transformada de Fourier e sua inversa têm emprego nos mais diversos ramos da matemática computacional, constituindo-se numa ferramenta conveniente aos matemáticos, físicos, geólogos, meteorologistas, etc. São instrumentos matemáticos dos mais usados na análise de circuitos eletrônicos e especialmente em processamento de sinais. Seu emprego na análise de sinais e sistemas digitais deve-se ao fato de facilitar a análise no domínio da freqüência, de vários fenômenos de compreensão difícil no domínio do tempo.

O FFTEX permite o cálculo da transformada de Fourier para sinais discretos no tempo. Um sinal discreto no tempo $x(n)$ é uma seqüência que faz corresponder a cada inteiro, um único número real ou complexo. Se os valores da seqüência formarem um conjunto enumerável de números reais ou complexos, o sinal é dito ser *digital.*

A transformada discreta de Fourier $X(e^{j\omega})$ da seqüência $x(n)$ é definida como

$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x(n)e^{-j\omega n}. \qquad (1)$$

Um método que implementa diretamente a equação (1) - DFT, será computacionalmente dispendioso pois o cálculo de $X(e^{j\omega})$ pela equação (1) requer N^2 multiplicações complexas e $2N^2$ multiplicações reais, além da possibilidade, devido ao grande número de operações, de propagação de erros de arredondamento e truncamento no cálculo da exponencial complexa.

Em 1965 Cooley & Tukey [11], tirando proveito das propriedades de periodicidade da função exponencial complexa, conceberam o algoritmo de transformação rápida de Fourier (*Fast Fourier Transform - FFT*), que reduziu a complexidade do cálculo da transformada a $N\log N$, em que o logaritmo é na base 2. Para se ter uma idéia da eficiência do algoritmo de FFT, para $N = 1.024$ a DFT necessita de aproximadamente 3.145.728 operações, enquanto a FFT necessita apenas 10.240. Para efeito de comparação entre um algoritmo de DFT e outro de FFT, em termos de tempo de processamento e precisão dos resultados obtidos, calculou-se a transformada de uma seqüência $x(n)$ de $N = 1.024$ pontos gerada aleatoriamente, utilizando-se um PC/AT

equipado com co-processador aritmético, pelos dois algoritmos. Utilizando a DFT foram necessários 2.017 seg, enquanto que utilizando a FFT o tempo cai para 8,32 seg. Em termos de precisão, a eficiência da FFT também é notada, resultando num erro médio quadrático da ordem de 10^{-6}, enquanto na DFT a precisão cai para 10^{-5}.

Muitos algoritmos de FFT foram desenvolvidos a partir da idéia original de Cooley & Tukey explorando alguma particularidade da seqüência de entrada, ou baseados em fundamentos matemáticos diferentes [12]. Entre eles podemos citar os de Gentleman & Sande [13] em 1966, Pease [14] em 1968, Singleton [15] em 1969, Uhrich [16] em 1969, e Temperton [17] em 1976. Cada algoritmo possui peculiaridades que o tornam mais ou menos adequado a determinada aplicação. Tais peculiaridades permitem que os algoritmos adquiram características próprias tais como maior ou menor velocidade de processamento, em contrapartida menor ou maior precisão, ou exigir maior ou menor espaço de memória para o processamento.

Em geral, os usuários de transformadas de Fourier não levam em conta as características dos algoritmos, limitando-se a utilizar um determinado método ou um certo *pacote* com o qual estão familiarizados, em todas as situações, sem uma análise mais cuidadosa que lhes permitiria alcançar maior eficiência e obter resultados de melhor qualidade. Essa análise, associada a do comportamento numérico dos algoritmos, não é trivial, requerendo do usuário um grau de especialização em mais de uma área de conhecimento, o que na maioria das situações não acontece. Dessa forma, procurou-se ouvir especialistas em transformadas de Fourier, análise numérica e matemática computacional, e também usuários, a fim de obter parâmetros para orientar o processo de decisão na escolha de um determinado algoritmo numa certa aplicação. Esse conhecimento, associado a informações oriundas de um processo de simulação (ver seção 3.4), constitui a base de conhecimentos do sistema especialista FFTEX, capacitando-o a fornecer a transformada de Fourier de uma seqüência unidimensional ou bidimensional, de maneira fácil, cômoda e eficiente, calculada pelo algoritmo mais adequado a uma situação dada.

O FFTEX possui ainda a propriedade da maleabilidade, isto é, é capaz de se adaptar ao uso a que se destina. Conforme já foi mencionado, os algoritmos foram desenvolvidos tirando proveito de determinadas particularidades; assim, um algoritmo considerado eficiente numa certa aplicação pode não ser em outra. Uma decisão baseada em preceitos rígidos poderia levar a uma escolha sem garantia de eficiência. Por exemplo, as seqüências oriundas de sinais de uso metereológico possuem natureza diferente das de sinais para processamento de voz, de maneira que um algoritmo considerado eficaz para o primeiro caso não necessariamente o será para o segundo. Quanto mais usado, mais o sistema irá se adaptar às particularidades das seqüências usadas, e suas decisões tendem a ser ótimas.

As seções seguintes fazem a descrição do sistema, tecem algumas considerações sobre os algoritmos que compõem o FFTEX, descrevem o processo de simulação utilizado, entre outras coisas, como base para a obtenção do arquivo de desempenho das rotinas, e tecem considerações sobre o aprendizado automático e a maleabilidade do sistema.

3.2 Descrição do Sistema

O FFTEX é um sistema especialista cuja finalidade é de calcular, de maneira eficiente, uma transformada de Fourier utilizando um dos algoritmos (o mais adequado a uma determinada situação) existentes no seu banco de algoritmos. O usuário situa facilmente o problema seguindo a orientação fornecida pelo sistema. A escolha do algoritmo é feita levando-se em conta três fatores, tempo de processamento, espaço de memória requerido, e precisão dos resultados. O usuário fornece o peso que cada um desses fatores tem na solução de seu problema. Baseado nesses pesos, o sistema escolhe e ativa o algoritmo de cálculo. O sistema permite o cálculo da FFT por mais de um algoritmo, dependendo do desejo do usuário.

Basicamente o FFTEX é composto de duas partes com funções distintas, acopladas entre si. Uma parte simbólica constituída do controle do sistema, da interface com o usuário, da representação do conhecimento (base de conhecimentos), do motor de inferência, e do módulo de aprendizado. A outra parte, a parte numérica, é responsável pela computação numérica das transformadas. Esquematicamente o FFTEX pode ser representado conforme a figura 3.1.

O módulo de controle é o responsável pelo controle geral do sistema e pela ativação dos demais módulos, executando a ligação da parte simbólica com a parte numérica, e ainda, responsabilizando-se pela entrada e saída de dados.

A base de conhecimentos armazena informações genéricas sobre os problemas de domínio do sistema. Essas informações foram obtidas do conhecimento extraído de pesquisa à literatura especializada, de entrevistas a especialistas e usuários de FFT, e pela observação do comportamento dos algoritmos em simulação de casos reais.

A memória de trabalho é uma base de dados que contém informações específicas sobre o problema em questão, os dados de entrada e os resultados intermediários de operações anteriores.

O motor de inferência efetua a seleção e o casamento entre informações contidas na memória de trabalho e na base de conhecimentos, gerando o conjunto de conflito, cuja solução pode modificar o conteúdo da memória de trabalho, enviar e/ou solicitar informações, ou ativar as rotinas numéricas.

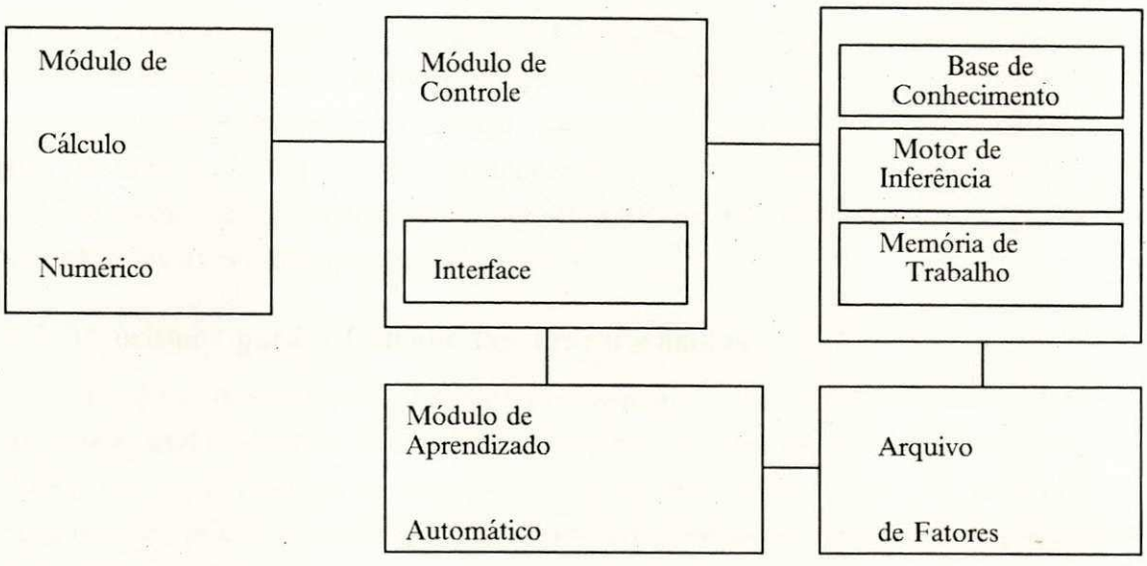

Figura 3.1: A Estrutura do FFTEX

O módulo de aprendizado automático permite ao sistema se adaptar à situação particular do usuário. A adaptação é feita através de modificações no arquivo de fatores, que orienta o sistema na decisão de escolha de um determinado algoritmo.

O arquivo de fatores contém o fator de desempenho de cada algoritmo em termos de tempo de processamento, espaço de memória e precisão dos resultados. Esse fator é determinado pelo desvio da média dos desempenhos obtidos por todos os algoritmos semelhantes, aplicáveis a uma determinada classe de problemas em execuções passadas. Inicialmente, os desempenhos são medidos baseados nos resultados dos algoritmos em cálculos simulados de FFT, de seqüências geradas aleatoriamente com tamanhos variados, abrangendo todo o escopo das rotinas implementadas.

O módulo de cálculo numérico contém os programas que implementam os algoritmos para a resolução numérica dos problemas, isto é, os algoritmos para a computação numérica das transformadas e transformadas inversas de Fourier, em uma e duas dimensões. A seção 3.3 fornece maiores detalhes sobre o processo de seleção e os algoritmos constantes no FFTEX. Para ligação com a parte simbólica, cada programa possui uma interface específica, cuja finalidade principal é de adequar os dados de entrada fornecidos pela parte simbólica (usuário + dedução) à necessidade específica de cada rotina, e vice-versa, adequando a resposta obtida na computação numérica a uma forma aceitável (padronizada) pela manipulação simbólica.

A parte simbólica está implementada em linguagem C, enquanto a parte numérica foi implementada utilizando um sub-conjunto do FORTRAN ANSI conhecido como PFORT [18]. O PFORT foi escolhido porque o FORTRAN oferece melhor controle das operações numéricas e permite alcançar portabilidade.

Atualmente o sistema é executável em microcomputadores compatíveis com o IBM PC/XT/AT, com configuração mínima de 1 Mega byte de memória, sendo recomendado, para melhor desempenho, o uso de co-processador aritmético. Uma versão do FFTEX está sendo preparada para executar em ambiente UNIX. Uma outra, para mainframes, está sendo pensada. A principal dificuldade está na portabilidade da interface (tela), pois a parte numérica foi implementada em PFORT visando justamente a adaptação do sistema a diversos ambientes.

3.3 Algoritmos para o Cálculo das Transformadas

Devido a diversidade de algoritmos existentes para transformação de Fourier, decidiu-se adotar algumas normas para selecionar aqueles que iriam compor o sistema. Assim, um algoritmo para ser incorporado ao sistema deveria passar por duas fases de seleção. Na primeira, um algoritmo seria selecionado se satisfizesse os seguintes requisitos

- ser consagrado pela literatura
- possuir uma boa documentação
- ser adaptável segundo o conceito de Smith [19] (requisito fundamental à transportabilidade).

Satisfeitas essas condições, passou-se então à segunda fase do processo, a fase de simulação (ver seção 3.4) que teve uma dupla função, a de servir de balizamento para o processo de seleção (conforme o desempenho um algoritmo é aceito ou rejeitado), e servir como fonte de informação sobre os algoritmos selecionados. Tais informações iriam compor a base de conhecimento, juntamente com outras informações (algumas delas conflitantes com as obtidas na prática) oriundas de pesquisa a literatura especializada, e informações obtidas em entrevistas com especialistas e usuários de transformadas de Fourier.

Boa parte dos algoritmos selecionados fazem parte da coletânea de programas sobre processamento digital de sinais do IEEE [4]. A coleção de programas pode ser adquirida diretamente do IEEE.

Embora os programas do IEEE, na sua maioria, sejam bem escritos, foram necessárias algumas modificações e correções para torná-los operacionais e mais eficientes. Como exemplo, foram eliminados os ineficientes IF's aritméticos presentes na maioria dos programas, as conversões intrínsecas de tipo, e variáveis para as quais não eram atribuídos valores iniciais. Também foram introduzidas modificações nas implementações originais, visando padronizar todos os programas selecionados, seguindo uma rígida disciplina de programação ditada pelos padrões da biblioteca BITAN [20].

Outras modificações inseridas nos programas, em geral, tiveram a finalidade de automatizar e minorar o esforço de passagem de parâmetros às rotinas. Entre essas

modificações pode-se citar a introdução de uma rotina para fatoração de N (número de elementos da seqüência) no programa que implementa o algoritmo de Temperton, que originalmente exigia do usuário o fornecimento da seqüência dos fatores de N.

Até o presente momento os algoritmos constantes no FFTEX são os abaixo relacionados, em que FFT se refere à transformação e IFFT à transformação inversa.

a) Algoritmos para seqüências unidimensionais

a.1- FOUREA (Rader [4]). FFT-IFFT para uma seqüência de dados complexos. O número de elementos N deve ser uma potência de 2.

a.1.1- FOURER. Versão do programa FOUREA para seqüências de dados reais.

a.2- FASTFR (Bergand & Dolan [4]). FFT-IFFT para uma seqüência de dados reais. Utiliza bases 2 e 4.

a.3- FOUFAR (Bergand & Dolan [4]). FFT-IFFT para uma seqüência de dados reais. Utiliza bases 2, 4 e 8.

a.4- FFT842 (Bergand & Dolan [4]). FFT-IFFT para uma seqüência de dados complexos. Utiliza bases 2, 4 e 8.

a.4.1- FFT84R. Versão do programa FFT842 para seqüências de dados reais.

a.5- SINGLE (Singleton [15]). FFT-IFFT para uma seqüência real ou complexa em que o número N de elementos não necessariamente é potência de 2.

a.5.1- SINGER. Versão do programa SINGLE para seqüências de dados reais.

a.6- WFFAST (McClellan & Nawab [4]). FFT-IFFT para uma seqüência real ou complexa, em que N, o número de elementos da seqüência, é o produto de fatores relativamente primos tomados do conjunto {2, 3, 4, 5, 7, 8, 9, 16}.

a.6.1- WFFASR. Versão do programa WFFAST para seqüências de dados reais.

a.7- RADIX4 (Morris [4]). FFT-IFFT para uma seqüência real ou complexa. Utiliza bases 2 e 4. Eficiente para aplicações onde uma transformada de tamanho fixo é executada repetidamente, tais como análise/síntese de voz, processamento de imagens, etc.

a.7.1- RADIXR. Versão do programa RADIX4 para seqüências de dados reais.

a.8- TEMPER (Temperton [17]). FFT para uma seqüência real ou complexa. Utiliza base mista (N não necessariamente é uma potência de 2).

a.8.1- TEMPRR. Versão do programa TEMPER para seqüências de dados reais.

a.9- DFTEOC. Implementa a transformada de Fourier de uma maneira direta. Usado para pequenas seqüências reais ou complexas.

a.9.1- DFTEOR. Versão do programa DFTEOC para seqüências de dados reais.

b) Algoritmos para seqüências bidimensionais

b.1- SING2D. FFT-IFFT de seqüência de reais ou complexas de duas dimensões usando base mista. Utiliza a rotina FFT contida no programa SINGLE.

b.1.1-SIN2DR. Versão do programa SING2D para seqüências de dados reais.

b.2- RADI2D. FFT-IFFT de seqüência de reais ou complexas de duas dimensões usando base 4. Utiliza a rotina RADIX4 de Morris.

b.2.1-RAD2DR. Versão do programa RADI2D para seqüências de dados reais.

b.3- FFA82D. Combina as rotinas FFA e FFT842 de Bergand & Dolan [4] para produzir uma efhciente maneira de se calcular a FFT-IFFT de uma seqüência real bidimensional com $N = N_1 \times N_2$, onde N_1 e N_2 devem ser potências de 2.

b.4- DFTE2D. Implementa a transformação de Fourier de seqüências bidimensionais de uma maneira direta. Usado para pequenas seqüências reais ou complexas.

b.4.1-DFT2DR. Versão do programa DFTE2D para seqüências de dados reais.

É bom lembrar que os algoritmos para seqüências unidimensionais podem ser utilizados no caso bidimensional, processando linhas e colunas sucessivamente, ou seja,

$$X(k, r) = \sum_{n=0}^{N-1} G(k,n) W_n^{rn}, \quad k = 0, 1, \ldots, M-1, \quad r = 0, 1, \ldots, N-1 \tag{2}$$

onde

$$G(k,r) = \sum_{m=0}^{M-1} x(m,n) W_m^{km} \tag{3}$$

Note que o resultado da equação (3) alimenta a equação (2) e que $G(k, n)$ será uma seqüência complexa independente da seqüência original (x) ser real ou complexa.

A função $G(k, n)$ consiste de N transformadas de seqüências unidimensionais, uma para cada coluna de $x(m, n)$. A transformada em duas dimensões, $X(k, r)$, é obtida implementando M transformadas de seqüências unidimensionais, uma para cada linha da seqüência $G(k, n)$.

Os algoritmos do grupo b), específicos para seqüência em duas dimensões, são na verdade algoritmos para uma dimensão combinados. O algoritmo FFA82D combina dois dos mais velozes algoritmos de FFT que utilizam base 2, um próprio para seqüência de entrada real (usa a equação 3) e outro para seqüência complexa (calcula a equação 2). O algoritmo RADI2D utiliza de uma maneira ótima o fato da rotina RADIX4 de Morris não necessitar de redefinições dos valores de variáveis no cálculo de transformações repetitivas. O algoritmo SING2D é apropriado para uma seqüência qualquer, em vez de

chamar N vezes um algoritmo para seqüências unidimensionais para os cálculos da equação (2), o faz de uma maneira eficiente de modo que com apenas três chamadas da rotina FFT, independente do tamanho da seqüência, calcula a bitransformação.

3.4 Simulação

Várias razões contribuíram para a decisão de se lançar mão de um processo de simulação de casos reais para auxiliar a construção do FFTEX. Entre elas, a falta de informações sobre o comportamento dos algoritmos nas diversas aplicações a que se destina o sistema; as informações existentes nem sempre puderam ser comprovadas na prática, entrando em conflito, algumas vezes, com os resultados obtidos; a necessidade de comparação entre o desempenho dos algoritmos exigia uma uniformidade de condições na aplicação dos testes.

Três parâmetros servirão para avaliação dos algoritmos: tempo de processamento, espaço de memória necessário e precisão dos resultados obtidos.

Em primeiro lugar separou-se os algoritmos por classe de problemas que propunha resolver, ou seja, aplicável a seqüências de entrada unidimensionais ou bidimensionais, a seqüências exclusivamente reais ou complexas, aplicável a seqüências de tamanho N, para o caso de seqüências unidimensionais, sendo N potência de 2 e 4 (base 4), potência de 2 (base 2) ou não (base mista), ou a seqüências de tamanho $N_1 \times N_2$ para o caso bidimensional, sendo N_1 e N_2 potências de 2 e 4, potências de 2 ou não.

Pela uniformidade dos testes, e levando-se em conta o fato de que muitos algoritmos terem comprovadamente comportamentos discrepantes conforme a seqüência de entrada aplicada, decidiu-se pelo uso, para todos os testes, de uma seqüência gerada utilizando a função UNI de Alan Gross [4], que produz uma seqüência pseudo-aleatória de valores satisfazendo uma distribuição uniforme em [0,1].

Os números de amostras utilizados foram 16, 64, 256, 1024 e 4096, para seqüências unidimensionais, exceto para o teste com as rotinas WFFAST e WFFASR cujas amostras foram de tamanho 16, 63, 240, 1008 e 2520 (sendo também repetidos para as rotinas SINGER, SINGLE, TEMPRR e TEMPLE, que utilizam bases mistas). Para seqüências bidimensionais foram tomadas matrizes 16×16, 16×64, 64×16, 64×64, 64×128, 128×128 (as duas últimas não foram aplicadas às rotinas RAD2DR e RADI2D).

Os compiladores utilizados foram o WATFIV para *mainframes* IBM, FORTRAN VS da IBM, FORTRAN 4.01 da Microsoft e WATFOR 77 para PC/DOS. Os dois últimos foram usados com e sem co-processador aritmético num XT 8086 de 8MHz e num AT 80286 de 10MHz.

A precisão dos resultados foi avaliada comparando-se a seqüência de entrada com a transformada inversa dessa mesma seqüência. Em outras palavras, em cada teste transformou-se a seqüência e aplicou-se a transformada inversa para se poder medir a

discrepância. A medida de erro utilizada foi a raiz quadrada da média dos quadrados da diferença entre cada elemento da seqüência de entrada e o correspondente da seqüência obtida da transformação inversa. Esse critério foi adotado por fornecer uma informação mais significativa sobre toda a seqüência, ao contrário da critério mini-max (norma infinita), adotado por alguns autores, em que a anomalia de um único ponto pode mascarar a precisão conseguida nos demais pontos.

Além de servir de base para a construção do arquivo de fatores (ver próxima seção) e servir como fonte de conhecimentos sobre os algoritmos, os testes simulados desempenharam um papel importante na construção do sistema, influindo inclusive na sua arquitetura.

Algumas conclusões importantes, que podem ser encontradas com mais detalhes em Hattori at al. [21], serão citadas a seguir:

a) A variação do tempo de execução medido para um mesmo problema é significativa, chegando a mais de 10%. Isso implica que para se comparar o tempo de execução de duas rotinas é necessário resolver diversas vezes um mesmo problema e adotar a média dos tempos como medida de comparação.
b) Uma implementação de um algoritmo *A* que se mostra mais eficiente que a implementação de um outro algoritmo *B* num determinado ambiente (hardware, sistema operacional e compilador) pode não sê-la em outro.
c) Os testes de avaliação para serem significativos podem se tornar extremamente dispendiosos. No caso, para cada opção de compilador foram necessárias 345 execuções. No total foram 2.070 execuções.

3.5 Aprendizado Automático

Conforme já se mencionou, os algoritmos de FFT foram desenvolvidos explorando alguma particularidade da seqüência de entrada ou baseados em fundamentos matemáticos diferentes. Com isso, algumas peculiaridades foram introduzidas nos algoritmos, de modo que um algoritmo que se mostra mais eficiente que os outros para um determinado tipo de seqüência de entrada, pode não sê-lo para outro tipo de seqüência. Até mesmo a mudança de ambiente computacional (hadware + sistema operacional + compilador) pode influir no desempenho dos algoritmos (veja Hattori et al. [21]). de tal forma, não se tem uma regra geral para classificar o desempenho dos algoritmos principalmente no que diz respeito ao tempo de processamento e a precisão dos resultados.

Observando esse comportamento é que se procurou dar ao FFTEX a capacidade de se ajustar ao ambiente e ao uso a que se destina, baseando esse ajuste na observação do comportamento dos algoritmos em casos passados, devidamente registrados pelo próprio sistema e, traduzido sob a forma de fatores no arquivo de fatores.

3.5.1 Geração e Atualização do Arquivo de Fatores

O arquivo de fatores é composto de registros agrupados por classe de problema. Cada registro corresponde a um algoritmo aplicável a uma classe de problemas. Contém os pesos relativos ao respectivo algoritmo em termos de tempo de processamento, espaço de memória e precisão dos resultados. A Tabela 3.1 esquematiza o arquivo de fatores.

Tabela 3.1: Arquivo de Fatores

Classe 1: Seqüências unidimensionais estritamente reais N. potência de 2 e 4	registro	1	(FOURER):	P_t	P_p	P_e
	"	2	(FASTFR):	"	"	"
	"	3	(FOUFAR):	"	"	"
	"	4	(FFT84R):	"	"	"
	"	5	(SINGER):	"	"	"
	"	6	(RADIXR):	"	"	"
	"	7	(TEMPRR):	"	"	"
	"	8	(DFTEOR):	"	"	"
Classe 2: Seqüências unidimensionais estritamente reais. N potência de 2	registro	1	(FOURER):	P_t	P_p	P_e
	"	2	(FASTFR):	"	"	"
	"	3	(FOUFAR):	"	"	"
	"	4	(FFT84R):	"	"	"
	"	5	(SINGER):	"	"	"
	"	6	(TEMPRR):	"	"	"
	"	7	(DFTEOR):	"	"	"
Classe 3: Seqüências unidimensionais estritamente reais. N não é potência de 2	registro	1	(WFFASR):	P_t	P_p	P_e
	"	2	(SINGER):	"	"	"
	"	3	(TEMPRR):	"	"	"
	"	4	(DFTEOR):	"	"	"
Classe 4: Seqüências unidimensionais complexas. N potência de 2 e 4	registro	1	(FOUREA):	P_t	P_p	P_e
	"	2	(FFT842):	"	"	"
	"	3	(SINGLE):	"	"	"
	"	4	(RADIX4):	"	"	"
	"	5	(TEMPLE):	"	"	"
	"	6	(DFTEOC):	"	"	"
Classe 5: Seqüências unidimensionais complexas. N potência de 2	registro	1	(FOUREA):	P_t	P_p	P_e
	"	2	(FFT842):	"	"	"
	"	3	(SINGLE):	"	"	"
	"	4	(TEMPLE):	"	"	"
	"	5	(DFTEOC):	"	"	"
Classe 6: Seqüências unidimensionais complexas. N não é potência de 2	registro	1	(WFFAST):	P_t	P_p	P_e
	"	2	(SINGLE):	"	"	"
	"	3	(TEMPLE):	"	"	"
	"	4	(DFTEOC):	"	"	"
Classe 7: Seqüências bidimensionais ($N_1 \times N_2$) reais. N_1, N_2 potências de 2 e 4	registro	1	(FFT82D):	P_t	P_p	P_e
	"	2	(SIN2DR):	"	"	"
	"	3	(RAD2DR):	"	"	"
	"	4	(DFT2DR):	"	"	"

Tabela 3.1: Arquivo de Fatores (continuação)

Classe 8: Seqüências bidimensionais ($N_1 \times N_2$), reais	registro 1 (FFT82D):	P_t	P_p	P_e
	" 2 (SIN2DR):	"	"	"
	" 3 (DFT2DR):	"	"	"
Classe 9 : Seqüências bidimensionais ($N_1 \times N_2$), reais. N_1, N_2 não são potências de 2	registro 1 (SIN2DR):	P_t	P_p	P_e
	" 2 (DFT2DR):	"	"	"
Classe 10: Seqüências bidimensionais complexas. N_1, N_2 potências de 2 e 4	registro 1 (FFT82D):	P_t	P_p	P_e
	" 2 (SING2D):	"	"	"
	" 3 (RADI2D):	"	"	"
	" 4 (DFTE2D):	"	"	"
Classe 11: Seqüências bidimensionais complexas. N_1, N_2 potências de 2	registro 1 (FFT82D):	P_t	P_p	P_e
	" 2 (SING2D):	"	"	"
	" 3 (DFTE2D):	"	"	"
Classe 12: Seqüências bidimensionais complexas. N_1, N_2 não são potências de 2	registro 1 (SING2D):	P_t	P_p	P_e
	" 2 (DFTE2D):	"	"	"

Devido à falta de maiores informações sobre o comportamento dos algoritmos e pela diversidade de aplicações a que se destina o sistema, procurou-se gerar essas informações a partir da simulação de casos reais, com a preocupação de não serem tendenciosas. Para isso optou-se pelo uso de uma seqüência gerada aleatoriamente, à qual se aplicaria, sob condições uniformes, todos os algoritmos de transformação testados. Dessa forma, os FFTEX's para todas as aplicações a que se destinam, partem inicialmente com o mesmo arquivo de fatores. Com a utilização eles vão se diferenciando, sendo as informações contidas no arquivo de fatores original, pouco a pouco substituídas por novas informações extraídas do arquivo de ocorrências, arquivo este gerado pelo próprio sistema, que armazena as informações de desempenho dos algoritmos nas aplicações particulares dos usuários. A atualização do arquivo de fatores é feita automaticamente pelo módulo de aprendizado após constatar, em exame completo do arquivo de ocorrências, a existência de um número X (a ser determinado) de registros de casos semelhantes. A atualização do arquivo de fatores é feita apenas nos registros correspondentes a classe a que pertence o caso em questão.

Seja na criação do arquivo de fatores original a partir das informações oriundas dos testes simulados, ou na atualização deste a partir das informações obtidas com o processamento de casos reais armazenadas no arquivo de ocorrências, o cálculo dos fatores relativos a um algoritmo é feito da mesma forma segundo o algoritmo abaixo:

1) Identificar todos os casos semelhantes (seqüências de mesmo tamanho, de mesma natureza, estritamente reais ou complexas, e unidimensionais ou bidimensionais) no arquivo de ocorrências.
2) Para cada algoritmo, calcular a média dos tempos de processamento gastos, precisão dos resultados obtidos, avaliado pelo erro médio quadrático entre a seqüência inicial e a transformada inversa da seqüência transformada, e espaços de memória requeridos.
3) Calcular a média global de tempo, precisão e espaço entre todos os algoritmos da mesma classe.
4) Calcular o desvio relativo de cada algoritmo, dividindo as médias de tempo, precisão e espaço referentes ao algoritmo, pela média global respectiva. Os resultados serão os novos fatores do algoritmo.
5) Identificar no arquivo de fatores o registro correspondente à classe de problema e ao algoritmo, substituindo os valores existentes pelos novos fatores.
6) Apagar as informações processadas no arquivo de ocorrências, colocando em seu lugar as médias obtidas por cada algoritmo. Tais médias representarão um caso a mais no próximo ajuste.

3.5.2 Classificação dos Algoritmos

A ativação de um determinado algoritmo dentro de uma classe é feita obedecendo uma classificação feita conforme o coeficiente de desempenho (CD) de cada um. Esse coeficiente reflete o fato de que quanto menos recursos utilizar melhor o desempenho e é calculado utilizando a equação

$$CD = P_{temp} \times P_t + P_{prec} \times P_p + P_{espac} \times P_e$$

onde, P_{temp}, P_{prec} e P_{espac} são os fatores de importância do tempo de processamento, precisão dos resultados e espaço de memória requerido, na resolução do problema específico. Tais fatores são fornecidos pelo usuário sob a forma de porcentagem. P_t, P_p e P_e são fatores de desempenho do algoritmo corespondentes ao tempo, à precisão e ao espaço, obtidos consultando o arquivo de fatores.

A ativação dos algoritmos obedece a uma ordem de prioridade dada pelo coeficiente de desempenho de cada um.

3.6 Conclusão

O FFTEX é um sistema acoplado, isto é, com acoplamento entre a computação simbólica e numérica. Por ser um sistema acoplado, possui características próprias que o distinguem de um sistema especialista usual (Kitzmiller [3]). Considere por exemplo um sistema de diagnósticos clínicos. Com as informações obtidas de maneira interativa com o usuário, pode chegar a um possível diagnóstico e, em alguns casos, aplicar uma receita visando a cura do paciente. Num sistema acoplado, as informações são processadas e também chega-se a um *diagnóstico*, no caso, a escolha de um determinado algoritmo. Só que o trabalho não para aí, pois a partir da escolha, o controle passa à parte numérica, com a ativação da rotina que implementa o algoritmo escolhido. A seguir, com os cálculos numéricos executados, o controle retorna à computação simbólica onde se inicia um processo de validação com a análise dos resultados obtidos. É como se o sistema de diagnósticos obtivesse do paciente a resposta sintomática e instantânea dos efeitos da receita prescrita, para a partir daí, processar-se a análise dessa resposta.

Num sistema acoplado, como em qualquer sistema especialista, deseja-se que a decisão de escolha seja a melhor possível. Por isso a capacidade de aprender com os erros e acertos dessa escolha é uma característica bastante desejável. O FFTEX possui essa característica, aprendendo com suas próprias respostas. O módulo de aprendizado registra as respostas no arquivo de ocorrências. Após constatar a existência de respostas a um certo número de casos semelhantes, o arquivo de fatores, base da decisão de escolha, é modificado em função dessas respostas. Com isso espera-se que o uso continuado do sistema faça com que a decisão de escolha tenda a ser ótima.

O número de casos semelhantes registrados no arquivo de ocorrências que provoca o processo de modificação do arquivo de fatores, não está determinado, por não se ter nenhuma informação que possa servir de parâmetro, devido ao caracter pioneiro do sistema. Em virtude desse fato, protótipos do FFTEX deverão ser implementados em alguns centros de pesquisa de natureza distinta.

O acompanhamento do desempenho do sistema com anotações das ocorrências e sugestões apresentadas pelos diversos pesquisadores das instituições selecionadas, terá um papel fundamental para a elaboração da versão final do FFTEX.

CAPÍTULO 4

O SISTEMA SIPREX

4.1 Introdução

O SIPREX, um sistema especialista a ser utilizado no projeto e análise de filtros digitais, é um sistema acoplado constituído de uma estrutura modular, que foi concebido como resultado de estudos e da experiência adquirida com a construção de outros protótipos dentro da linha de sistemas acoplados (EXBITAN e FFTEX). Está dividido em três partes interligadas, com funções específicas. A primeira parte é constituída de uma estrutura de interligação do sistema, que tem a função de interagir com o usuário e promover a ligação com os demais módulos. É composta de uma interface inteligente e de um módulo de controle. A segunda parte, a estrutura de computação simbólica, é a responsável pela computação simbólica do sistema. É composta pelos módulos central, de explanação e de aprendizado automático. A terceira parte é constituída pela estrutura de computação numérica, composta pelo módulo numérico e pelo módulo gráfico. Tem a função de realizar os cálculos numéricos do sistema. As três estruturas serão descritas com detalhes nos três capítulos seguintes. Neste mesmo capítulo, nas seções seguintes, serão descritas a motivação para a construção deste sistema e sua arquitetura.

4.2 Motivação

De uma maneira geral, o campo de Processamento Digital de Sinais pode ser dividido em duas sub-áreas principais: Filtragem Digital e Análise Espectral. Para ilustrar a motivação por trás do presente projeto de pesquisa, serão considerados os aspectos envolvidos na primeira dessas sub-áreas, ou seja, no projeto, implementação e aplicação de filtros digitais.

O primeiro passo do processo de síntese de um filtro digital é a obtenção, a partir da aplicação desejada, de um conjunto de especificações que caracterizam o comportamento do filtro. Essas especificações podem ser obtidas no domínio do tempo (resposta ao impulso ou resposta ao degrau), ou no domínio da freqüência (freqüências de corte, desvio máximo nas faixas de passagem, linearidade de fase, etc.). Em função dessas especificações, uma escolha é então feita entre um dos dois tipos básicos de filtros digitais: FIR (Resposta ao Impulso Finita) e IIR (Resposta ao Impulso Infinita). Filtros FIR com fase exatamente linear podem ser facilmente obtidos, sendo os mais indicados para aplicações onde dispersão de freqüências, devido à não-linearidade de fase, é prejudicial (por exemplo, processamento de sinais de voz e comunicações de dados). Filtros IIR, por sua vez, são em geral computacionalmente mais eficientes que os FIR,

necessitando de um menor número de operações por amostra para satisfazer um dado conjunto de especificações. A escolha, portanto, vai depender das prioridades atribuídas a esses e outros fatores e do compromisso a ser atingido entre eles.

Uma vez escolhido o tipo de filtro mais adequado para a aplicação desejada, o seu projeto pode ser decomposto em quatro etapas básicas:

1) Resolução de um problema de aproximação, de modo a determinar coeficientes para o filtro, que satisfaçam as especificações dadas.
2) Escolha de uma estrutura específica para realização do filtro e quantização dos coeficientes, isto é, representação dos coeficientes em palavras com tamanho finito.
3) Quantização das variáveis do filtro, isto é, variáveis de entrada, saída e variáveis intermediárias num tamanho determinado de palavra.
4) Utilizando simulação, verificar se o filtro resultante satisfaz as especificações dadas.

Os resultados da etapa 4 podem levar a revisões e repetições das etapas 2 e 3, visando melhorar o desempenho do filtro.

Maiores detalhes sobre o procedimento geral acima descrito, podem ser encontrados em [22] e [23].

Do acima exposto, pode-se observar que cada etapa do processo de síntese de um filtro digital envolve uma tomada de decisão, sempre visando obter o projeto final que melhor satisfaça as especificações originais. Essas tomadas de decisão exigiriam, idealmente, do projetista do filtro, conhecimento detalhado não só dos tipos básicos de filtros e suas características, como também dos vários métodos existentes para projetos de filtros digitais. Na prática, contudo, na maioria das vezes esse conhecimento não existe, limitando-se o projetista a utilizar técnicas com as quais está familiarizado, ou para as quais dispõe de pacotes de software (sub-rotinas) de fácil uso. Esta situação é compreensível, se levarmos em conta que as técnicas de projeto de filtros digitais disponíveis, principalmente no caso de filtros ótimos, utilizam ferramentas como cálculo numérico, teoria da otimização e programação linear com as quais o engenheiro comum, em geral, está familiarizado, quando está, de uma maneira apenas superficial. O resultado é um filtro cujo desempenho embora aceitável, poderá estar abaixo do ótimo possível para aquela dada aplicação. Daí a vantagem de se ter um sistema capaz de armazenar o conhecimento específico de cada especialista e adequá-lo a uma determinada situação. Isso é nada mais nada menos que a definição de sistema especialista.

Além do método usado para determinar os coeficientes de um filtro digital de modo a aproximar um conjunto de especificações dado, dois outros fatores possuem considerável influência sobre o desempenho: (a) a estrutura específica a ser usada na

implementação e (b) o tamanho do registro utilizado na quantização dos coeficientes e variáveis do filtro (etapas 2 e 3 da relação acima).

De uma maneira geral, os coeficientes de um filtro digital são obtidos por meio de um procedimento teórico utilizando uma representação com precisão infinita. Na prática, esses coeficientes devem ser quantizados em um número fixo de bits, correspondendo ao tamanho do registro utilizado. Em conseqüência, uma diferença vai existir entre a resposta em freqüência do filtro implementado na prática e a do filtro que seria obtido se um registro de tamanho infinito estivesse disponível.

Os efeitos da quantização de coeficientes e variáveis, seja por arredondamento ou por truncamento, no desempenho de um filtro digital vai depender do tipo de estrutura usada na implementação (recursiva ou não-recursiva), do tipo de aritmética utilizada (ponto fixo ou ponto flutuante) e da maneira como os números são representados (sinal e magnitude, complemento de dois e complemento de 1). Novamente, para que a escolha ótima possa ser feita, fatores devem ser considerados (como por exemplo a função densidade de probabilidade do erro de quantização para cada uma das opções acima descritas) que geralmente escapam ao conhecimento do engenheiro comum. O resultado é uma utilização não-otimizada das técnicas e algoritmos disponíveis, gerando sistemas com desempenho abaixo do ótimo.

Então, a construção de um sistema especialista com a finalidade de auxiliar no projeto ótimo de um filtro será uma solução bastante desejável. Depara-se agora com a dificuldade de se construir um sistema especialista capaz de manipular informações, ativar algoritmos, lidar e controlar instabilidades numéricas e analisar as respostas dos algoritmos.

A inadequação das linguagens que manipulam proposições lógicas na implementação dos algoritmos numéricos e vice-versa, aliado ao necessário domínio do ambiente computacional e sua influência na instabilidade numérica dos algoritmos, resultam na dificuldade de construção de sistemas especialistas com acoplamento entre computação simbólica e computação numérica.

O que se pretende com esse trabalho é fornecer subsídios à construção de sistemas especialistas com acoplamento entre computação simbólica e numérica, promovendo estudos com a finalidade de obter uma modelagem adequada para a representação e manipulação do conhecimento utilizando adequadamente técnicas como regras de produção e árvores semânticas. O produto final será um sistema especialista (SIPREX), cuja finalidade é orientar o engenheiro de processamento digital de sinais, na utilização ótima dos métodos existentes para análise e síntese de sinais e sistemas digitais. Esse sistema terá utilidade em áreas de aplicação de processamento digital de sinais, como telecomunicação, estudos de sinais biomédicos, automação e outras.

A construção de sistemas especialistas acoplados, principalmente os de aplicação à engenharia, tem sido feita de maneira incipiente e de certa forma artesanal, sem explorar toda a potencialidade das técnicas de IA. A maioria dos sistemas desenvolvidos até o momento apenas utilizam uma base de conhecimentos para auxiliar o processo de seleção de algoritmos, como o sistema de Monard [24] que auxilia a utilização das rotinas da biblioteca NAG, para auxiliar na seleção de comandos e especificação de argumentos, como o sistema de Sueda [6] utilizado para o processamento de imagens, ou para realizar uma tarefa específica do sistema, como o caso do sistema de Gandelin [25] para reconhecimento de organismos marinhos a partir de imagens monoculares. Este último exemplo, utiliza uma base de conhecimentos para reconhecer objetos com ajuda de curvas de contorno, obtida com a utilização de métodos numéricos. Alguns sistemas utilizam interfaces inteligentes com alguma sofisticação, como o caso do Gabriel [26], que embora não sendo propriamente um sistema acoplado, possui uma interface gráfica inteligente que auxilia o usuário a especificar, a partir de blocos pré-definidos, a estrutura de implementação pretendida para o sistema digital.

Mesmo produtos comerciais já foram também desenvolvidos com a finalidade de projetar filtros digitais, como o comercializado pela Monarch [27]. Esses pacotes trazem uma interface amigável na definição dos parâmetros de um determinado programa, podendo exibir os resultados de forma gráfica. Entretanto, nenhuma inteligência é utilizada, sendo um algoritmo ativado de cada vez, sem nenhum conhecimento prévio sobre o mesmo e sem qualquer tipo de análise dos resultados obtidos.

Um outro caso bem peculiar de sistemas acoplados de aplicação à engenharia, é o sistema para projeto de filtros digitais desenvolvido por Nie at al. [28]. Tal sistema é composto de três módulos que trabalham independentemente, sem conexão entre eles: um contém algoritmos de projeto de filtros, um outro é capaz de analisar filtros projetados, e um terceiro módulo, consistindo de um sistema especialista cuja função é orientar o usuário na decisão sobre qual dos algoritmos utilizar, bem como relacionar os parâmetros que devem ser fornecidos para que se possa ativar o referido algoritmo. É bom observar que o acoplamento entre a parte de manipulação simbólica e a numérica, neste caso, é realizado pelo próprio usuário, não ocorrendo nenhuma comunicação direta entre os módulos.

Por outro lado, o SIPREX é um sistema completo, sob o ponto de vista de estrutura de sistemas acoplados. Nele estão presentes uma interface inteligente com a preocupação de obter do usuário as especificações do problema de uma maneira amigável e consistente, uma base de conhecimentos com a função de selecionar o algoritmo adequado a uma dada aplicação, um módulo de controle capaz, entre outras funções, de ativar os programas de cálculos existentes na unidade de processamento numérico e analisar os resultados obtidos, uma unidade gráfica para exibir e facilitar ao usuário a análise do produto fornecido pelo sistema, um módulo de explanação que

fornece o caminho usado para se chegar à solução, e uma unidade de aprendizado automático, com o objetivo de refinar o sistema, alterando as heurísticas formalizadas na base de conhecimentos.

4.3 Arquitetura do SIPREX

Os sistemas acoplados foram concebidos com a finalidade de facilitar, tornar eficiente e ampliar o domínio de resolução de problemas. Para tanto, necessita de uma arquitetura que possibilite uma perfeita harmonia entre o usuário e entre as partes que o compõem, ou seja, as estruturas de computação simbólica, numérica e de interligação. Para isso, procurou-se dar ao SIPREX uma arquitetura que atendesse aos requisitos dos sistemas acoplados, de tal forma que essa arquitetura possa ser adotada por outros sistemas acoplados semelhantes.

Assim, o SIPREX é constituído de um módulo de controle, de um módulo de interface, de um módulo central que utiliza o sistema de quadro-negro (*blackboard*), composto de uma unidade de memória de trabalho, uma base de conhecimentos e um motor de inferência, um módulo de explanação, um módulo de aprendizado automático, um módulo numérico e um módulo gráfico. A figura 4.1 mostra a estrutura do SIPREX com as ligações entre seus componentes.

Os diversos módulos que compõem o SIPREX serão descritos nos capítulos seguintes.

4.4 Expectativa de Uso

Espera-se que o SIPREX seja utilizado, além de seu papel como ferramenta de CAD, como um modelo na construção de sistemas especialistas acoplados, principalmente para sistemas na área de processamento digital de sinais e controle automático de processos, vindo a constituir-se num marco para a investigação desse processo e trazendo uma significativa contribuição ao avanço dessa tecnologia.

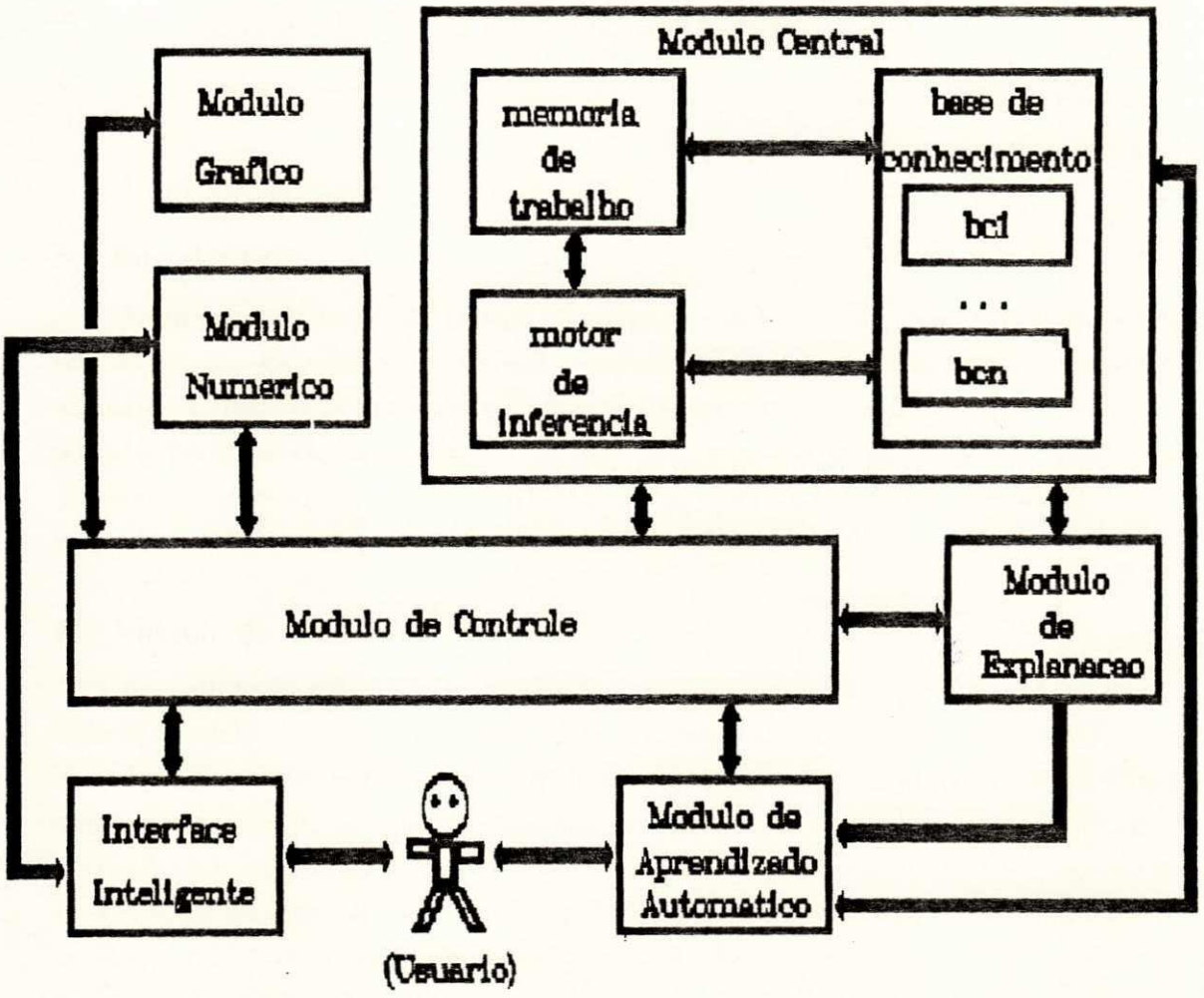

Figura 4.1: Estrutura do SIPREX

CAPÍTULO 5

ESTRUTURA DE INTERLIGAÇÃO

5.1 Introdução

Num sistema acoplado existem duas partes distintas que são responsáveis pelo funcionamento do sistema, uma pela computação simbólica e outra pela computação numérica. Estas partes precisam ser ligadas e trocarem informações. Para isso existe uma terceira parte, a estrutura de interligação que também promove o intercâmbio de informações com o usuário. No SIPREX ela é composta dos módulos de controle e interface inteligente que serão descritos nas seções seguintes.

5.2 Módulo de Controle

O módulo de controle do SIPREX é o responsável pela ativação ordenada dos demais módulos, pela análise dos resultados intermediários e pela apresentação ao usuário dos resultados finais (projeto do filtro) obtidos. É também o responsável pelo acoplamento entre as partes simbólica e numérica do sistema. As informações necessárias à computação simbólica obtidas durante o processo de voluntariamento (isto é, o usuário informa apenas o que lhe é solicitado), realizado pela interface junto ao usuário, são transferidas ao módulo central. Este por sua vez, processa essas informações obtendo um diagnóstico. O módulo de controle captura esse diagnóstico, e, após ter preparado o arquivo [ESPECE.DAT] com os dados adequados ao programa de projeto, liga-se ao módulo numérico com a ativação do programa sugerido pelo módulo central. No final dos cálculos numéricos, recupera as informações geradas transmitindo o projeto obtido ao usuário. No caso de filtros IIR, um primeiro programa de projeto é acionado, para em seguida ser ativado um outro programa com a finalidade de otimizar o projeto primitivo. É então efetuada a análise do projeto obtido. Desta análise, resulta a decisão de se ativar ou não um outro programa de otimização. A análise pode ser feita de maneira automática ou com a interferência do usuário. Se o usuário desejar, o módulo de controle pode fornecer informações sobre as razões que levaram o sistema a decidir-se por um determinado algoritmo, listando os fatos e as regras utilizadas no processo de dedução, pela ativação do módulo de explanação. Outra função do módulo de controle é a ativação do módulo de aprendizado automático, visando refinar a solução obtida. Isso é feito em duas etapas. A primeira tenta encontrar uma outra solução. A segunda, cria uma nova regra que possibilite chegar-se a essa solução. A nova regra é então armazenada na base de conhecimentos. O acesso à

segunda etapa do módulo de aprendizado somente é permitida a especialistas. O controle é feito pelo uso de uma senha de acesso.

A figura 5.1 esquematiza a seqüência de controle. Maiores detalhes sobre esta seqüência, como as etapas e operações realizadas, serão apresentadas na próxima seção.

Mesmo durante a ativação, e por conseguinte durante a transferência do controle a outros módulos, o módulo de controle permanece residente na memória transmitindo ao usuário informações, como a data, a hora, as operações realizadas e em andamento.

5.2.1 Seqüência de Controle

Em primeiro lugar o controle ativa o módulo de interface, responsável pela coleta de informações relativas ao problema em questão, através de um processo de voluntariamento junto ao usuário. Concluída essa etapa, ativa-se o módulo central, onde o processamento simbólico de informações ocorre, através de um mecanismo de inferência. Após concluído o processo de inferência com a determinação de um diagnóstico, o módulo de controle captura esta informação pela leitura do arquivo [BASE.OUT], fornecido pelo módulo central. De acordo com o diagnóstico e com as especificações do problema recebidas do módulo de interface (via arquivo [INTERFACE.TMP]), é então preparado o arquivo de dados [ESPECE.DAT] que será utilizado pelos programas contidos no módulo numérico. Em seguida transfere o controle ao módulo numérico com a ativação do programa de projeto recomendado pelo diagnóstico obtido do módulo central.

Para a determinação do projeto final usa diferentes procedimentos em se tratando de filtros FIR e IIR. Se o filtro projetado for do tipo FIR, os resultados (resposta ao impulso do filtro) são apresentados ao usuário. Uma visualização gráfica do projeto é exibida, com a ativação do módulo gráfico. No caso do filtro IIR, é necessário otimizar o projeto obtido. Para isso, ativa-se novamente o módulo numérico, desta vez através do primeiro programa de otimização. O projeto otimizado é analisado com o auxílio de outro programa do módulo numérico, o ANALIS. Antes disso, é oferecida ao usuário a opção de fazer sua própria análise. Neste caso, o programa ANALIS é usado apenas para calcular os desvios em cada faixa. Esses resultados, bem como a resposta em freqüência (magnitude, fase e atraso de grupo) do projeto produzido, sob a forma gráfica, são apresentados ao usuário, para auxiliar a análise do projeto. Se a análise for automática, isto é, realizada exclusivamente pelo sistema, o programa ANALIS completa a análise e indica a necessidade ou não de uma outra tentativa de otimização do projeto. Em caso afirmativo, o módulo numérico é solicitado a ativar o segundo programa de otimização. Uma análise comparativa entre o projeto anterior e o otimizado é então processada, e o melhor projeto, aquele que mais se aproximar das especificações da magnitude, será considerado o projeto final. Mais uma vez, o usuário pode tomar essa decisão se assim o desejar.

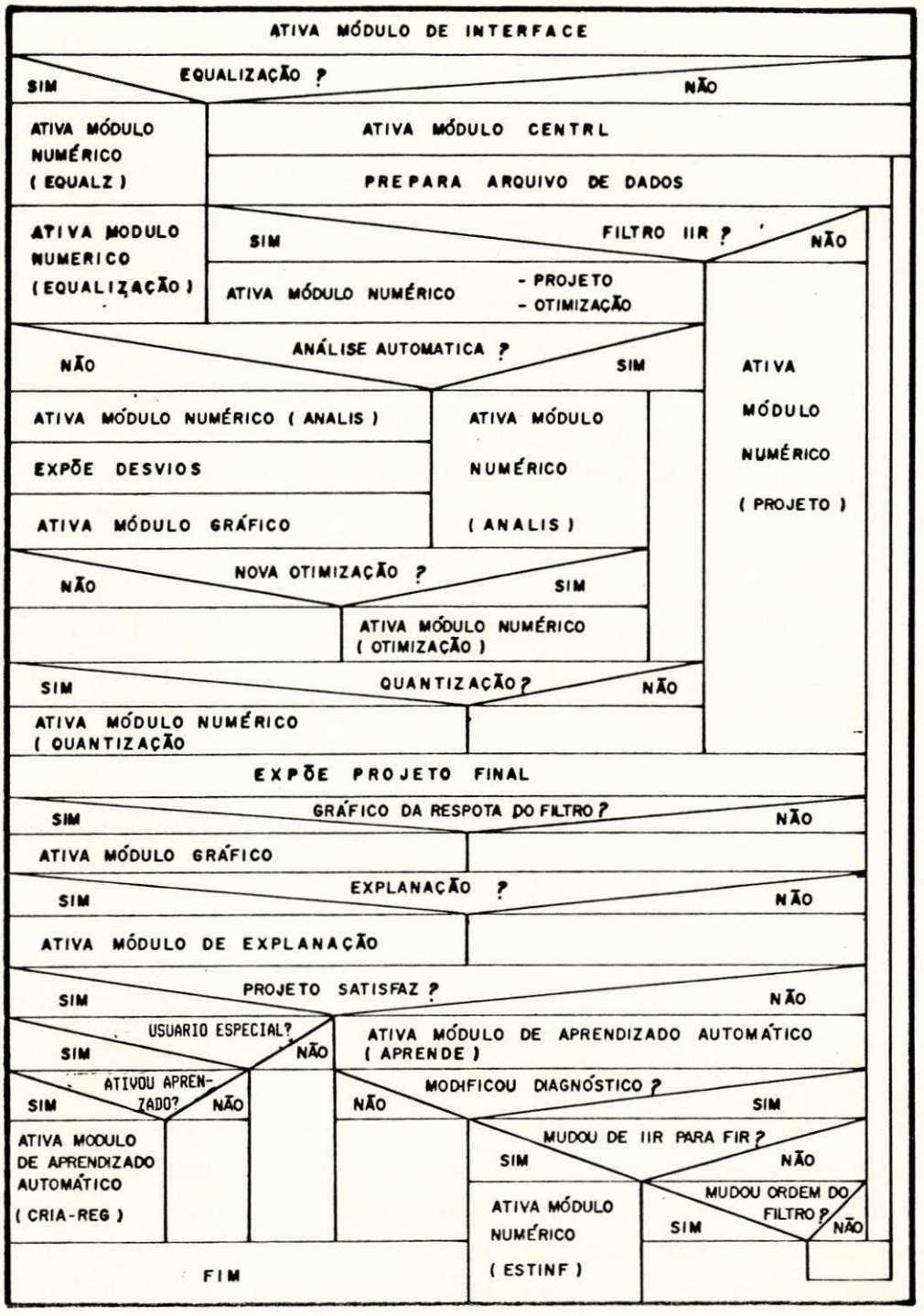

Figura 5.1: Diagrama de Nassi-Schneiderman da Seqüência de Controle

Após a obtenção do projeto final pelo sistema, caso o usuário deseje um projeto quantizado, opção feita durante o voluntariamento, o módulo de controle novamente

aciona o módulo numérico, ativando o programa de quantização. O projeto final é então apresentado ao usuário sob forma de coeficientes da realização em cascata de blocos de 2ª ordem do filtro projetado, para o caso de filtros IIR, e da resposta ao impulso, se o filtro for FIR. O módulo gráfico é ativado em seguida, permitindo ao usuário uma melhor visualização do projeto produzido.

O projeto final é então criticado pelo usuário. Caso não satisfaça as especificações fornecidas, o módulo de controle aciona o módulo de aprendizado que tentará obter um outro caminho, possibilitando ao sistema produzir uma nova solução. Aos usuários especiais (especialistas com conhecimento da senha de acesso) é permitido o acesso à segunda etapa do módulo de aprendizado automático, que em caso de sucesso da nova solução, insere o caminho que levou a ela na base de conhecimentos.

Se a tarefa pretendida for equalização de fase de um projeto de filtro recursivo já existente, o módulo de controle recebe esta informação diretamente do módulo de interface juntamente com as especificações do filtro. O módulo central não é ativado para este caso, pois trata-se de um processo puramente determinístico: o módulo de controle aciona o módulo numérico inicialmente pela ativação do programa auxiliar EQUALZ, cuja função é de calcular os polos e zeros do filtro dado, preparando as informações de maneira aceitável com o padrão exigido pelo programa que realizará a equalização, que será ativado em seguida. Após concluída a etapa de equalização o processo segue o caminho normal com a análise, possível otimização e quantização dos coeficientes, se for o caso.

5.3 Interface Homem/Máquina Dotada de Inteligência

Ativada pelo módulo de controle, a unidade de interface inteligente é a responsável pelo voluntariamento do usuário, extraindo deste, de maneira amigável, informações relativas às características e aplicabilidade do filtro pretendido. Com base nessas informações, as especificações do filtro são definidas, permitindo que o sistema forneça o projeto de um filtro ótimo, apropriado a sua (do usuário) aplicação. É dita inteligente, pois de acordo com as respostas do usuário, ela o induz a um caminho conveniente, e também, possui a capacidade de dedução, simplificando o trabalho do usuário, além de diminuir a possibilidade de erro. Após o voluntariamento, é preparado o arquivo de dados [BASE.DAT] com os fatos relevantes (necessários) de maneira adequada à utilizada pelo processamento simbólico (unidade central) do sistema. Além disso, esta interface transfere as informações relativas às especificações do filtro ao módulo central. Tais informações darão origem ao arquivo de especificações [ESPECE.DAT], utilizado pelos programas de projeto contidos no módulo numérico.

A unidade de interface inteligente comunica-se de modo direto com o módulo numérico ativando o programa auxiliar ESTINF para a estimativa do tamanho de um filtro FIR, quando for o caso. Como o programa de estimativa é um programa bastante

rápido, o processamento em tempo real da interface não fica comprometido. Para maiores esclarecimentos sobre as questões propostas ao usuário, a interface é dotada de um mecanismo de ajuda *on line* bastando para tanto, o usuário pressionar a tecla de função F1.

5.3.1 Voluntariamento Inteligente

A sistemática de voluntariamento é feita de maneira estruturada e inteligente, procurando facilitar o trabalho do usuário. Nela está incorporado um processo de dedução, o que simplifica e objetiva mais ainda o voluntariamento. Por exemplo, se o usuário informa que o filtro pretendido é do tipo passa-faixa, baseado esclusivamente nesta informação é deduzido que o filtro referido deve possuir 3 faixas, sendo a 1ª de corte, a 2ª de passagem e a 3ª de corte. O valor desejado para a magnitude da resposta do filtro deve ser 0 na primeira faixa, 1 na segunda e 0 na terceira. A seguir abre uma janela onde solicita ao usuário os valores da freqüência de corte em cada uma das três faixas, bem como o valor do peso atribuído a cada uma delas. Os valores devem ser informados em campos específicos da janela, devidamente exibidos de acordo com o número de faixas, indicados pelo cursor pulsante.

O processo de voluntariamento inteligente foi implementado utilizando a estrutura de árvore de decisão com múltiplas ramificações. Um caminho é percorrido de acordo com os fatos (nós) que são obtidos do usuário ou por dedução a partir de informações anteriores. A figura 5.2 esquematiza a árvore de decisão utilizada na interface durante o processo de voluntariamento.

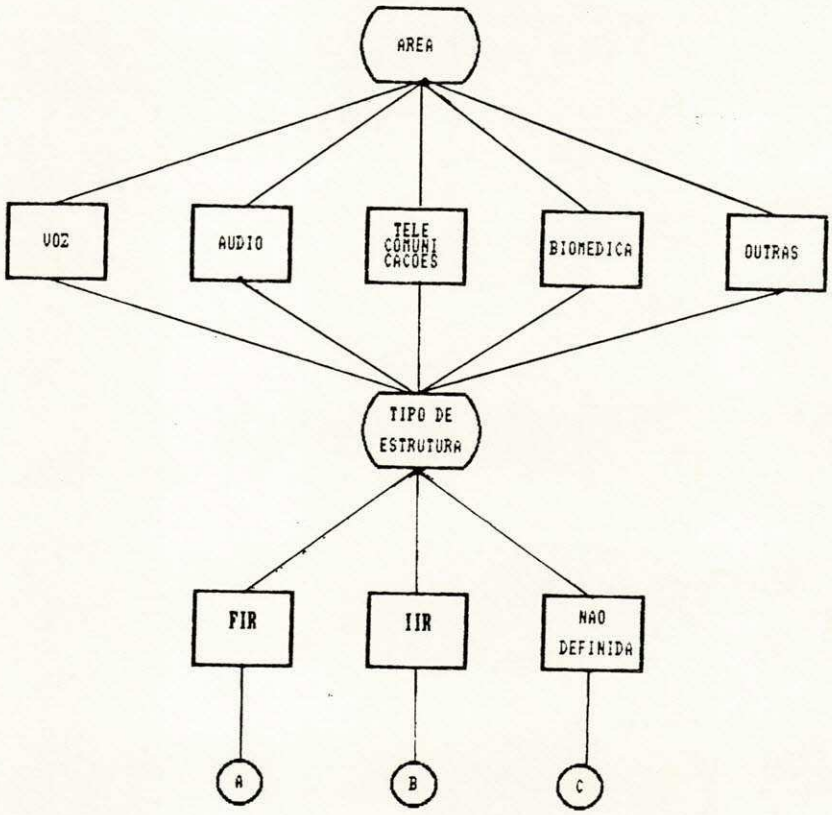

Figura 5.2: Árvore de Decisão

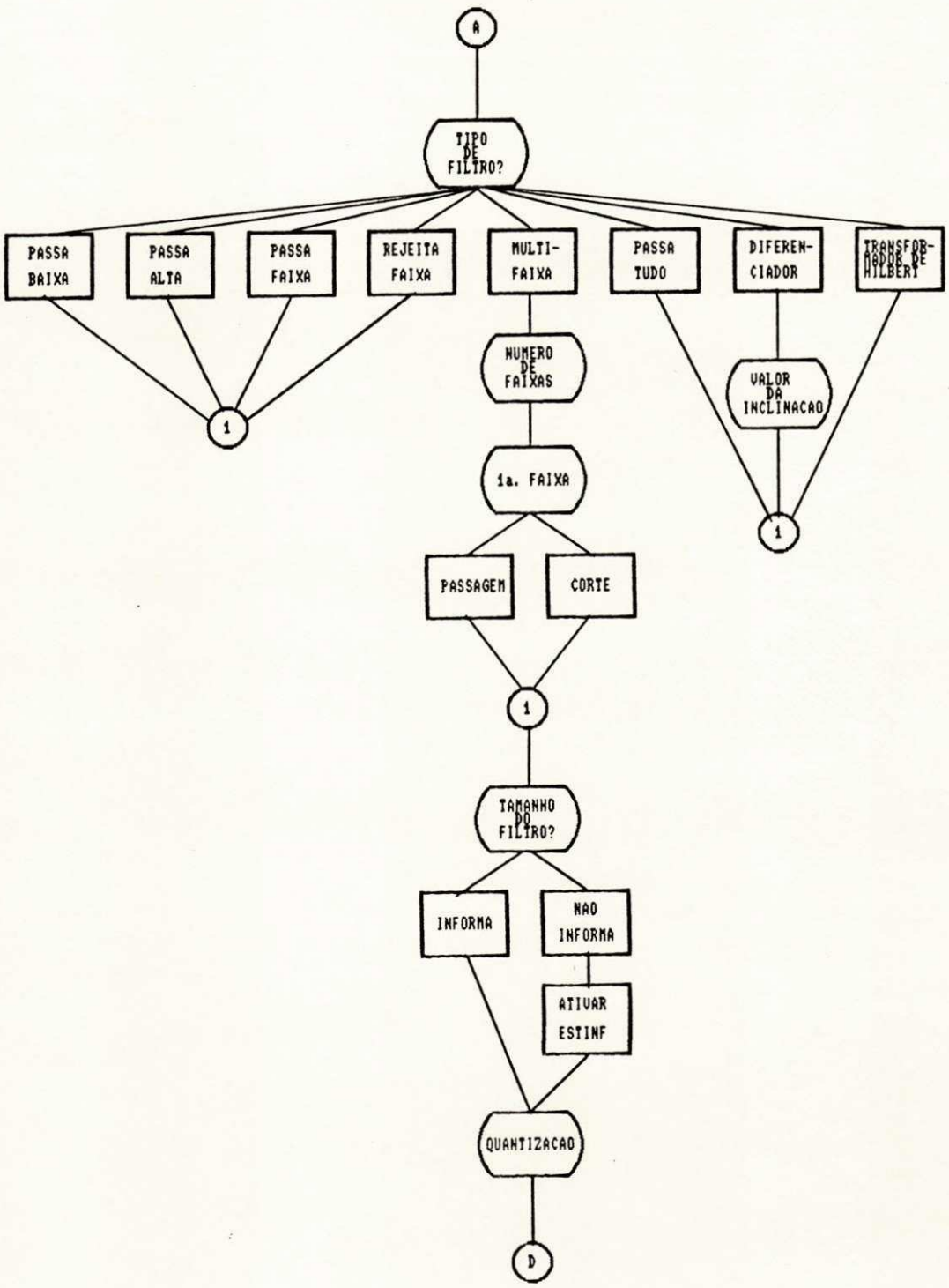

Figura 5.2: Árvore de Decisão (continuação)

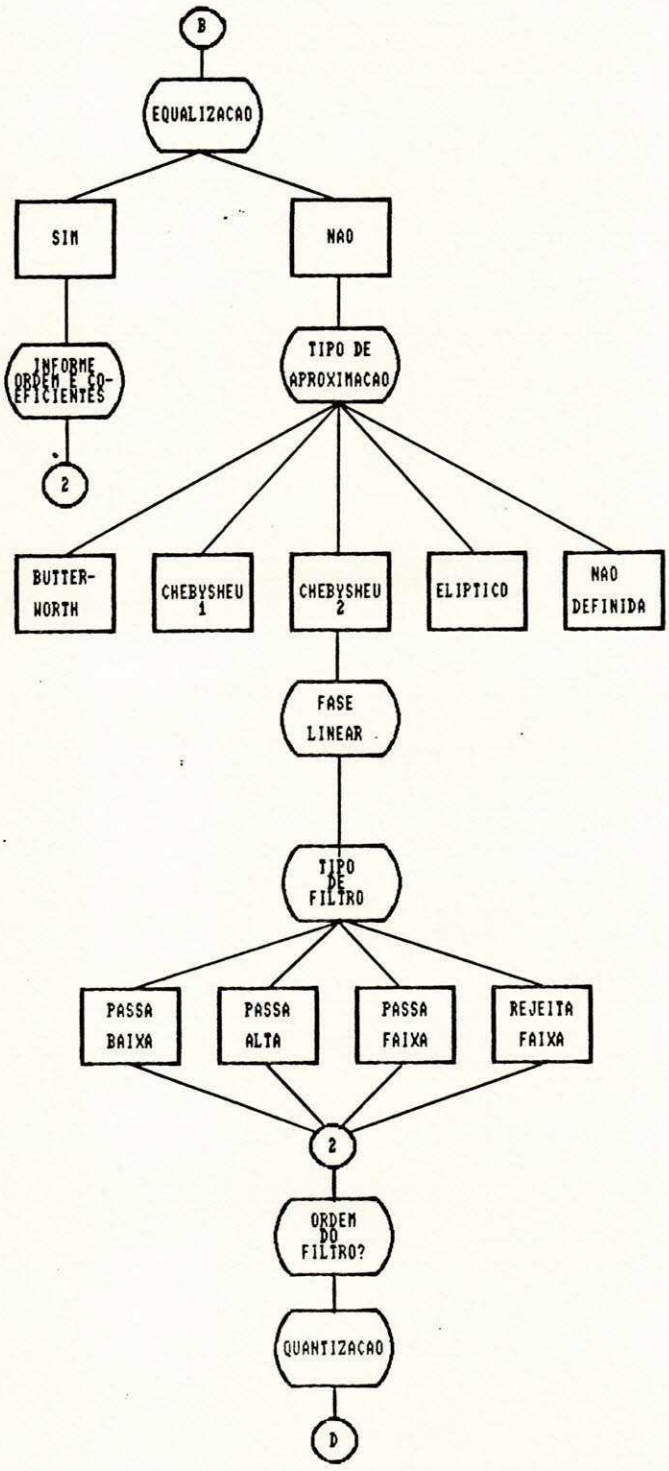

Figura 5.2: Árvore de Decisão (continuação)

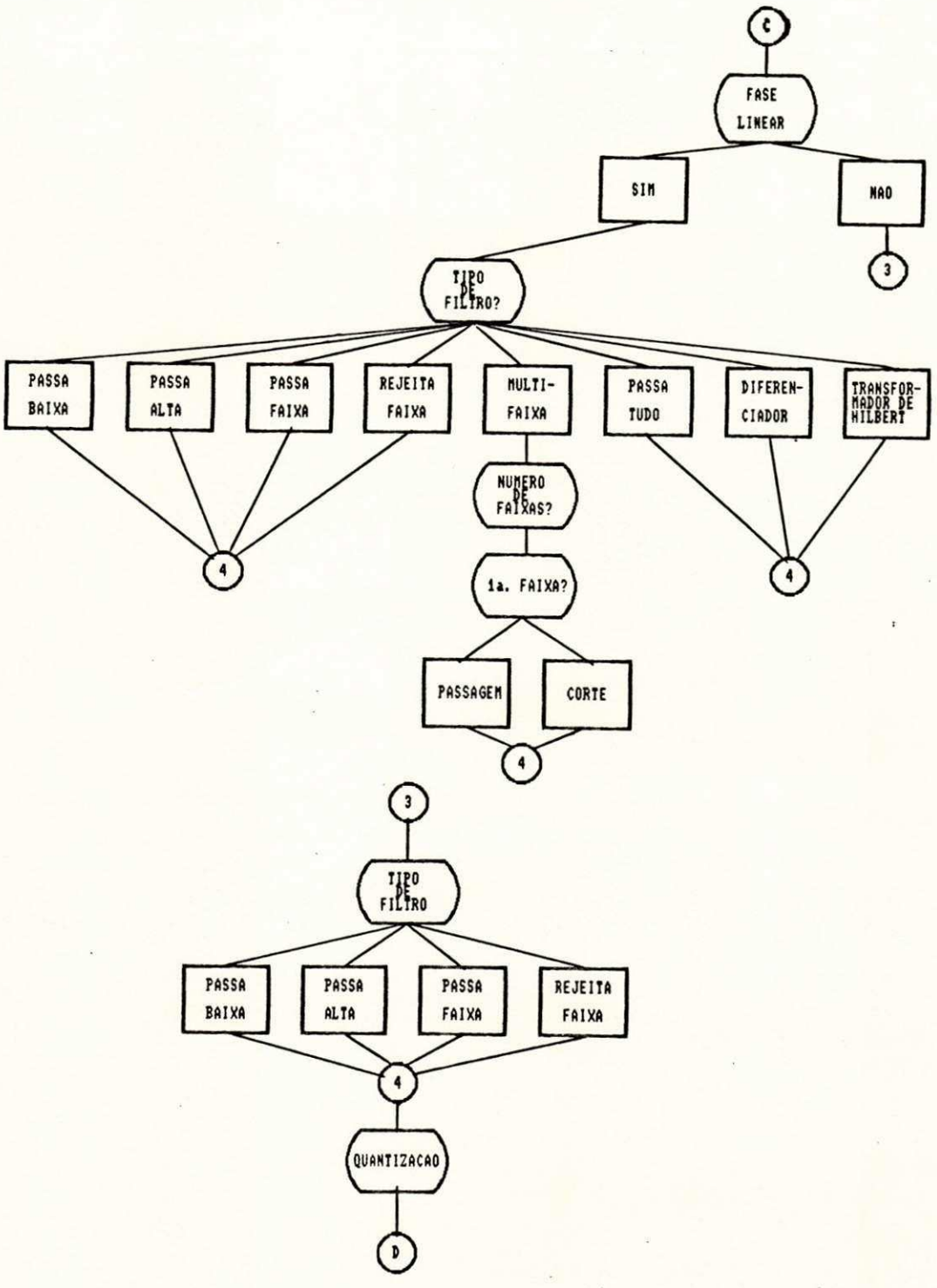

Figura 5.2: Árvore de Decisão (continuação)

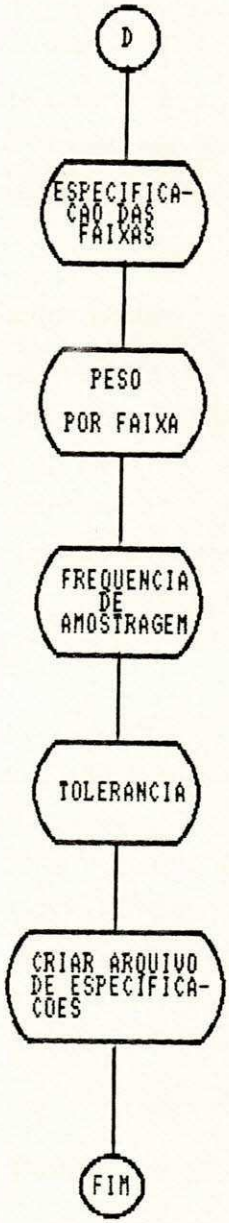

Figura 5.2: Árvore de Decisão (continuação)

Um fato quando solicitado implica na abertura de uma janela-menu, com as alternativas possíveis, das quais uma deve ser escolhida pelo usuário. Outra situação ocorre quando o usuário deve fornecer valores (as freqüências de corte, por exemplo). Novamente uma janela é aberta, com o cursor indicando o campo a ser preenchido. Valores inconsistentes ou fora dos padrões, são rejeitados pelo controle da interface após um processo de verificação de consistência.

O processo de voluntariamento somente é considerado completo após o usuário ter fornecido as informações mínimas necessárias ao projeto. Antes disso, o retorno ao módulo de controle, e por conseguinte a continuidade do processo de obtenção do projeto do filtro fica vetado.

5.3.2 Verificação de Consistência dos Dados

À medida que o usuário fornece um dado valor, atendendo a uma solicitação, um processo de verificação de consistência é desencadeado. Se um dado for rejeitado, uma mensagem de erro explicativa é apresentada no rodapé, e o dado é novamente requisitado. Os seguintes testes são processados logo na entrada dos dados:

- testa a consistência entre o tipo de caracteres fornecidos e o dado solicitado, somente permitindo a entrada de caracteres consistentes.
- testa se a ordem do filtro (N_F) está dentro da faixa de valores aceitáveis
  - se FIR, $3 \leq N_F \leq 128$
  - se IIR, $1 \leq N_F \leq 10$
- testa o número de faixas, quando o filtro for do tipo multifaixas ($N_{faixa} \leq 5$)
- testa se há superposição de faixas
- testa se o desvio normalizado informado está dentro do aceitável ($0 \leq \delta \leq 1$)
- testa se a freqüência de amostragem satisfaz o princípio de Nyquist.

É bom lembrar que o uso do processo de voluntariamento com inteligência diminui a possibilidade de erro, pois somente as questões pertinentes ao problema são colocadas e de acordo com as respostas do usuário, evitando-se com isso a introdução de dados desnecessários e contraditórios.

5.3.3 Preparação do Arquivo de Dados

De posse das informações, a interface prepara o arquivo de dados [BASE.DAT] cujo conteúdo será, numa etapa posterior, lido e transferido para a memória de trabalho pelo mecanismo de inferência do módulo central. As informações são codificadas de acordo com as convenções utilizadas pelo módulo central. A seguir será apresentado, de forma esquemática, a composição deste arquivo.

1º) Área de aplicação

1 → se Voz

2 → se Áudio

3 → se Telecomunicação

4 → se Sinais Biomédicos

5 → se Aplicações Gerais.

2º) Tipo de estrutura

0 → Não definida

1 → se FIR

2 → se IIR

3º) Tipo de aproximação

0 → se filtro FIR

1 → se IIR e Butterworth

2 → se IIR e Chebyshev I

3 → se IIR e Chebyshev II

4 → se IIR e Elíptica

5 → se IIR e Não definida.

4º) Tipo de filtro

1 → passa-baixa

2 → passa-alta

3 → passa-faixa

4 → rejeita-faixa

5 → multifaixa

6 → passa-tudo

7 → diferenciador

8 → transformador de Hilbert.

5º) Tipo de fase

1 → linear

2 → não-linear.

6º) Quantização

1 → sim

2 → não

7º) Ordem do filtro

0 → não definida (para filtros IIR)

1 → se $N_F \leq 10$

2 → se $11 \leq N_F \leq 31$

3 → se $31 \leq N_F \leq 41$

4 → se $41 < N_F \leq 51$

5 → se $51 < N_F$.

8º) Atenuação na faixa de rejeição (em dB)

1 → se $\delta_2 < 20$

2 → se $20 \leq \delta_2 < 30$

$3 \rightarrow$ se $30 \leq \delta_2 < 44$

$4 \rightarrow$ se $44 \leq \delta_2$.

9º) Largura normalizada da menor faixa de transição

$1 \rightarrow$ se $L < 0.030$

$2 \rightarrow$ se $0.030 \leq L < 0.050$

$3 \rightarrow$ se $0.050 \leq L < 0.065$

$4 \rightarrow$ se $0.065 \leq L$.

As faixas de codificação utilizadas nos registros 7, 8 e 9 do arquivo são de acordo com as informações armazenadas na base de conhecimentos.

5.4 Sumário

Neste capítulo foi descrita a estrutura de interligação do SIPREX e os módulos que a compõem, ou seja, o módulo de controle e a unidade de interface inteligente. A linguagem de programação utilizada para implementá-la foi a linguagem C, julgada a mais apropriada, resultando um programa claro (modular) e eficiente (rapidez + espaço de memória), além de diversos recursos para exibição de janelas e menus.

CAPÍTULO 6

ESTRUTURA DE COMPUTAÇÃO SIMBÓLICA

6.1 Introdução

Num sistema acoplado, a estrutura de computação simbólica é a parte inteligente propriamente dita do sistema. Nela está armazenado o conhecimento sobre os processos envolvidos na tarefa específica para a qual o sistema foi concebido. O processo de manipulação simbólica das informações tem lugar nesta parte do sistema, compreendendo os mecanismos de inferência e de aprendizado automático. A recuperação dos caminhos de decisão também faz parte desta estrutura. No SIPREX ela é constituída dos módulos central, de explanação e de aprendizado automático, que serão descritos a seguir.

6.2 Módulo Central

O módulo central do SIPREX é a parte inteligente do sistema, onde ocorre o processo de inferência, que seleciona o algoritmo apropriado e define os procedimentos para se chegar a um projeto que mais se aproxime das especificações propostas. Está organizado numa arquitetura em quadro negro (*blackboard*). Nesta arquitetura somente os fatos e as regras que serão executadas, aquelas que se aplicam à situação corrente, permanecem numa memória de trabalho, sendo portanto a mais eficiente para esta aplicação. É composto de três seções principais: base de conhecimentos, memória de trabalho e unidade de inferência. Esses componentes serão descritos a seguir. Maiores detalhes sobre o processo de aquisição de conhecimentos, modelagem da base de conhecimentos, justificativas para o uso de regras de produção, detalhes sobre o processo de decisão e sobre cada uma das unidades que o compõem, estão em Mendes [29].

6.2.1 Base de Conhecimentos

A base de conhecimentos é a parte do sistema especialista que armazena informações sobre o domínio do sistema. No SIPREX, a base de conhecimentos possui uma estrutura modular composta de células, que contêm informações referentes às áreas de processamento digital de sinais onde os filtros são empregados e sobre os algoritmos para projetos de filtros digitais presentes no módulo numérico. Cada célula possui informações sobre uma área específica de PDS e sobre os algoritmos de projeto que possam ser utilizados naquela área. O caráter modular da base de conhecimentos

permite alterar o conteúdo de uma célula, bem como incluir outras células sem afetar as demais nem a estrutura do sistema.

O procedimento adotado para a construção da base de conhecimentos pode ser resumido no diagrama apresentado na figura 6.1.

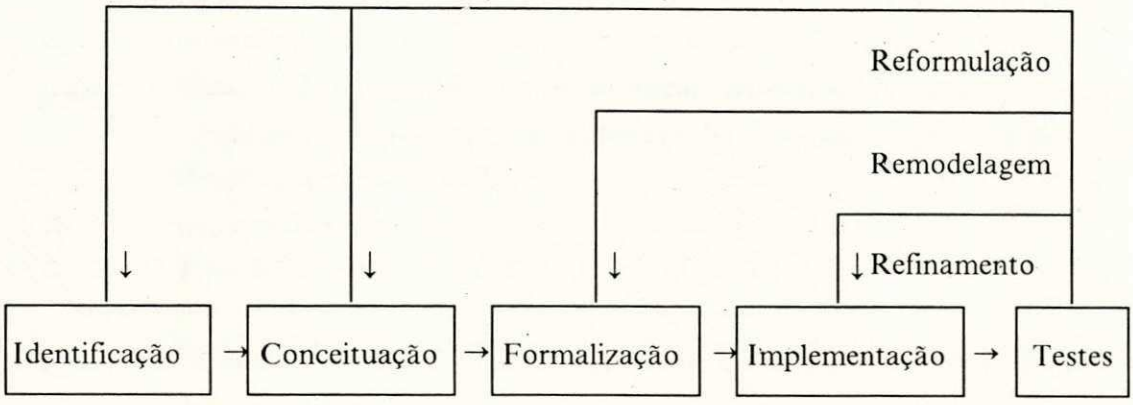

Figura 6.1: Processo de Modelagem da Base de Conhecimentos

a) Fase de Identificação e Conceituação

Identificaram-se conceitos e aspectos considerados importantes ao PDS, às áreas de aplicação de PDS e aos algoritmos para projetos de filtros digitais, pela pesquisa bibliográfica, pelas consultas a especialistas e pela análise numérica e gráfica do comportamento dos algoritmos selecionados nos diversos testes simulados que eles foram submetidos.

b) Fase de Formalização

As informações obtidas são analisadas quanto à consistência e abrangência. Aqui foi decidido o tipo de representação do conhecimento adequado à estrutura encontrada.

c) Fase de Implementação

Para representar o conhecimento, foram usadas regras de produção. As regras são agrupadas formando uma célula associada às áreas de aplicação de PDS. O conjunto de células compõe a base. A linguagem de implementação escolhida foi o PROLOG por se tratar de uma linguagem formal e precisa. As regras assumiram a forma de predicados PROLOG, com o seguinte formato:

regra (número da regra):(SE(antecedente) ENTÃO(conseqüente)).

Como já foi mencionado, a base de conhecimentos é formada por células que contêm informações sobre os algoritmos e sobre cada uma das áreas de aplicação de PDS. Internamente uma célula contém quatro arquivos, [*.reg], [*.con], [*.inf] e [*.pri], descritos a seguir.

[*.reg] - arquivo de regras, contém o conjunto de regras que compõe a célula.

[*.con] - arquivo de contagem, indica o número de fatos existentes no antecedente de uma regra e o conectivo usado entre as condições. Tem o seguinte formato:

con (C1, C2, N, C),

onde:

C1 - quantidade de fatos no antecedente da regra.

C2 - número de condições (fatos) a serem satisfeitas. A medida que uma condição for satisfeita, C2 é decrementado. Quando chegar ao valor zero, a regra é dita executável.

N - número da regra

C - Tipo de conectivo: se C = 1, conectivo OU
se C = 0, conectivo E

[*.inf] - arquivo de influência, indica os fatos que influenciam cada uma das regras. Tem o seguinte formato:

influ (fato,N),

onde:

N - número da regra.

[*.pri] - arquivo de prioridades, indica a prioridade das regras. Tem o seguinte formato:

prior (P, N),

onde:

P - prioridade da regra

N - número da regra.

O arquivo de prioridades tem a incumbência de direcionar o caminho de busca quando mais de uma opção se apresenta. É particularmente importante no aprendizado automático, quando novas informações são acrescentadas à base de conhecimentos, gerando caminhos alternativos à busca da solução.

Como exemplo, considere a seguinte regra contida no arquivo de regras de uma determinada célula,

regra(4,SE(filtro(passa-baixa),estrutura(fir))ENTÃO salve(fir1)).

O registro correspondente a esta regra no arquivo de contagem seria:

con (2, 2, 4, 0)

onde,

2 é o número de condições.

4 é o número da regra

0 é o tipo de conectivo (0 = E)

O arquivo de influência conteria registros que assinalariam os elementos de influência desta regra:

```
influ (4, filtro(passa-baixa))
influ (4, estrutura(fir)).
```

Finalmente, o registro de prioridades traria a prioridade da regra:

```
prio (4, 0)
```

onde a prioridade para esta regra é assinalada como zero.

d) Fase de Testes

Como é usual num processo de modelagem deste tipo, submeteu-se o modelo a testes de avaliação de seu desempenho. A avaliação indicou a necessidade de um maior refinamento do sistema. O refinamento se repetiu até que resultados considerados satisfatórios pelos especialistas foram obtidos.

6.2.2 Memória de Trabalho

A memória de trabalho é uma base de dados que armazena as informações do problema corrente. Numa arquitetura em quadro negro, ela desempenha um papel importante, limitando o universo de busca, reduzindo, portanto, o tempo de processamento. É formada, inicialmente, a partir de informações (fatos) fornecidos pelo usuário via interface e pela célula correspondente ao problema em questão. A unidade de inferência interage com a memória de trabalho armazenando nesta os arquivos [*.reg], [*.con], [*.inf] e [*.pri] das células correspondentes, bem como os novos fatos que são gerados durante o processo de inferência.

6.2.3 Unidade de Inferência

A unidade de inferência é o componente do sistema responsável pela busca da solução e o raciocínio envolvido no processo de inferência, executando a seleção e o casamento entre as informações contidas na memória de trabalho e na base de conhecimentos. A máquina de inferência utilizada é aquela desenvolvida por Perkusich [30], tendo sido adaptada às necessidades do sistema, com a introdução de algumas modificações para possibilitar a apreciação de novas regras que são criadas pelo módulo de aprendizado automático. É composta de cinco partes: supervisor, escalonador, acionador, diagnosticador e explanador.

O supervisor identifica a área a que o problema pertence e armazena na memória de trabalho a célula referente ao problema em questão. Em seguida ativa o escalonador.

O escalonador analisa os fatos contidos na memória de trabalho, fatos esses fornecidos pelo usuário durante o voluntariamento realizado pela interface; verifica as condições (fatos) das regras para identificar qual delas é influenciada por estes fatos. Em seguida, prepara uma lista contendo o conjunto de regras executáveis, obedecendo a ordem de prioridades contidas no arquivo [*.pri].

O acionador recebe a lista de regras executáveis e desencadeia o processo de inferência, que utiliza a estratégia de encadeamento progressivo (*forward chaining*). Parte-se das premissas das regras procurando chegar à solução, ou seja, à satisfação das condições. À medida que as regras vão sendo executadas, novos fatos podem ser gerados e adicionados à memória de trabalho, necessitando da ativação do escalonador para identificar as novas regras executáveis. Quando se chega a uma solução, ativa-se o diagnosticador que tem a função de preparar o diagnóstico de uma maneira que possa ser entendida pelo módulo de controle.

A existência de uma regra semelhante (mesma premissa), embora com prioridade menor, pode produzir uma solução alternativa. Neste caso, o diagnosticador apresenta ao módulo de controle a solução obtida pelo caminho de maior prioridade, enquanto as demais possibilidades são armazenadas num arquivo auxiliar [ALTER.OUT] para que, posteriormente, possa vir a ser utilizadas de acordo com a solicitação do módulo de aprendizado automático.

Após obter o diagnóstico, a unidade de inferência, ou mais precisamente, o supervisor, retira da memória de trabalho os fatos e as regras que foram utilizadas e transfere-os para um arquivo de dados, o EXP.OUT, que será em seguida, capturado pelo módulo de explanação.

6.3 Módulo de Explanação

Unidade presente na maioria dos sistemas especialistas, desempenha um papel fundamental na análise do raciocínio que foi empregado num determinado processo e na justificativa da solução encontrada. Permite a recuperação do caminho utilizado, listando os fatos e as regras que foram usadas para se chegar à solução.

Outro papel importante do módulo de explanação é de fornecer subsídios ao processo de aprendizado realizado pelo sistema. As informações armazenadas são passadas ao módulo de aprendizado automático que, após a análise da solução junto com o usuário, pode sugerir mudanças no caminho utilizado, alterando a base de conhecimentos, quando for o caso.

O módulo de explanação interage diretamente com o módulo central, ou mais precisamente com a unidade de inferência. Após obter o diagnóstico, o supervisor (um dos componentes da unidade de inferência) transfere da memória de trabalho para um arquivo de dados, EXP.OUT, os fatos e as regras que foram executadas durante o processo de inferência que levou à obtenção daquele diagnóstico. Essas informações são capturadas e re-arrumadas, constituindo a essência do módulo de explanação.

6.4 Módulo de Aprendizado Automático

O filtro obtido pelo algoritmo selecionado pela unidade de inferência pode não satisfazer as expectativas do usuário. Neste caso, o módulo de aprendizado automático (MAA) verifica inicialmente se existe outra possibilidade de solução, pesquisando o arquivo de soluções alternativas, [ALTER.OUT]. No caso afirmativo, a próxima solução é apresentada ao módulo de controle para a obtenção de um novo filtro, para em seguida ser novamente apresentado ao usuário. Se ainda não satisfizer as especificações dadas, e se não existir mais nenhuma possibilidade, o MAA tenta chegar a uma nova solução, a partir das especificações do projeto fornecidas inicialmente pelo usuário durante o processo de voluntariamento e de críticas à resposta em freqüência do filtro obtido, buscando encontrar uma solução satisfatória. Se teve sucesso, uma regra que possibilite esta solução será criada, desde que o usuário seja um especialista com autorização para alterar a base de conhecimentos (unidade CRIA_REGRA), de tal modo que numa situação semelhante (mesmas especificações de projeto) esta nova regra será executada pelo motor de inferência, constituindo-se num caminho alternativo. Cada vez que a execução de uma regra leva ao sucesso, a unidade CRIA_REGRA incrementa a prioridade desta regra.

O MAA é composto de duas unidades independentes: APRENDE e CRIA_REGRA. Ambas foram implementadas em linguagem PROLOG, utilizando a mesma nomenclatura comum a todos os módulos que compõem a estrutura de computação simbólica do SIPREX. A unidade APRENDE tem como objetivo refinar o processo de solução e a unidade CRIA_REGRA cria a nova regra que gera a escolha do algoritmo que possibilita a obtenção do melhor projeto. Todo usuário tem acesso a unidade APRENDE mas, apenas alguns usuários (os que tem autorização) têm acesso a unidade CRIA_REGRA, já que esta unidade altera a base de conhecimentos.

6.4.1 Unidade APRENDE

Esta unidade tem como objetivo encontrar uma nova solução, quando o projeto inicialmente produzido pelo SIPREX não satisfaz as especificações iniciais fornecidas pelo usuário durante o processo de voluntariamento.

O procedimento inicial do APRENDE é verificar se o motor de inferência encontrou mais de uma solução. Se encontrou, a próxima solução, por ordem de prioridade, é executada. No caso de não existir uma outra solução, a unidade APRENDE tentará encontrar uma nova possibilidade apropriada ao projeto do usuário.

Para encontrar essa nova solução, criou-se um conjunto de tabelas (vide a seguir) a partir da árvore de decisão utilizada para implementar a base de conhecimentos.

Tabelas de Características Relevantes dos Algoritmos

Tabelas 6.1 para valores de $1 < N \leq 11$

algoritmo	ordem	atenuação	largura
f21	ord1	att1	larg1
f26	ord1	att2	larg1
f21	ord1	att3	larg1
f21	ord1	att4	larg1

algoritmo	ordem	atenuação	largura
f26	ord1	att1	larg2
f27	ord1	att2	larg2
f27	ord1	att3	larg2
f27	ord1	att4	larg2

algoritmo	ordem	atenuação	largura
f26	ord1	att1	larg3
f26	ord1	att2	larg3
f27	ord1	att3	larg3
f27	ord1	att4	larg3

algoritmo	ordem	atenuação	largura
f10	ord1	att1	larg4
f10	ord1	att2	larg4
f27	ord1	att3	larg4
f27	ord1	att4	larg4

Tabelas 6.2 para valores de $11 < N \leq 31$

algoritmo	ordem	atenuação	largura
f10	ord2	att1	larg1
f10	ord2	att2	larg1
f10	ord2	att3	larg1
f10	ord2	att4	larg1

algoritmo	ordem	atenuação	largura
f10	ord2	att1	larg2
f10	ord2	att2	larg2
f10	ord2	att3	larg2
f10	ord2	att4	larg2

algoritmo	ordem	atenuação	largura
f10	ord2	att1	larg3
f10	ord2	att2	larg3
f10	ord2	att3	larg3
f10	ord2	att4	larg3

algoritmo	ordem	atenuação	largura
f24	ord2	att1	larg4
f24	ord2	att2	larg4
f24	ord2	att3	larg4
f25	ord2	att4	larg4

Tabelas 6.3 para valores de $31 < N \leq 41$

algoritmo	ordem	atenuação	largura
f21	ord3	att1	larg1
f10	ord3	att2	larg1
f10	ord3	att3	larg1
f10	ord3	att4	larg1

algoritmo	ordem	atenuação	largura
f10	ord3	att1	larg2
f10	ord3	att2	larg2
f10	ord3	att3	larg2
f10	ord3	att4	larg2

algoritmo	ordem	atenuação	largura
f10	ord3	att1	larg3
f10	ord3	att2	larg3
f10	ord3	att3	larg3
f10	ord3	att4	larg3

algoritmo	ordem	atenuação	largura
f10	ord3	att1	larg4
f24	ord3	att2	larg4
f24	ord3	att3	larg4
f27	ord3	att4	larg4

Tabelas 6.4 para valores de $41 < N \leq 51$

algoritmo	ordem	atenuação	largura
f21	ord4	att1	larg1
f10	ord4	att2	larg1
f10	ord4	att3	larg1
f10	ord4	att4	larg1

algoritmo	ordem	atenuação	largura
f10	ord4	att1	larg2
f10	ord4	att2	larg2
f10	ord4	att3	larg2
f10	ord4	att4	larg2

algoritmo	ordem	atenuação	largura
f26	ord4	att1	larg3
f10	ord4	att2	larg3
f10	ord4	att3	larg3
f25	ord4	att4	larg3

algoritmo	ordem	atenuação	largura
f25	ord4	att1	larg4
f25	ord4	att2	larg4
f25	ord4	att3	larg4
f25	ord4	att4	larg4

Tabelas 6.5 para valores de $51 < N \leq 128$

algoritmo	ordem	atenuação	largura
f21	ord5	att1	larg1
f10	ord5	att2	larg1
f10	ord5	att3	larg1
f10	ord5	att4	larg1

algoritmo	ordem	atenuação	largura
f10	ord5	att1	larg2
f10	ord5	att2	larg2
f10	ord5	att3	larg2
f10	ord5	att4	larg2

algoritmo	ordem	atenuação	largura
f24	ord5	att1	larg3
f24	ord5	att2	larg3
f24	ord5	att3	larg3
f23	ord5	att4	larg3

algoritmo	ordem	atenuação	largura
f10	ord5	att1	larg4
f10	ord5	att2	larg4
f10	ord5	att3	larg4
f10	ord5	att4	larg4

Os parâmetros das tabelas são:

ordem do filtro (N)

ord1 = valor de N, $1 < N \leq 11$
ord2 = valor de N, $11 < N \leq 31$
ord3 = valor de N, $31 < N \leq 41$
ord4 = valor de N, $41 < N \leq 51$
ord5 = valor de N, $51 < N \leq 128$

atenuação em DB (δ)

att1 = valor de δ, $0 < \delta < 20$
att2 = valor de δ, $20 \leq \delta < 30$
att3 = valor de δ, $30 \leq \delta < 44$
att4 = valor de δ, $44 \leq \delta$

largura da faixa de transição (L)

larg1 = valor da largura L, $0.00 < L < 0.03$

larg2 = valor da largura L, $0.03 \leq L < 0.05$

larg3 = valor da largura L, $0.05 \leq L < 0.065$

larg4 = valor da largura L, $0.065 \leq L$.

Essas tabelas indicam as características dos filtros (a ordem, a atenuação e a largura da faixa de transição) gerados por cada algoritmo. Partindo-se dessas tabelas, criou-se um conjunto de cláusulas do tipo:

tab (ordem(X), atenuação(Y), largura(Z), algoritmo)

onde,

ordem(X) é o valor da ordem do filtro,
atenuação(Y) é o valor da atenuação na faixa de rejeição,
largura é o valor da largura normalizada da faixa de transição e
algoritmo é o algoritmo que obtém o filtro ideal para os
valores da ordem, atenuação e largura da faixa de transição fornecidos.

É bom lembrar que o projeto de um filtro é determinado levando-se em conta o compromisso entre os valores da ordem do filtro, atenuação e largura da faixa de transição, portanto, este conjunto também define as características dos filtros gerados pelos algoritmos. Daí, se chega a uma solução, por uma instância ao conjunto de cláusulas que contenham os valores da ordem, atenuação e largura definidos, obtendo-se então o algoritmo que melhor satisfaça o problema.

Para se identificar qual desses valores está comprometendo o filtro produzido, são feitas consultas ao usuário a fim de determinar o que ele deseja melhorar no filtro gerado pelo algoritmo selecionado anteriormente, ou seja, se ele deseja melhorar a ordem do filtro e/ou a atenuação e/ou a largura. Então, a partir das respostas procura-se um novo algoritmo que seja capaz de melhorar o que foi determinado.

Exemplificando o processo descrito, considere o seguinte subconjunto de cláusulas:

a. tab (ordem(ord3), atenuação(att1), largura(larg3), f10)
b. tab (ordem(ord3), atenuação(att2), largura(larg3), f10)
c. tab (ordem(ord3), atenuação(att3), largura(larg3), f10)
d. tab (ordem(ord3), atenuação(att4), largura(larg3), f10)
e. tab (ordem(ord3), atenuação(att1), largura(larg4), f10)
f. tab (ordem(ord3), atenuação(att2), largura(larg4), f24)
g. tab (ordem(ord3), atenuação(att3), largura(larg4), f24)
h. tab (ordem(ord3), atenuação(att4), largura(larg4), f27)

Se os valores das especificações iniciais forem: ordem(ord3), atenuação(att3) e largura(larg4) e o usuário deseja melhorar apenas a atenuação, então aumenta-se o valor da atenuação para atenuação(att4), mantendo-se os valores da ordem e da largura.

Dessa forma, pela cláusula h encontra-se o algoritmo f27, que corresponde a um algoritmo existente no módulo numérico capaz de gerar um filtro com estas características. Em seguida, transfere-se esta informação ao módulo de controle para obter-se o novo projeto. Este processo é repetido até que se obtenha um projeto aceitável, ou não seja mais possível melhorar o projeto.

Nem sempre é possível encontrar um novo algoritmo a partir das especificações fornecidas. Portanto, antes de iniciar a consulta ao usuário, o APRENDE verifica se realmente é possível escolher um novo algoritmo para o projeto em questão. Por exemplo, se o filtro desejado for um filtro FIR para a área de voz, e for requerido a quantização dos coeficientes, então o único algoritmo existente no módulo numérico para esse projeto é o FIR4. Nesse caso o usuário será informado da impossibilidade de melhorar o projeto solicitado, sendo aconselhado a rever suas especificações.

6.4.2 Unidade CRIA_REGRA

Se o algoritmo executado foi escolhido por uma regra já existente, então a prioridade desta regra será aumentada, desde que o filtro gerado por ele tenha sido satisfatório. Se o algoritmo foi escolhido por indicação da unidade APRENDE então deve-se criar uma regra que permita ao motor de inferência noutra situação escolher esse algoritmo. Esta é a função principal da unidade CRIA_REGRA. A nova regra a ser criada é do tipo:

Se (premissas) Então (conseqüente).

As premissas da nova regra são iguais às premissas da regra que levou à solução rejeitada, e o conseqüente é o algoritmo que obteve sucesso. Chamando-se o primeiro algoritmo de D e o segundo de E, então, se a regra que selecionou o algoritmo D foi a regra:

regra 10 (se A, B, C então D).

A nova regra será:

regra X (se A, B, C então E),

onde X é o número de ordem da regra, sucessor do número da última regra na base de conhecimentos.

Essa regra é gravada numa base de conhecimentos distinta que contém apenas as regras geradas por este módulo. Assim, como a base de conhecimentos original, esta nova base também é auxiliada por três arquivos: novo arquivo de contagem, novo arquivo de influência e novo arquivo de prioridade. Nos novos arquivos, os predicados referentes a nova regra são iguais aos da regra original nos arquivos originais, com exceção do número da regra, já que as premissas da regra original são as mesmas da nova regra. A prioridade inicial da nova regra será zero.

Quando o motor de inferência seleciona uma regra, ele sempre o faz levando em conta a prioridade das regras. Se duas regras que podem ser executadas possuirem a

mesma prioridade, ele dá preferência a que tiver o menor número de ordem. Se à medida que o sistema for utilizado, a diferença entre as prioridades de duas regras semelhantes ultrapassar 95% do valor da maior prioridade entre elas, então a regra de menor prioridade será retirada da base, já que seu conseqüente não é capaz de produzir um projeto aceitável.

Espera-se que com a utilização do sistema o módulo de aprendizado automático desempenhe um papel importante de ajuste do sistema, levando a produzir um projeto cada vez mais refinado.

6.5 Sumário

Neste capítulo foi descrita a estrutura de computação simbólica do SIPREX, que é composta do módulo central, do módulo de explanação e do módulo de aprendizado automático.

CAPÍTULO 7

ESTRUTURA DE COMPUTAÇÃO NUMÉRICA

7.1 Introdução

A Estrutura de Computação Numérica constitui a parte do sistema acoplado onde são realizados os processamentos numéricos intensivos. Contém os módulos que implementa os algoritmos numéricos. No SIPREX é composta do módulo numérico, que implementam algoritmos de projeto, e o módulo gráfico, que implementa algoritmos capazes de exibir a curva de resposta em freqüência do filtro.

Numa estrutura de computação numérica a precisão e o tempo de execução são considerados fatores críticos, por isso, sua implementação exige o emprego de linguagens que preencham esses requisitos. Dessa forma, o módulo numérico, cuja preocupação maior é com a precisão dos resultados, utilizou a linguagem FORTRAN 77 para a implementação de seus algoritmos, e o módulo gráfico, em que a preocupação maior é com a eficiência de exibição de uma curva, utilizou a linguagem C.

Nas próximas seções os módulos que compõem esta estrutura serão descritos.

7.2 Módulo Numérico

É o responsável pelos cálculos numéricos do sistema. Nele estão implementados os programas para projeto de filtros digitais. Esses programas têm origem na coletânea de programas para processamento digital de sinais editada e divulgada pelo Digital Signal Processing Comitee da IEEE Acoustics, Speech and Signal Processing Society [4]. Os programas originais sofreram substanciais modificações neste trabalho, a fim de torná-los mais eficientes e robustos e também de adaptá-los ao sistema SIPREX, para adequá-los à padronização das especificações requeridas, como à execução de uma determinada tarefa.

Os programas utilizados estão divididos em duas classes: programas para síntese de filtros digitais recursivos (IIR), e para síntese de filtros não recursivos (FIR).

Há também no módulo numérico três outros programas auxiliares, um para análise da magnitude da resposta em freqüência do filtro projetado, outro para estimar, com base nas especificações do projeto, o tamanho do filtro, e um terceiro para, quando da equalização de fase, calcular os polos e zeros do projeto fornecido, bem como as suas localizações em relação ao círculo unitário.

7.2.1 Especificações do Projeto

A necessidade de se permitir ao sistema a ativação de qualquer um dos programas do módulo numérico de acordo com as especificações de um projeto, a preocupação em se definir uma maneira uniforme de especificar um projeto de um filtro digital, e também procurando minorar o esforço do usuário na definição dos diversos parâmetros exigidos pelos programas, levou a execução das seguintes etapas iniciais:

a) Definir uma forma padrão de se especificar um filtro. Tais especificações são armazenadas num arquivo de dados (ESPECE.DAT) que fornecerá os valores aos programas.

b) Introduzir modificações nos programas de modo que permita a definição dos parâmetros específicos de cada programa, necessários a sua ativação, a partir das especificações padronizadas contidas no arquivo ESPECE.DAT.

c) Os parâmetros devem usar valores assumidos (*default*) caso não sejam supridos pelo ESPECE.DAT. Tais valores seriam fixados baseados na observação dos seus comportamentos nos testes realizados com esta finalidade específica.

Com base nessas orientações, o arquivo ESPECE.DAT foi definido com a seguinte estrutura:

a) Linha1: NORD, JTIPO, NFAIXA, FREQAM, NAUX

NORD : Variável inteira, a ordem do filtro.
$1 \leq$ NORD ≤ 10, se o filtro for IIR
$3 \leq$ NORD ≤ 128, se o filtro for FIR

JTIPO : Variável inteira, o tipo de filtro.
= 1, Passa-baixa
= 2, Passa-alta
= 3, Passa-faixa
= 4, Rejeita-faixa
= 5, Multi-passa-rejeita-faixas
= 6, Passa-tudo
= 7, Diferenciador
= 8, Transformador de Hilbert

NFAIXA : Variável inteira, o número de faixas (passa-rejeita).
$1 \leq$ NFAIXA ≤ 5

FREQAM : Variável real, freqüência de amostragem.

NAUX : Variável inteira, assumindo um significado diferente, de acordo com o programa:
Prog. IIR1: tipo de aproximação a ser utilizada.
= 1, filtro Elíptico
= 2, filtro Chebyshev I (passa-faixa)
= 3, filtro Chebyshev II (rejeita-faixa)

= 4, filtro Butterworth (*maximally-flat*)

= 0, valor *default* utilizado (Elíptico)

Prog. IIR2: aproximação fornecida pelo programa.

= 1, somente magnitude

= 2, somente atraso de grupo

= 3, equalização de fase

= 4, magnitude e atraso de grupo

= 0, valor *default* utilizado (magnitude).

Programas IIR3 e IIR4: fixa coeficientes.

= 0, fixa coeficientes da variável de maior expoente

= 1, nenhum coeficiente será fixado

= 0, valor default utilizado (fixa coeficientes).

Prog. FIR2 tipo de janela a ser utilizada

= 1, retangular

= 2, triangular

= 3, Hamming

= 4, Hamming generalizada

= 5, Hanning

= 6, Kaiser

= 7, Chebyshev.

b) Linha2: FRCORT

FRCORT : Vetor real, armazena as faixas de freqüências especificadas pelas freqüências superior e inferior de corte.

c) Linha3: FREDES

FREDES : Vetor real, armazena a resposta em freqüência desejável em cada faixa.
- Em filtros diferenciadores representa a inclinação da curva da resposta em freqüência.

d) Linha4: PESO

PESO : Vetor real, armazena a ponderação de cálculo em cada faixa.

e) Linha5: DELTA1, DELTA2

DELTA1 : Variável real, desvio máximo na(s) faixa(s) de passagem(ns).

DELTA2 : Variável real, desvio máximo na(s) faixa(s) de corte(s).

f) Linha6: NBIT1, EQUALZ

NBIT1 : Variável inteira, tamanho da palavra definida para a implementação do filtro (somente utilizada pelo programa IIR4).

EQUALZ : Variável real, fator de equalização (somente utilizado pelo programa IIR2, quando da equalização de fase).

Observação: O campo não utilizado é preenchido com zero.

Cada programa que compõe o módulo numérico será descrito a seguir.

7.2.2 Programas para Projeto de Filtros Recursivos (IIR)

São quatro programas com finalidades distintas porém complementares. O programa IIR1 fornece um primeiro projeto do filtro a partir das especificações fornecidas. Desse projeto são extraídas informações, tais como natureza e localização dos polos e zeros, as quais serão utilizadas como ponto de partida para a otimização realizada pelo programa IIR2. Como resultado dessa otimização, obtém-se, em geral, um projeto mais refinado. O IIR2 pode ser usado também para equalização do atraso de grupo (*group delay*) de um dado filtro. Não existe garantia de que a solução obtida pelo programa IIR2 seja globalmente ótima, porque o critério de convergência usado é o da norma do gradiente do erro ser menor do que um certo valor (EPS), considerado pequeno. Esta é uma condição necessária mas não suficiente para garantir a otimização global nos diversos casos de não-linearidade dos problemas propostos. Por essa razão, optou-se pela inclusão de mais de um programa de otimização no sistema. Assim, havendo necessidade de um maior refinamento da solução, necessidade essa decidida pela análise do projeto efetuada pelo programa auxiliar ANALIS, o sistema tenta uma outra otimização, com a ativação do programa IIR3, que utiliza um método de otimização que toma como ponto de partida o projeto fornecido pelo IIR3. A decisão pela tentativa de uma melhor solução, com a ativação do IIR3, pode ser tomada de maneira automática pelo sistema, baseado nos valores dos desvios nas faixas de passagem e corte, entre a magnitude da resposta em freqüência do filtro projetado e o valor desejado, avaliados num número de pontos igual a 20 vezes o valor da ordem do filtro, utilizando os critérios de erro médio quadrático e erro máximo em cada faixa. Alternativamente, essa decisão pode ser tomada pelo próprio usuário independente de qualquer critério. Neste caso, o sistema informa os valores dos desvios em cada faixa e o gráfico da resposta em freqüência (magnitude e fase) do projeto em questão, para dar subsídios à sua decisão.

Não necessariamente o projeto fornecido pelo IIR3 é melhor que o fornecido pelo IIR2. O que se tem observado é que, em geral, quando o projeto fornecido está bastante fora das especificações iniciais, o IIR3 consegue melhorá-lo. Uma vez ativado o IIR3, o sistema processa uma ánalise comparativa entre o projeto anterior e o novo projeto obtido. O projeto considerado melhor pelo critério de compração dos desvios em cada um dos projetos é considerado como projeto final. Novamente o usuário pode optar e decidir ele próprio pela escolha do projeto final. Se o usuário desejar um projeto quantizado, isto é, adequado a uma implementação com tamanho de palavra finito, o sistema aciona o programa IIR4 que a partir do projeto obtido com coeficientes de alta

precisão, fornece um novo projeto com base nas restrições do tamanho de palavra para representar os coeficientes do filtro.

O projeto final do filtro é fornecido pelos coeficientes da função de transferência

$$H(z^{-1}, x) = k \prod_{i=1}^{N} \frac{A2_i z^{-2} + A1_i z^{-1} + 1}{B2_i z^{-2} + B1_i z^{-1} + 1}$$

na seguinte seqüência :

$$N, K,$$

$$x = \{A2_1, A1_1, B2_1, B1_1, A2_2, A1_2, B2_2, B1_2, \ldots, A2_n, A1_n, B2_n, B1_n\}$$

N, a ordem do filtro é entendida como o número de seções de 2ª ordem da realização em cascata do filtro ($1 \leq N \leq 10$). Se o usuário não definir a ordem desejada o sistema o fará automaticamente buscando a menor ordem possível na obtenção de um projeto capaz de satisfazer as especificações exigidas. O mesmo ocorre quando não se consegue satisfazer essas especificações com a ordem dada.

7.2.2.1 Programa IIR1

Baseado no programa DOREDI de Dehner [4], tem a função de resolver o problema clássico de aproximação para projetos de filtros digitais recursivos dos tipos passa-baixa, passa-alta, passa-faixa, e rejeita-faixa. Do programa original, apenas a parte relativa a síntese de filtros foi utilizada, mesmo assim após profundas modificações para adequá-lo ao sistema.

O método usado produz inicialmente um projeto analógico que aproxima o esquema de tolerâncias obtido das especificações do usuário, em seguida por um mapeamento para o plano Z pela transformação bilinear, obtém o filtro digital desejado. As aproximações possíveis são: a magnitude da resposta em freqüência mais plana (*maximally flat*) nas faixas de passagem e corte (Butterworth), nas faixas de passagem ou corte *equiripple* (Chebyshev I e II), e aproximação elíptica com *equiripple* nas faixas de passagem e corte (Cauer) [31,32].

Além dos coeficientes do filtro, o programa fornece o número de polos e zeros da função de transferência e suas localizações em relação ao círculo unitário no plano Z, assumindo a realização como uma cascata de seções de 2ª ordem. Essas informações são utilizadas pelo programa IIR2.

7.2.2.2 Programa IIR2

Baseado no programa LPIIR de Deczky [4,33], produz um filtro digital estável com resposta arbitrária em freqüência, utilizando o critério da norma p mínima do erro e o

método de otimização de Fletcher e Powell [34]. O problema inicial é tratado como um problema clássico de aproximação. Usando o critério da norma p mínima do erro, gera-se um problema de minimização de uma função de n variáveis, que é resolvido iterativamente com aplicação do método de Fletcher e Powell, com modificações para garantir a estabilidade. O erro quadrático ($p=2$) é o assumido (*default*). A função do sistema do filtro é representada por seus polos e zeros. O erro pode ser minimizado em termos da magnitude da resposta em freqüência, em termos da aproximação do atraso de grupo, ou em ambos simultaneamente.

Como todo problema de otimização, o programa necessita de uma aproximação (projeto de filtro) inicial. No caso da equalização de fase de um filtro já existente, essa aproximação (ou projeto inicial) é fornecida pelo usuário, sendo necessário um pré-processamento pelo programa auxiliar EQUALZ, que extrai as informações necessárias do projeto fornecido, com respeito a polos e zeros. Essa aproximação pode ser também o próprio projeto obtido do programa IIR1, que já fornece os dados de uma maneira que permite o uso direto do programa IIR2.

O programa original sofreu várias modificações procurando adequá-lo ao sistema, torná-lo mais robusto e mais eficiente. Além da forma padronizada imposta pelo sistema, muitas informações que eram exigidas do usuário, passaram a ser deduzidas das próprias especificações do projeto, ou arbitrados com os valores assumidos, obtidos de maneira empírica. Salvaguardas contra *overflow* e *underflow* foram introduzidas em trechos críticos detetados. A rotina que calcula e avalia a função de minimização teve que ser implementada em precisão dupla para contornar problemas de precisão surgidos em alguns casos testados. Trechos do programa tiveram que ser reescritos para permitir a sua execução que era comprovadamente impossível na sua forma original.

Restrições foram impostas na abrangência do programa devido a falta de uma metodologia (mesmo empírica) que permitisse estimar a localização de polos e zeros de uma maneira que pudesse garantir a convergência para filtros não convencionais. Quanto à equalização de fase só foi permitida a filtros passa-baixa, pois para outros tipos de filtros, não se obteve um projeto satisfatório.

O programa original exigia que o usuário fornecesse duas rotinas para especificar a magnitude da resposta em freqüência desejada e para especificar o atraso de grupo sempre que desejasse utilizá-lo para um projeto específico. Essa exigência foi suprimida e as referidas rotinas passaram a ser geradas automaticamente a partir das especificações dadas.

7.2.2.3 Programa IIR3

Baseado no programa OPTIIR de Dolan-Kaiser [4] utiliza o método de otimização da função de penalidade. Dados os coeficientes da realização em cascata das seções de 2ª ordem da função de transferência de um filtro digital, o programa varia os coeficientes

até que as especificações da magnitude da função no domínio da freqüência sejam atendidas. Um critério de otimização minimax é utilizado, ou seja, se um filtro for aceitável, então maximize o mínimo dos valores pelos quais ele excede as especificações; se não for aceitável, então minimize o máximo dos valores em que ele falha. Esta avaliação é feita em diversos pontos do espectro. O programa original requeria do usuário o fornecimento desses pontos de freqüência, os valores correspondentes às respostas mínima e máxima aceitas para cada um deles, bem como os pesos relativos a estes pontos. Essa maneira de especificar o problema exigia uma grande quantidade de dados a ser passada ao programa. Para se ter uma idéia, para um filtro de 3ª ordem, seria necessário o fornecimento de 75 valores só para especificar a magnitude da resposta desejada. Esse problema foi sanado com uma alteração no programa original, passando os pontos a serem calculados de maneira criteriosa pelo próprio programa, sendo o número deles proporcional à ordem do filtro, e com maior densidade nas regiões consideradas críticas, como em torno das freqüências de corte.

O projeto inicial a ser otimizado é fornecido pelo programa IIR2.

7.2.2.4 Programa IIR4

Baseado no programa FWIIR de Steiglitz-Ladendorf [4], a partir de um projeto otimizado de um filtro digital com coeficientes de alta precisão, produz um filtro com coeficientes quantizados, com um tamanho finito de palavra definido pelo usuário, procurando adequá-lo às especificações em magnitude da resposta em freqüência.

O ponto de partida para a otimização é obtido arredondando-se os coeficientes do projeto inicial, para um número definido de bits. Uma versão randomizada do algoritmo de busca de Hooke-Jeeves [35] é então utilizada, para otimizar os coeficientes para o tamanho de palavra definido.

Com a otimização alguns polos ou zeros podem ser deslocados para fora do círculo unitário no plano complexo. Para garantir a estabilidade do filtro, após a convergência, os polos e zeros que se encontram fora do círculo unitário são invertidos.

A avaliação da função de otimização é feita em pontos pré-determinados. No programa original, o usuário deveria fornecer estes pontos, o que acarretaria um grande esforço devido a quantidade de informações a ser fornecida, pois além do valor da freqüência, deveria ser fornecido o valor da resposta em freqüência e a tolerância para cada ponto. Mais uma vez isso foi solucionado no SIPREX, com os pontos passando a ser determinados pelo próprio programa de uma maneira racional, com maior densidade em torno de regiões consideradas críticas, como foi feito nos programas anteriores. As opções do programa de fixar os primeiros coeficientes das seções de 2ª ordem, o número de bits para o incremento inicial de pesquisa de otimização dos coeficientes tomado igual ao número de bits do comprimento da palavra determinada menos um, o ponto de partida do algoritmo de randomização, a utilização da segunda fase que inverte os polos

e zeros fora do círculo unitário, foram fixados pelo programa e são os valores assumidos pelo sistema. Essa decisão foi tomada após um exaustivo processo de acompanhamento do desempenho do algoritmo em cada situação em uma bateria de testes. A alteração de qualquer um desses parâmetros pode ser feita pelo sistema.

7.2.3 Programas para Projeto de Filtros Não-Recursivos (FIR)

Quatro programas foram incluídos no módulo numérico do SIPREX com a capacidade de projetar filtros digitais não-recursivos com fase linear. Operam independentemente, mas produzem projetos, de alguma forma, diferenciados, o que justifica a diversificação de algoritmos. O processo de projeto de um filtro é sempre um compromisso entre o tamanho do filtro, a largura da faixa de transição e as tolerâncias nas faixas de passagem e corte. A decisão por um dos programas é feita de acordo com as especificações do filtro desejado e o comportamento dos programas levando em conta esses parâmetros. A necessidade de quantização dos coeficientes do filtro faz com que o sistema opte pela ativação do programa FIR4, que é o único entre eles capaz de fornecer um projeto quantizado.

Os programas selecionados utilizam as técnicas mais comuns a projetos de filtros FIR com fase linear. O programa FIR1 utiliza a aproximação ótima de Chebyshev, usando o algoritmo de permutação de Remez [36]. O programa FIR2 implementa o método clássico de janelas. São sete tipos de janela: retangular, triangular, de Hamming, de Hanning, generalizada de Hamming, de Kaiser, e a de Chebyshev. O sistema escolhe uma dentre as sete janelas, levando em conta as especificações do filtro desejado. O programa FIR3 implementa o critério de resposta em freqüência mais plana possível (*maximally-flat*) para filtros passa-baixa de fase linear. O quarto programa, o FIR4, implementa um método para síntese de um filtro com tamanho de palavra finito, isto é, os coeficientes são representados com um número fixo de bits. O programa determina o tamanho mínimo de palavra capaz de satisfazer as especificações do filtro proposto.

O projeto produzido por qualquer um dos programas é fornecido ao sistema sob a forma de resposta do filtro ao impulso, sendo armazenado no arquivo [FIR*.OUT], no seguinte formato:

```
NFILT, NS
H(1), H(2),..., H(n)
FN, F1, FREQAM
```

onde,

NFILT - tamanho do filtro

NS - indicador de simetria.
= 1, filtro simétrico
= -1, filtro anti-simétrico

H(i) - resposta ao impulso do filtro projetado

FN, F1 - a maior e a menor freqüência do intervalo amostrado

FREQAM - freqüência de amostragem.

7.2.3.1 Programa FIR1

Baseado no programa EQFIR de McClellan, Parks e Rabiner [4] produz uma aproximação ótima de Chebyshev utilizando o algoritmo de permutação de Remez [37], implementado de maneira eficiente, tirando vantagens de sua estrutura. Com isso se consegue projetar filtros maiores em tempos computacionalmente razoáveis. A estrutura especial de implementação do algoritmo de Remez leva em conta o fato da função a ser aproximada ser constante nas faixas de passagem (= 1) e corte (= 0) [8]. A aproximação é ótima pelo critério minimax onde nas faixas de interesse, a resposta em freqüência obtida é a que minimiza o erro máximo ponderado. A função de ponderação do erro é constante e diferente em cada faixa.

O programa pode ser usado para projetar todos os tipos de filtros digitais clássicos seletivos em freqüência, mais os filtros passa-tudo, diferenciador, e transformador de Hilbert. Para os três últimos, os testes mostraram que o programa deve ser restrito a filtros de ordem par.

O número de faixas para os filtros multifaixas está restrito a 5, por imposição do programa original.

7.2.3.2 Programa FIR2

Baseado no programa FWFIR de Rabiner e McGonegal [4], implementa a técnica popular de janelas para projetar filtros digitais lineares não recursivos dos tipos passa-baixa, passa-alta, passa-faixa, e rejeita-faixa. As janelas implementadas são do tipo retangular, triangular, de Hamming, de Hanning, generalizada de Hamming, Chebyshev e Kaiser. A decisão pelo uso de uma delas é feita pelo sistema, a partir de informações contidas em sua base de conhecimentos. Essas informações são oriundas de observações de projetos fornecidos por cada uma das janelas, nos diversos testes a que o programa foi submetido, tendo sempre como base o compromisso entre o tamanho do filtro, a largura da faixa de transição e do desvio nas faixas de passagem e corte. Vale a pena salientar que o método de janelas é uma implementação de fórmulas fechadas e sem necessidade de otimização, o que pode ser bastante vantajoso em alguns casos.

A janela generalizada de Hamming necessita da definição de uma constante α. O valor assumido é $\alpha = 0.54$, obtido experimentalmente com testes.

A janela de Chebyshev necessita da definição de dois dentre os três parâmetros: tamanho do filtro (NF), largura normalizada da faixa de transição (ΔL), e tolerância (δ_p em dB). Optou-se por conveniência, pela fixação do tamanho do filtro e da tolerância. A largura de transição foi estimada pela fórmula de Chebyshev

$$\Delta L = \frac{1}{\pi}\cos^{-1}\left[\cosh\left(\frac{\cosh^{-1}\left(\frac{1+\delta_p}{\delta_p}\right)}{NF-1}\right)\right]^{-1}.$$

7.2.3.3 Programa FIR3

Baseado no programa MXFLAT de Kaiser [4], fornece os coeficientes de um filtro mais plano possível nas faixas de passagem e corte, com um número ímpar de termos. O tamanho mínimo do filtro que satisfaz as especificações é estimado em função da largura normalizada e do valor da resposta correspondente à freqüência central da faixa de transição. A faixa de transição é definida como sendo a região onde a magnitude da resposta ao impulso do filtro cai de 95% do valor desejado para a faixa de passagem, até um valor de 5%. A largura da faixa de transição é obtida das especificações fornecidas, enquanto a freqüência central ótima é estimada pelo programa pelo emprego do método de tentativas e bisseção. Os coeficientes são determinados pela amostragem uniforme da resposta em freqüência, seguida da execução da inversa da transformada discreta de Fourier do conjunto de amostras obtido.

A ordem do filtro é inversamente proporcional ao quadrado da largura da faixa de transição.

O programa conseguiu resultados aceitáveis a partir de uma largura de transição normalizada de 0.1.

7.2.3.4 Programa FIR4

Baseado no programa FWFIR de Heute [1], produz um filtro FIR de fase linear com os coeficientes quantizados, isto é, representados em uma palavra com um número finito e determinado de bits. Pode ser usado nos projetos de filtros passa-baixa, passa-alta, passa-faixa simétrico, rejeita-faixa simétrico, e transformadores de Hilbert. O programa transforma inicialmente as especificações de qualquer um destes filtros num protótipo passa-baixa. A seguir, determina o tamanho mínimo de palavra capaz de satisfazer as especificações transformadas em função do desvio tolerado nas faixas de passagem e corte, e da largura da faixa de transição. O tamanho do filtro é então estimado com base no procedimento de Herrmann et al. [37], para em seguida utilizar o algoritmo de Parks-McClellan [38] a fim de obter o projeto passa-baixa. O programa trabalha iterativamente verificando a cada passo uma possível violação das tolerâncias especificadas, fazendo o ajuste necessário quando essa violação ocorre. O projeto passa-baixa resultante com coficientes quantizados é então retransformado no tipo de filtro desejado.

7.2.4 Programas Auxiliares

7.2.4.1 Programa ANALIS

Analisa o projeto de um filtro em função da magnitude da resposta em freqüência do projeto de um filtro recursivo fornecido por qualquer um dos programas com esta finalidade. Essa análise consolida o processo de decisão, tanto na necessidade de ativação de um outro programa de otimização, quanto na escolha do melhor projeto obtido, no caso da segunda otimização já ter sido tentada. O intervalo de freqüência é dividido num número de pontos igual a 20 vezes o valor da ordem do filtro, para em seguida ser determinado o valor da magnitude da resposta corespondente a cada um destes pontos. A análise é baseada na norma quadrática, a raiz quadrada do valor médio do desvio quadrático entre a magnitude da resposta em freqüência do filtro projetado e o valor desejado desta função nas faixas de passagens e corte; calcula também os valores do desvio máximo nessas mesmas faixas. No caso de análise automática, em que o próprio sistema decide pela necessidade ou não de uma nova otimização, o programa ativa a rotina de diagnóstico que compara os valores dos desvios, calculados pelas duas formas citadas, com a tolerância permitida nas faixas de passagem e corte. Se qualquer um dos valores dos desvios violar a tolerância, o programa recomenda uma nova otimização.

7.2.4.2 Programa ESTINF

Utiliza o processo empírico de Hermann, Rabiner e Chan [37] para fornecer uma estimativa do tamanho de um filtro FIR. O projeto inicial é transformado num projeto passa-baixa equivalente. Em seguida é calculado o valor estimado do tamanho do filtro (NF) em função da largura da faixa de transição e da tolerância nas faixas de passagem e corte do projeto transformado. Introduziu-se uma pequena alteração na fórmula proposta por Hermann et al., tendo em visto os resultados obtidos em testes, de tal forma que o valor estimado de NF passou a ser calculado da seguinte maneira:

$$NF = 2 \times \frac{D(\delta_1, \delta_2)}{F_c - F_p},$$

onde

$$D(\delta_1, \delta_2) = \left[5.309 \times 10^{-3} \times (\log_{10} \delta_2)^2 + 7.114 \times 10^{-2} \times \log_{10} \delta_1 - - 0.4761\right] \times \log_{10} \delta_2$$
$$- \left[2.66 \times 10^{-3} \times (\log_{10} \delta_1)^2 + 0.5941 \times \log_{10} \delta_1 + 0.4278\right].$$

Os cálculos somente são efetuados para filtros passa-baixa, passa-alta, passa-faixa simétrico, rejeita-faixa simétrico, e transformador de Hilbert. Para os demais o programa estima um valor inicial (NF = 55) para efetuar os cálculos do projeto, servindo como um

ponto de referência para o usuário. Para o caso de filtros de uma única faixa como o transformador de Hilbert, diferenciador, e passa-tudo, o valor de NF é sempre estimado como sendo um número par, por imposição do programa FIR1.

7.2.4.3 Programa EQUALZ

Programa usado para determinar o número e a localização dos polos e zeros em relação ao círculo unitário no plano Z, de um dado filtro IIR para o qual se deseja proceder a equalização de fase. A saída dos dados deste programa segue o padrão ditado pelo programa IIR2 que executa a equalização. Desta forma os dados são fornecidos na seguinte ordem :

- número total de zeros e polos
- número de zeros reais
- número de zeros na fronteira do círculo unitário
- número de zeros complexos
- número de polos reais
- número de polos complexos

A localização dos zeros e polos é dada em termos de ângulos e raios na seguinte ordem: zeros reais, zeros na fronteira do círculo unitário, zeros complexos, polos reais, e polos complexos, obedecendo a seguinte orientação :

- zeros e polos reais são especificados por seus raios
- zeros na fronteira do círculo unitário são especificados por seus ângulos
- zeros e polos complexos são especificados por seus ângulos expressos como uma freqüência (isto é, ângulo em radiano/(2π) × freqüência de amostragem) seguidos por seus raios aos pares, isto é, ângulo e raio, ângulo e raio, etc.

Somente zeros e/ou polos com ângulos positivos são especificados, seus conjugados simétricos são subentendidos.

7.3 Módulo Gráfico

Com o objetivo de auxiliar o usuário na análise do projeto do filtro produzido, foi incorporado ao sistema um módulo gráfico que tem a capacidade de fornecer os gráficos relativos à magnitude da resposta em freqüência, da atenuação em dB, da fase e do atraso de grupo do projeto fornecido. Isto no caso de filtros recursivos. No caso de filtros não-recursivos, são fornecidos os gráficos da magnitude da resposta ao impulso e a atenuação do filtro em dB.

As curvas são obtidas unindo-se pontos (200 pontos) eqüidistantes no intervalo amostrado, normalizado pela freqüência de amostragem. No eixo das abscissas os pontos são determinados com até quatro dígitos de precisão, e duas casas no eixo das ordenadas. Para permitir ao usuário uma melhor visualização de uma determinada área, foi incorporado ao módulo gráfico a capacidade de ampliação (*zoom*) que tem a

vantagem adicional fornecer uma curva com maior nível de detalhes, pois ela é recalculada no intervalo delimitado (a ser ampliado) efetuando-se uma nova divisão noutros duzentos pontos eqüidistantes. Esse processo pode ser repetido até um intervalo se reduzir a um ponto na precisão adotada, ou um valor constante da curva for obtido numa determinada região, quando não haverá possibilidade de melhoria da precisão.

Quando um gráfico é exibido, no rodapé são assinaladas as opções do usuário: zoom (**Z**), gráfico anterior (**PgUp**) gráfico posterior (**PgDn**) e abandonar o módulo gráfico (**Esc**). Se o usuário decidir pela ampliação (teclando **Z**) surge no centro da tela uma linha de referência que delimitará o primeiro limite do intervalo a ser ampliado. As setas de cursor movimentam-na para a direita ou esquerda. O deslocamento da linha é feito pixel a pixel. Se desejar aumentar o passo para 10 pixels o usuário deverá pressionar a tecla junto com a seta de direção. O uso da tecla **Home** transporta a linha de referência ao ponto extremo esquerdo do gráfico e a tecla **End** ao extremo direito. Ao mesmo tempo que a linha é movimentada, o valor da freqüência correspondente é exibido no rodapé. A marcação do primeiro limite de ampliação é feita pressionando a tecla **Enter**, quando a linha de referência estiver sobre o ponto desejado. Imediatamente após a escolha do primeiro limite surge uma segunda linha de referência com as mesmas propriedades da primeira. Feita a escolha do segundo limite os cálculos são refeitos e o gráfico ampliado da região escolhida é exibido.

Os gráficos são apresentados na seguinte ordem: magnitude, atenuação, fase e atraso de grupo. Precionando-se a tecla **PgDn** o próximo gráfico é exibido, e pressionando-se a tecla **PgUp** é exibido o gráfico anterior.

A título de ilustração, considere o projeto de um filtro recursivo passa-baixa com freqüência normalizada de corte nas faixas de passagem e rejeição de 0.25 e 0.35 respectivamente, fornecido pelo módulo numérico. Os gráficos correspondentes a este projeto são apresentados nas figuras 7.1, 7.2, 7.3 e 7.4. A figura 7.5 é uma ampliação da faixa de corte. Note a possibilidade de nova ampliação num intervalo mais restrito.

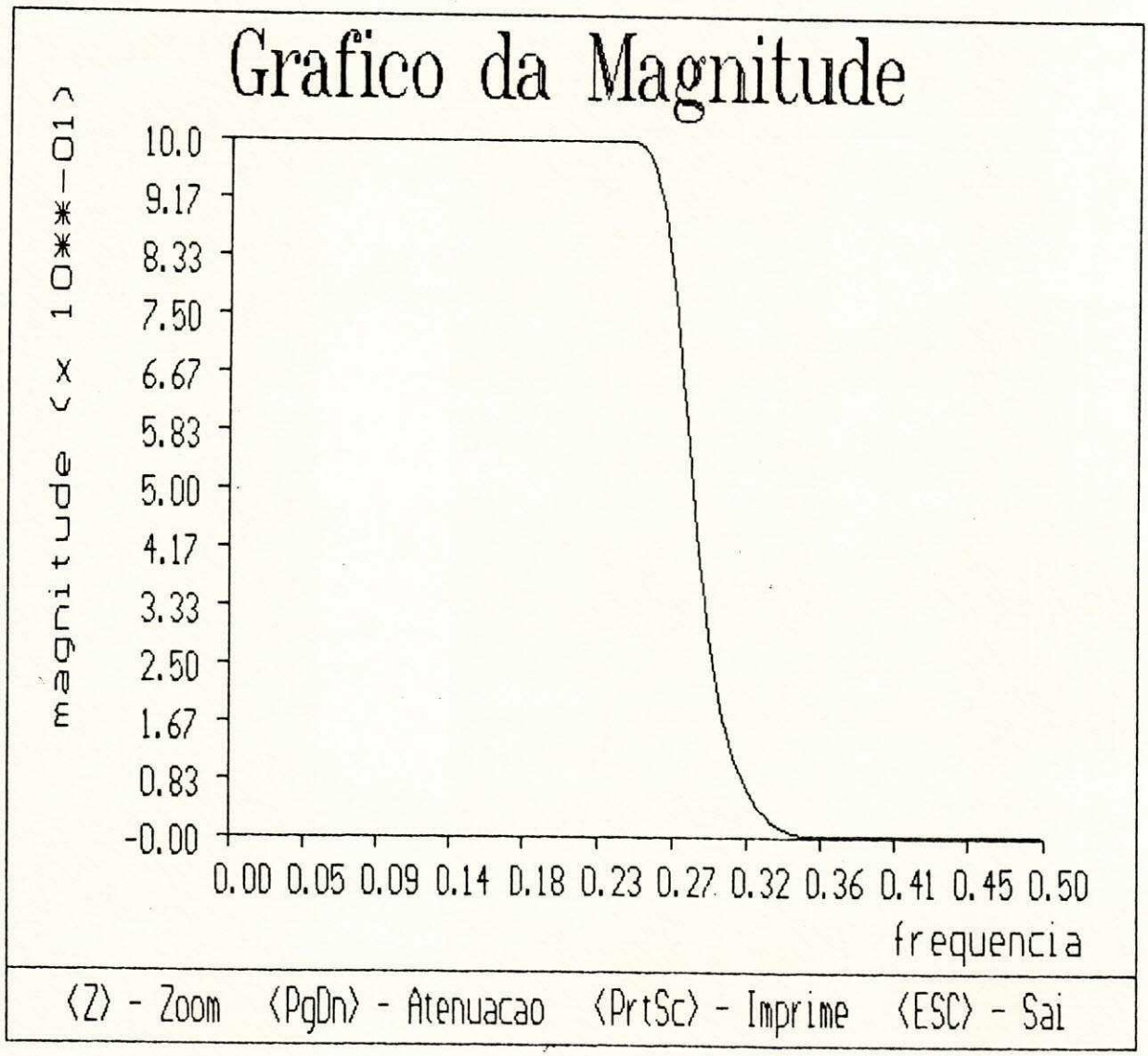

Figura 7.1: Gráfico da Magnitude da Resposta em Freqüência

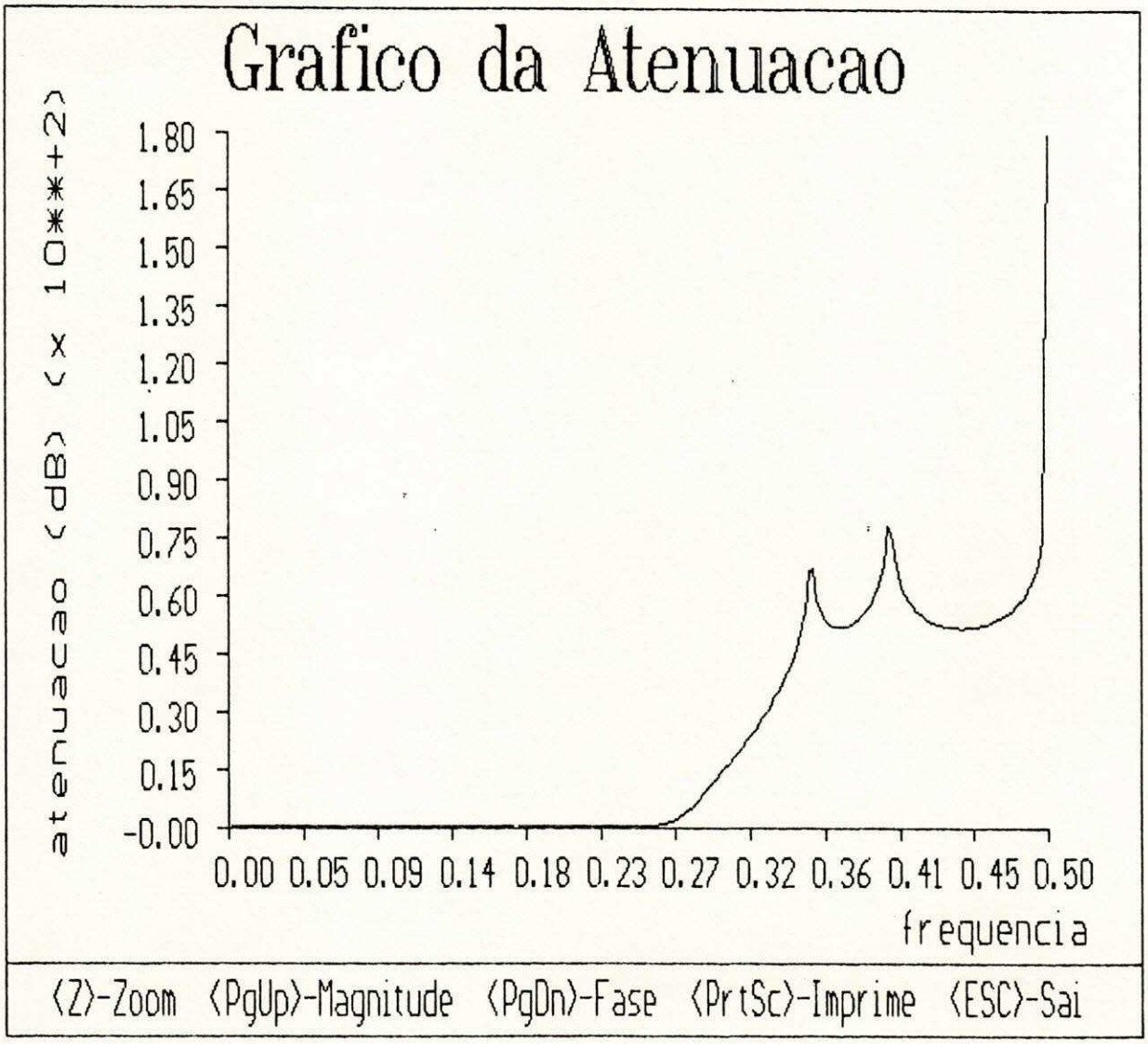

Figura 7.2: Gráfico da Atenuação em dB

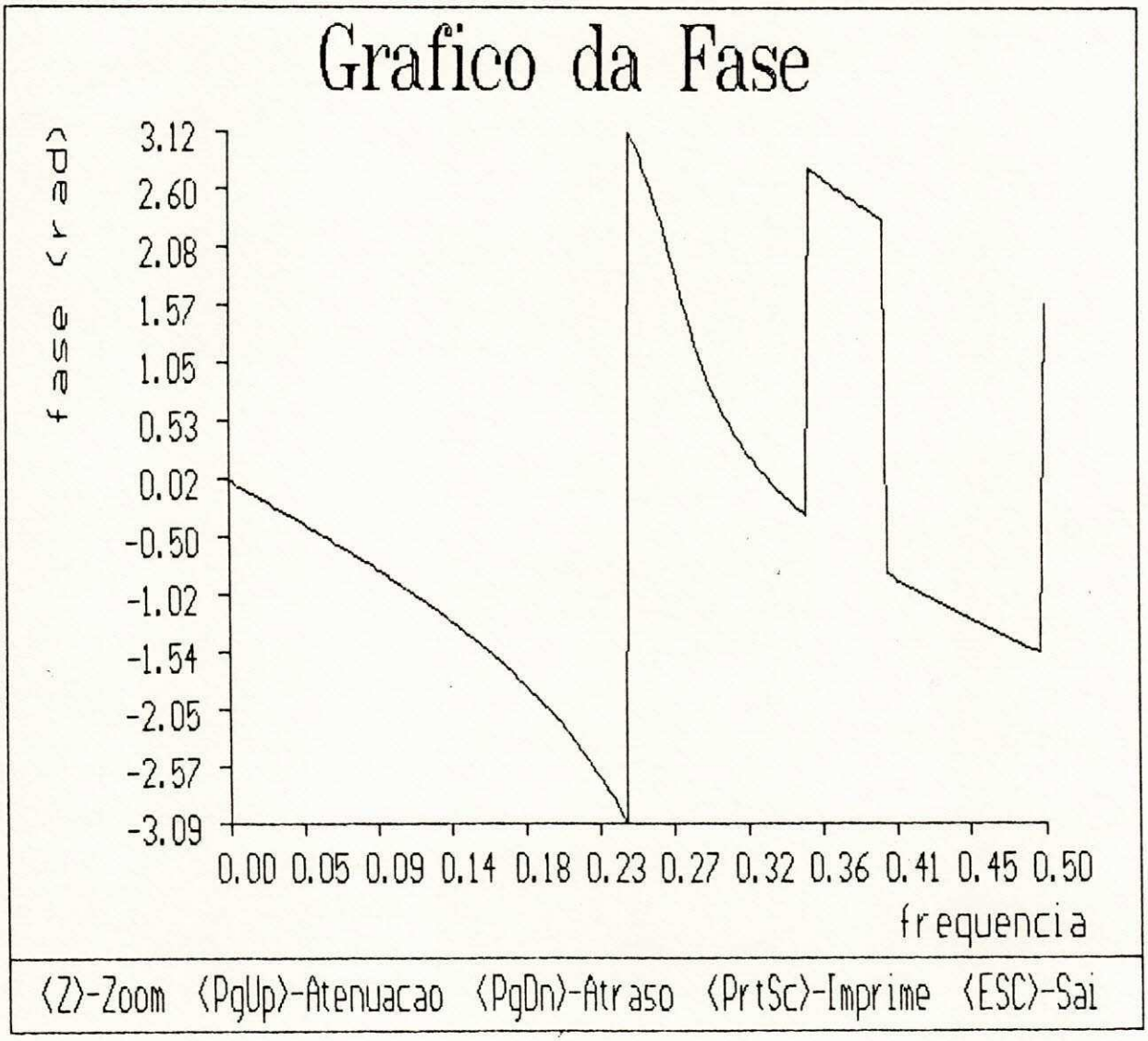

Figura 7.3: Gráfico da Fase

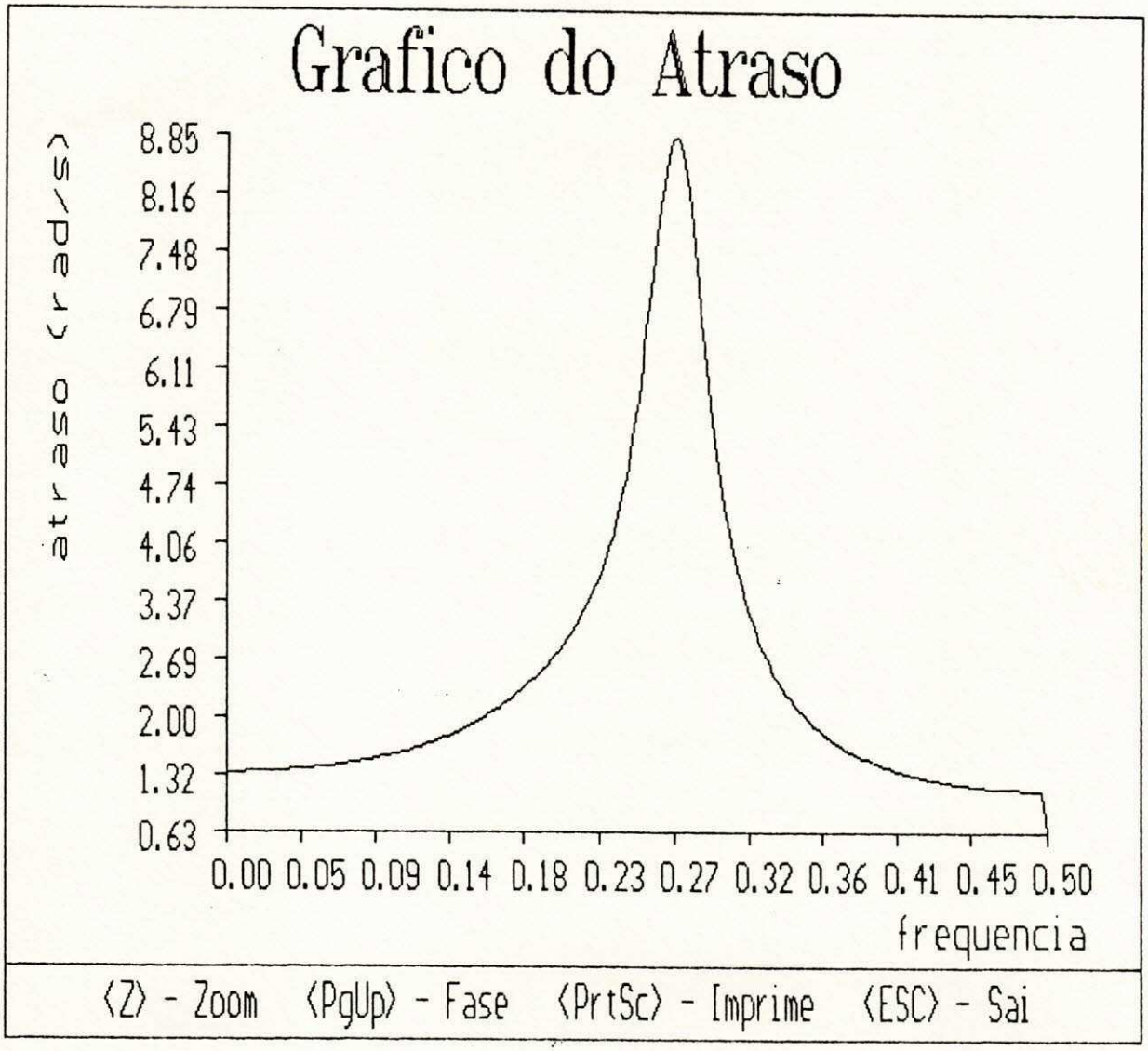

Figura 7.4: Gráfico do Atraso de Grupo

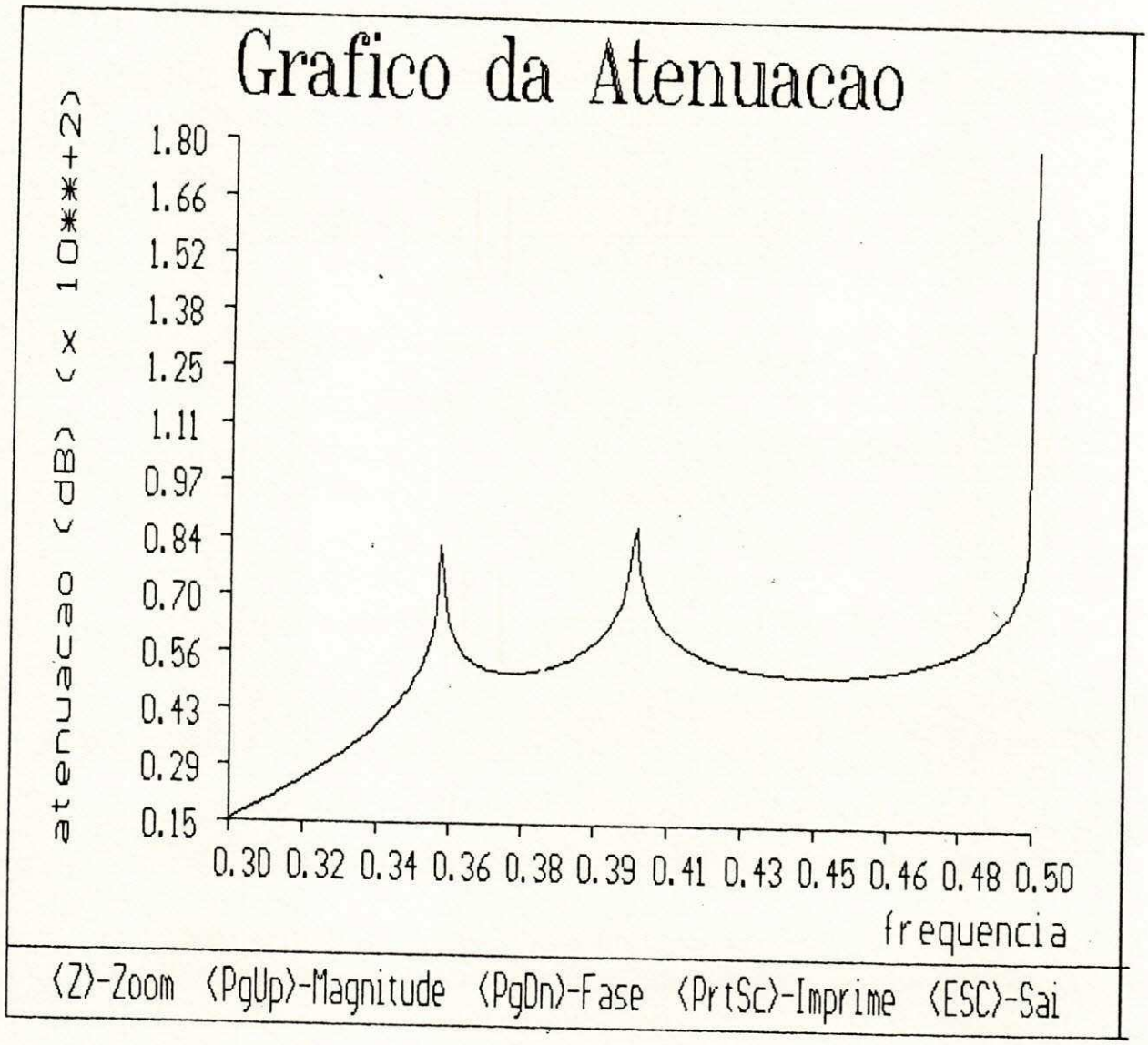

Figura 7.5: Ampliação da Faixa de Corte do Filtro

As próximas seções descrevem como os cálculos são efetuados para a obtenção dos gráficos.

O módulo gráfico está implementado na linguagem de programação C, escolhida devido às facilidades de manuseio de tela e recursos gráficos existentes.

7.3.1 Gráfico de Filtros Recursivos (IIR)

O projeto fornecido pelo módulo numérico é dado na forma dos coeficientes da realização em cascata de seções de 2ª ordem do filtro projetado, ou seja,

$$H(z^{-1}) = k \prod_{i=1}^{N} \frac{A2_i z^{-2} + A1_i z^{-1} + 1}{B2_i z^{-2} + B1_i z^{-1} + 1}.$$

Projetando-se sobre o círculo unitário no plano Z, tem-se

$$H(e^{j\omega}) = H(z)\Big|_{z = e^{j\omega}}.$$

A magnitude da resposta em freqüência é calculada por

$$\left| H(e^{j\omega_r}, r) \right| = k \left| \prod_{i=1}^{N} \frac{NN_i(e^{j\omega_r})}{DD_i(e^{j\omega_r})} \right|, \quad r = 1, 2, \ldots, N_p,$$

onde

$$NN_i(e^{j\omega_r}) = (1 + 2A2_i)\cos\omega_r + A1_i + [(1 - A2_i)\mathrm{sen}\omega_r]j,$$

$$DD_i(e^{j\omega_r}) = (1 + 2B2_i)\cos\omega_r + B1_i + [(1 - B2_i)\mathrm{sen}\omega_r]j.$$

O intervalo normalizado (de 0 à maior freqüência/freqüência de amostragem) é dividido pelo número de pontos (N_p) usado no traçado do gráfico (no caso 200 pontos). As freqüências correspondentes são obtidas de

$$\omega_r = F_a + r \times \frac{F_n - F_1}{N_p - 1}, \quad r = 1, 2, \ldots, N_p,$$

onde

F_a freqüência de amostragem,

F_n, F_1 a maior e a menor freqüências amostradas, respectivamente.

A atenuação é dada por

$$\mathrm{at}(r) = -20 \log \left| H(e^{j\omega_r}) \right|, \quad r = 1, 2, \ldots, N_p.$$

A fase é calculada como sendo

$$\phi(r) = \text{arctg}\left[\frac{H_{Im}(e^{j\omega_r})}{H_{Re}(e^{j\omega_r})}\right], \quad r = 1, 2, \ldots, N_p.$$

O atraso de grupo é obtido pela fórmula

$$GD(r) = \sum_{i=1}^{N}\left[\frac{N2_i(e^{j\omega_r})}{DD_i(e^{j\omega_r})} - \frac{N1_i(e^{j\omega_r})}{NN_i(e^{j\omega_r})}\right], \quad r = 1, 2, \ldots, N_p,$$

onde

$$N1_i = 2\cos\omega_r + A1_i + (2\text{sen}\omega_r)j,$$
$$N2_i = 2\cos\omega_r + B1_i + (2\text{sen}\omega_r)j.$$

A implementação destas fórmulas de cálculo foi feita de maneira eficiente, tendo-se o cuidado de evitar possíveis erros de *underflow* e *overflow*.

7.3.2 Gráfico de Filtros Não-Recursivos (FIR)

As fórmulas de cálculo utilizadas foram as propostas por Rabiner e Gold em [23]. Dependendo de sua natureza, os filtros foram classificados em quatro tipos, filtros de resposta simétrica e de tamanho par (1º caso), filtros de resposta simétrica e tamanho impar (2º caso), filtros de resposta anti-simétrica e tamanho par (3º caso) e filtro de resposta anti-simétrica e tamanho impar (4º caso). As fórmulas utilizadas para cada um dos casos são as seguintes:

a) 1º caso

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2}\sum_{n=0}^{(N-1)/2} a(n)\cos(\omega n),$$

onde

$$a(0) = h[(N-1)/2],$$
$$a(n) = 2h[(N-1)/2 - n], \quad n = 1, 2, \ldots, (N-1)/2.$$

A magnitude é calculada por:

$$|H(e^{j\omega})| = \left|\sum_{n=0}^{(N-1)/2} a(n)\cos(\omega n)\right|.$$

b) 2º caso

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \sum_{n=1}^{N/2} b(n) \cos[\omega(n-1/2)],$$

onde

$$b(n) = 2h(N/2-n), \quad n = 1, 2, \ldots, (N-1)/2.$$

A magnitude é calculada por:

$$|H(e^{j\omega})| = \left| \sum_{n=1}^{N/2} b(n) \cos[\omega(n-1/2)] \right|.$$

c) 3º caso

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} e^{j\pi/2} \sum_{n=1}^{(N-1)/2} c(n)\text{sen}(\omega n).$$

onde

$$c(n) = 2h[(N-1)/2-n], \quad n = 1, 2, \ldots, (N-1)/2.$$

A magnitude é calculada por

$$|H(e^{j\omega})| = \left| \sum_{n=1}^{(N-1)/2} c(n)\text{sen}(\omega n) \right|.$$

d) 4º caso

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} e^{j\pi/2} \sum_{n=1}^{N/2} d(n)\text{sen}[\omega(n-1/2)],$$

onde

$$d(n) = 2h(N/2-n), \quad n = 1, 2, \ldots, (N-1)/2.$$

A magnitude é calculada por

$$|H(e^{j\omega})| = \left| \sum_{n=1}^{N/2} d(n) \cos[\omega(n-1/2)] \right|.$$

Em todos os casos a atenuação é calculada como sendo

$$\mathrm{at}(\omega) = -20 \log_{10} \left| H(e^{j\omega}) \right|.$$

7.4 Sumário

Neste capítulo foi descrita a estrutura de computação numérica do SIPREX, sendo detalhados os seus módulos, ou seja, o módulo numérico e o módulo gráfico.

CAPÍTULO 8
TESTES DE VALIDAÇÃO E AVALIAÇÃO DO SISTEMA

8.1 Introdução

A validação de um sistema é um processo demorado, que somente pode ser dado como concluído se o sistema apresenta um desempenho satisfatório, após ser submetido a um número significativo de execuções com casos reais. Numa etapa inicial de testes a simulação de problemas reais, é um processo bastante aceitável, desde que os problemas apresentados sejam semelhantes aos problemas reais, com um certo grau de diversificação, de modo que permita uma parametrização do sistema em situações diversas. Esse foi o procedimento adotado, em que o sistema foi submetido a uma bateria de testes, procurando-se explorar o domínio de aplicação do sistema. O desempenho foi avaliado tendo por base a precisão da resposta gerada e o tempo necessário a obtenção desta resposta, sendo o tempo computado desde o início da etapa de especificações do problema até a obtenção da resposta final.

Uma amostra significativa dos testes processados será apresentada na seção seguinte.

8.2 Problemas de Teste

Dentre os problemas utilizados durante o processo de validação e avaliação do sistema, quatro deles são apresentados a seguir. O critério de escolha utilizado foi o da diversificação de situações de modo a levar o sistema a fornecer a solução por caminhos variados. São eles: projeto de um filtro passa-baixa aplicado à área de voz, projeto de um filtro não-recursivo do tipo passa-alta, projeto de um filtro recursivo rejeita-faixa com coeficientes quantizados e equalização de fase de um filtro não-recursivo do tipo passa-baixa. Os resultados dos testes são apresentados a seguir.

8.2.1 Projeto de um Filtro Passa-baixa Aplicada à Área de Voz

O projeto desejado é de um filtro passa-baixa de tamanho 25, com freqüência limite superior da faixa de passagem de 2.000 Hz e limite inferior da faixa de corte de 3.500 Hz, freqüência de amostragem de 10.000 Hz, desvios máximos normalizados de 0,1 e peso 10 na faixa de passagem e 1 na faixa de corte.

Solução

Como não foi especificada a natureza do filtro (se FIR ou IIR) e por se tratar de uma aplicação na área de voz, o sistema optou por um projeto FIR. O projeto final obtido é mostrado na figura 8.1. Quando o usuário foi induzido a criticar o filtro

produzido, ele não aceitou o valor da largura da faixa de transição, pois as freqüências limites (superior da faixa de passagem e inferior da faixa de corte) excedem os valores especificados. Isto fez com que o módulo de aprendizado fosse acionado, o que resultou numa nova solução. Observando-se a figura 8.2, percebe-se claramente a redução da largura da faixa de transição deste segundo projeto, o que satisfez ao usuário.

Todo o processo de definição do problema, apresentação da solução intermediária, crítica desta solução e solução final, está contido em detalhes no apêndice.

8.2.2 Projeto de um Filtro Não Recursivo do Tipo Passa-alta

Desta feita, deseja-se produzir um filtro não-recursivo passa-alta, de ordem 30, com freqüência de corte na faixa de passagem de 3 KHz e na faixa de corte de 2 KHz, freqüência de amostragem de 10 KHz, desvio normalizado permitido de até 0,001 na faixa de passagem e de 0,01 na faixa de corte. Os pesos utilizados nas faixas de passagem e corte são de 10 e 1, respectivamente.

Solução

A figura 8.3 mostra uma solução totalmente degenerada, obtida pelo sistema numa primeira tentativa. Isso é perfeitamente possível pelo fato de boa parte das informações sobre os algoritmos, contidas na base de conhecimentos, ter originado de simulação de casos reais, sendo então aceitável que, para alguma situação, um dado algoritmo apresente um comportamento diferente do usual. O usuário logicamente criticou o projeto exigindo a observância das especificações violadas (*ripples* inaceitáveis em todas as faixas, por exemplo). A segunda solução, conseguida desta feita com a intervenção do módulo de aprendizado, é mostrada na figura 8.4. Pode-se observar que o projeto agora atende as especificações exigidas.

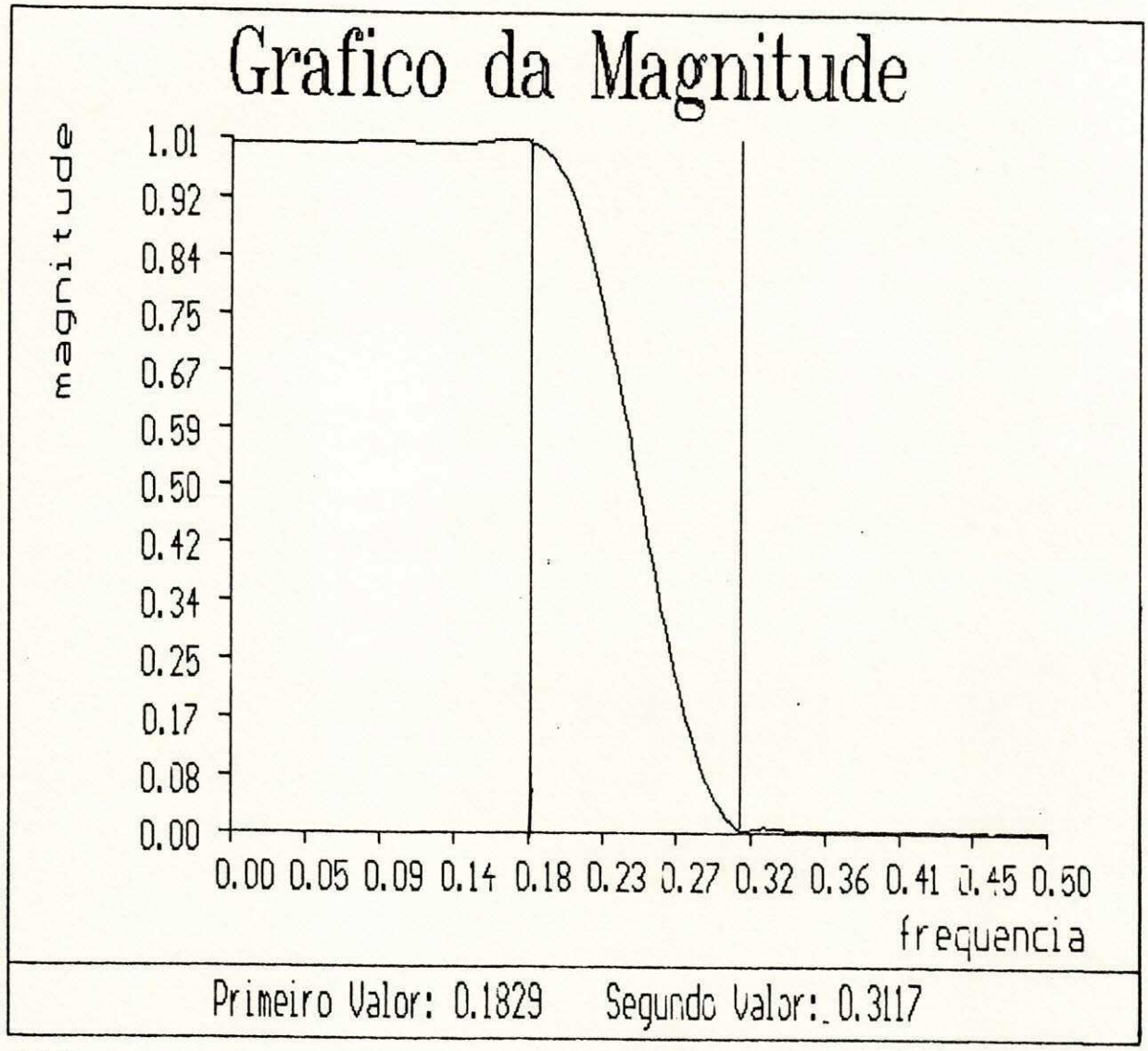

Figura 8.1: Exemplo 1: Primeira Solução Encontrada

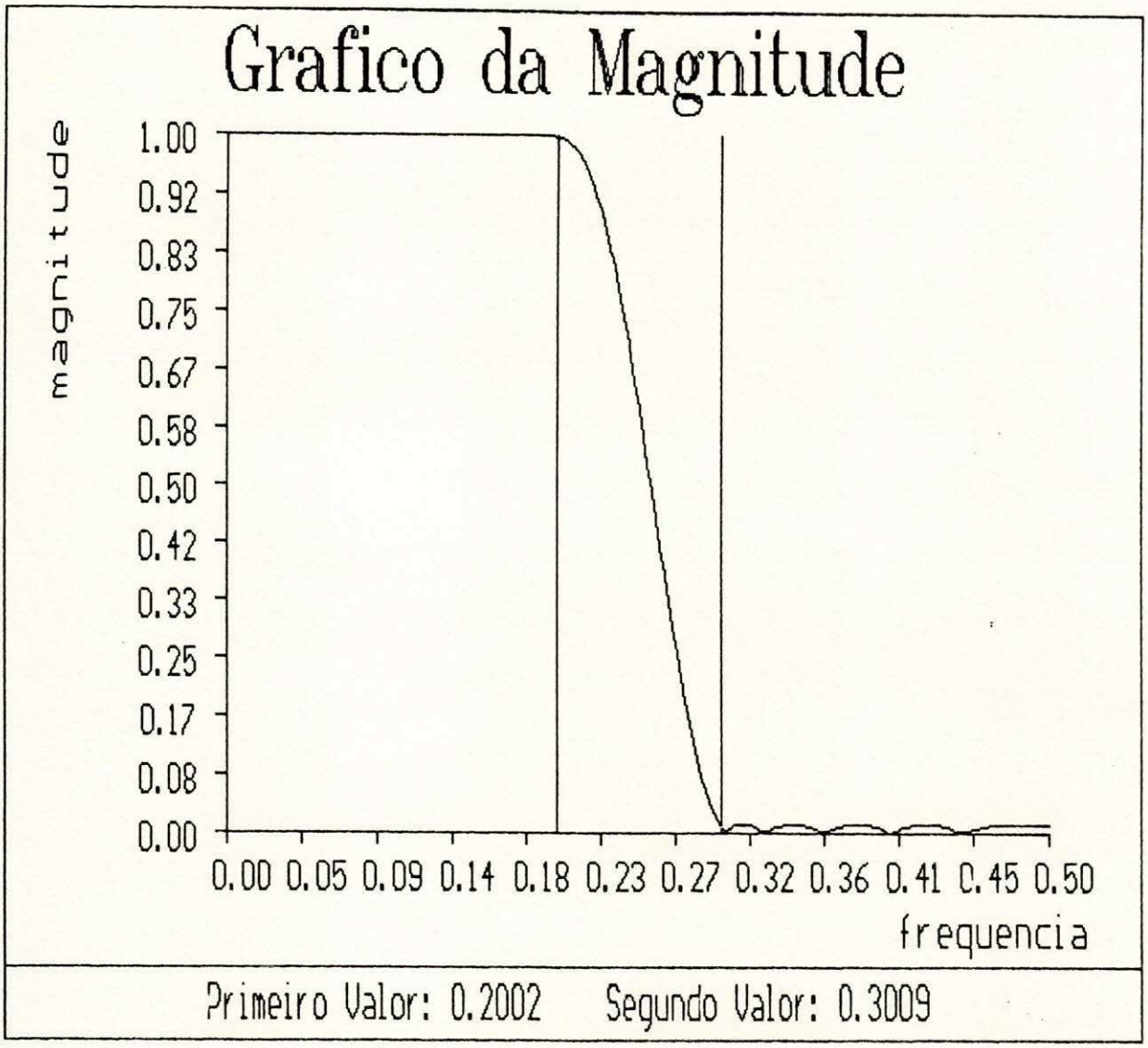

Figura 8.2: Solução após Ativação do Módulo de Aprendizado

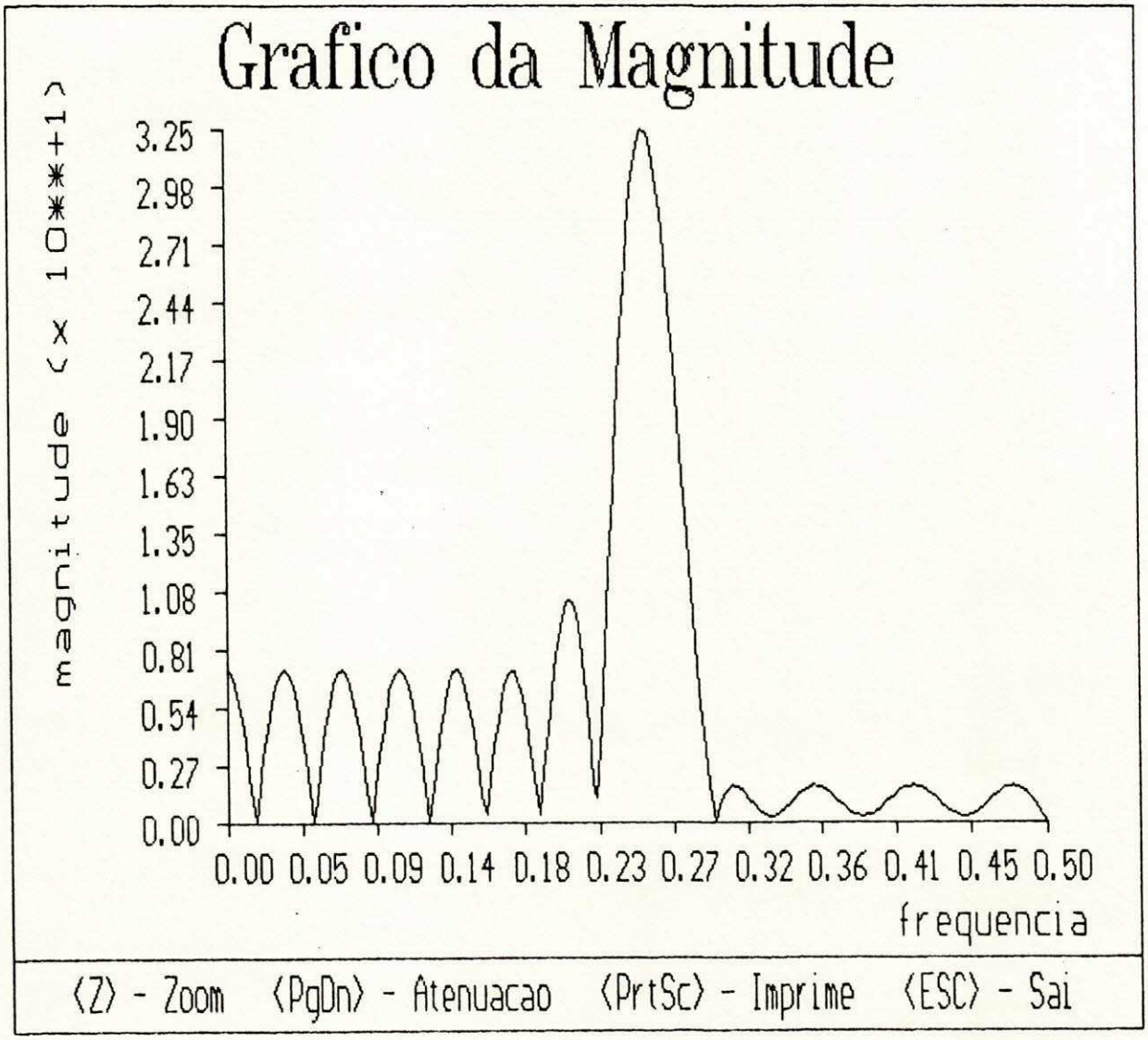

Figura 8.3: Exemplo 2: Primeira Solução Encontrada

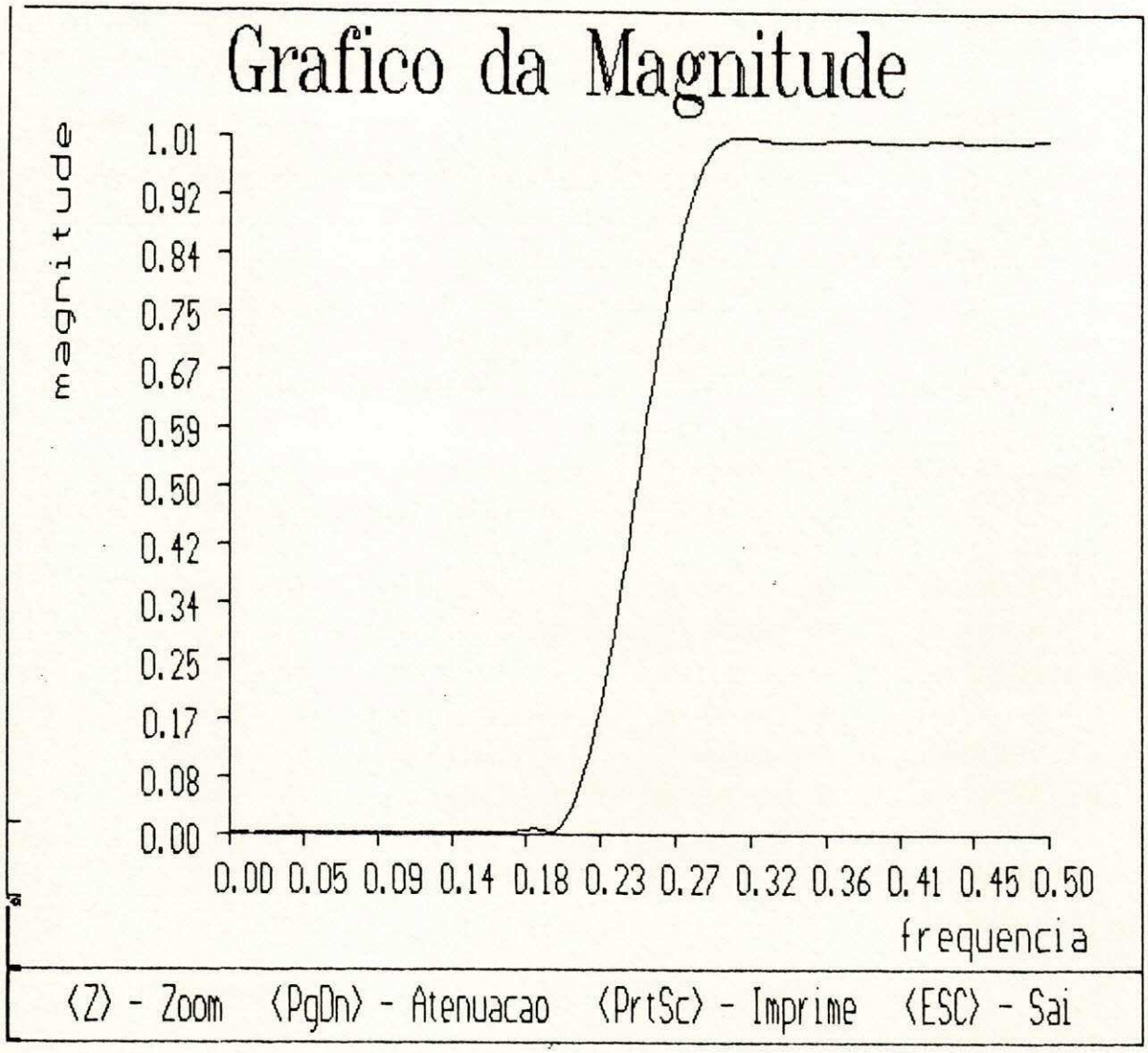

Figura 8.4: Solução após Ativação do Módulo de Aprendizado

8.2.3 Projeto de um Filtro Recursivo do Tipo Rejeita-faixa

O projeto desejado é de um filtro IIR rejeita-faixa, com freqüência de amostragem de 100 KHz, faixas de passagem de 0 KHz a 15 KHz e de 35 KHz a 50 KHz, faixa de corte de 17 KHz a 33 KHz, desvios máximos de 0,01 e peso de 10 nas faixas de passagem e de 1 na faixa de corte. O projeto final deve apresentar os coeficientes quantizados, com tamanho de palavra de 16 bits.

Solução

A ordem do filtro não foi especificada anteriormente, o que foi feito pelo sistema, calculando este valor em 6. O usuário deseja analisar ele próprio o projeto final, opção apresentada pelo sistema. A outra opção oferecida seria a de análise automática: o sistema decide sem participação do usuário. Desta forma, o sistema exibe os valores dos erros do projeto produzido, calculados de duas maneiras (erro médio quadrático e erro máximo), em cada uma das faixas (faixas 1 e 3, passagem e faixa 2, corte), como pode ser visto na figura 8.5. O usuário então verifica que o projeto preenche os requisitos, com erros satisfatórios em todas as faixas. Feito isto, passa-se a determinação do projeto quantizado, obtendo-se os seguintes coeficientes:

A2	A1	B2	B1
.10000000E+01	.93983459E+00	.93276978E+00	.10895538E+01
.10000000E+01	.76615906E+00	.72804260E+00	.10233917E+01
.10000000E+01	.32922363E+00	.29811096E+00	.76641846E+00
.10000000E+01	-.33767700E+00	.29843140E+00	-.76983643E+00
.10000000E+01	-.78065491E+00	.73060608E+00	-.10314789E+01
.10000000E+01	-.94691467E+00	.93273926E+00	-.10969086E+01

A figura 8.6 exibe o projeto final com os coeficientes quantizados.

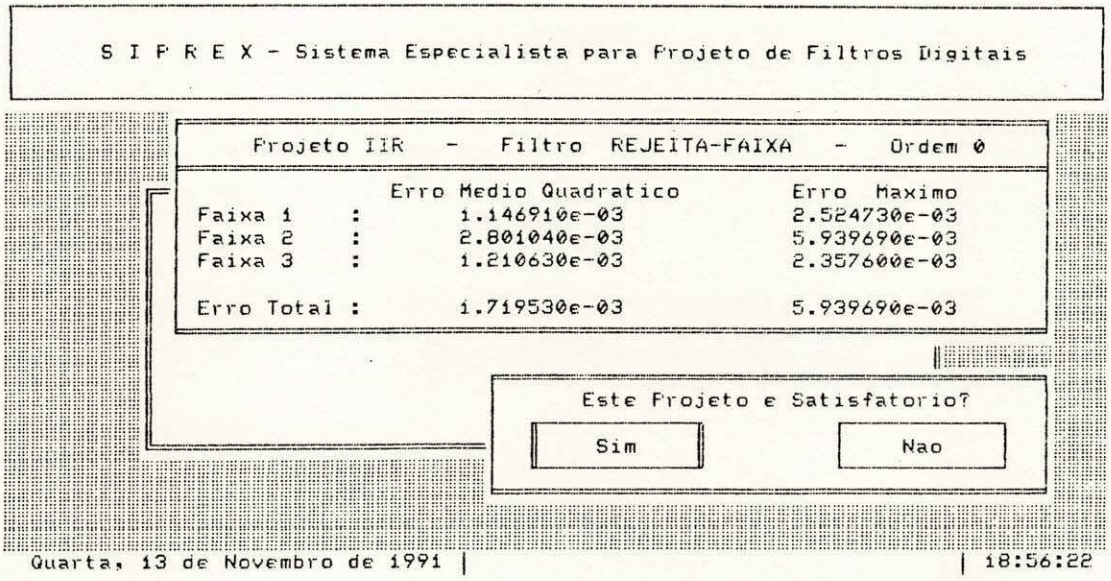

Figura 8.5: Informações para Auxiliar Análise, Apresentando os Desvios em cada Faixa

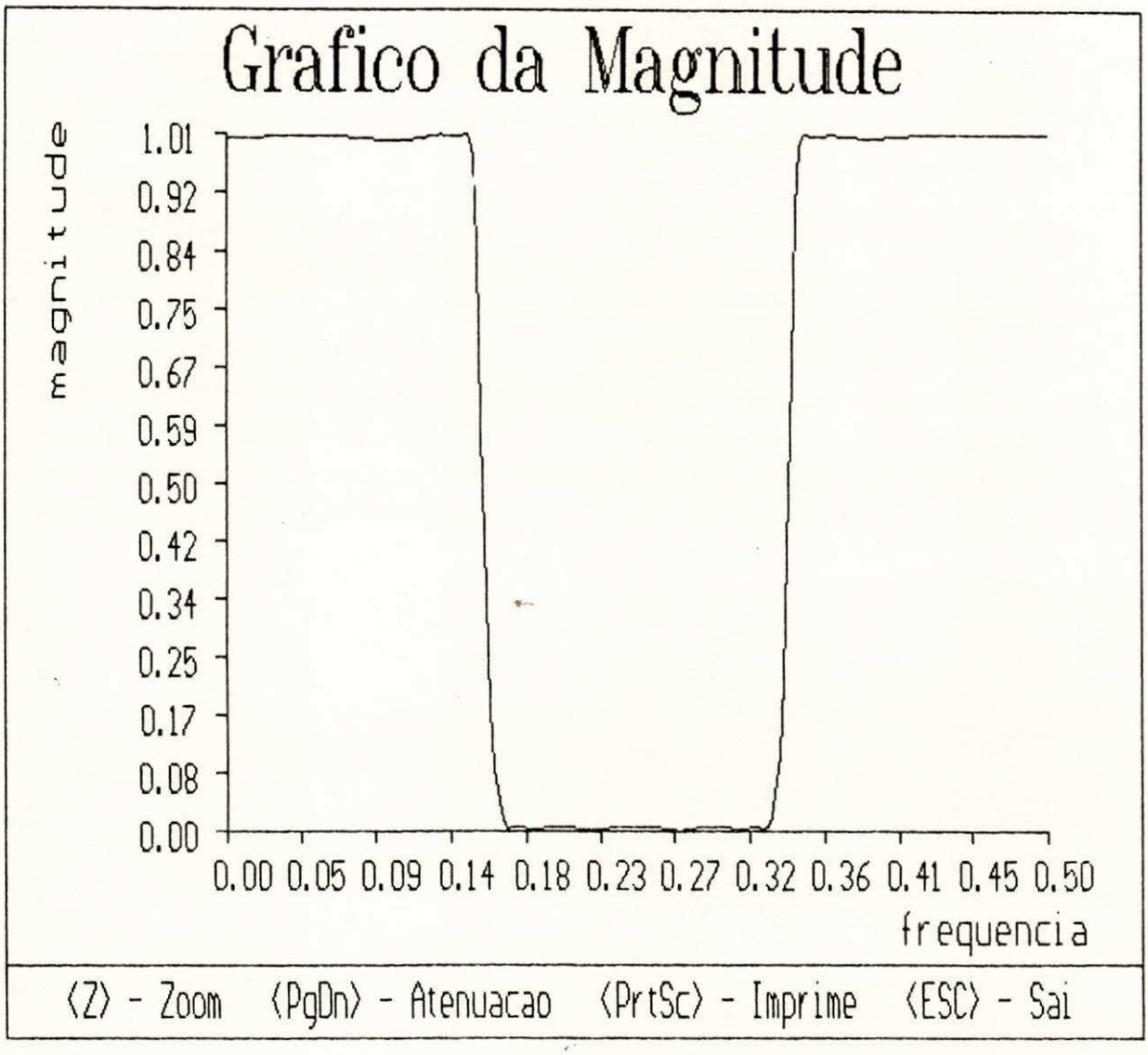

Figura 8.6: Projeto Quantizado, Utilizando Palavra de 16 bits

8.2.4 Equalização de Fase de um Filtro Recursivo do Tipo Passa-baixa

O filtro fornecido para equalização de fase, tem as seguintes especificações: freqüência limite superior na faixa de passagem de 1.800 Hz e inferior da faixa de corte de 2.500 Hz. Freqüência de amostragem de 10.000 Hz. Desvios em ambas as faixas de 0,1 e peso de 10 e 1 para as faixas de passagem e corte.

Solução

Processada a equalização de fase do filtro dado, observou-se que o projeto produzido apresentava desvios fora dos especificados, como pode ser comprovado pelo resultado da análise dos desvios mostrado na figura 8.7. Nesse caso, o usuário solicita uma nova otimização. O sistema processa a segunda otimização, obtendo um segundo projeto. A figura 8.8 mostra os desvios apresentados pelos dois projetos. O usuário escolhe então o projeto que apresenta o menor desvio, isto é, o segundo projeto, cuja curva de magnitude da resposta em freqüência é mostrada na figura 8.9. A figura 8.10 mostra a fase do filtro produzido.

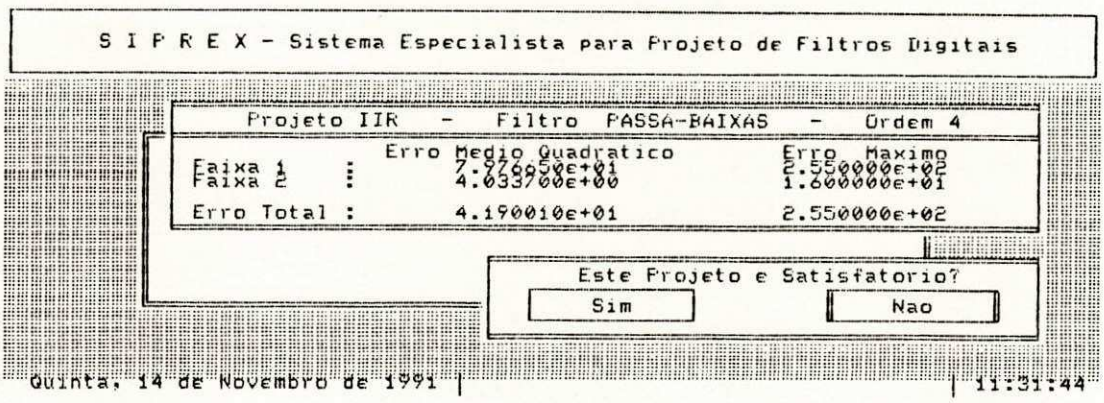

Figura 8.7: Exemplo 4: Desvio Fora do Especificado. Uma Nova Otimização é Necessária

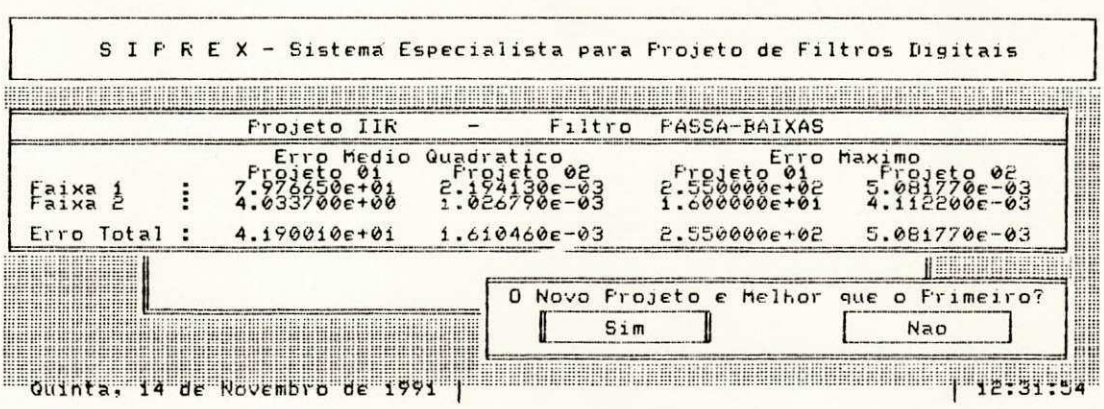

Figura 8.8: Desvios Apresentados pelos Projetos 1 e 2

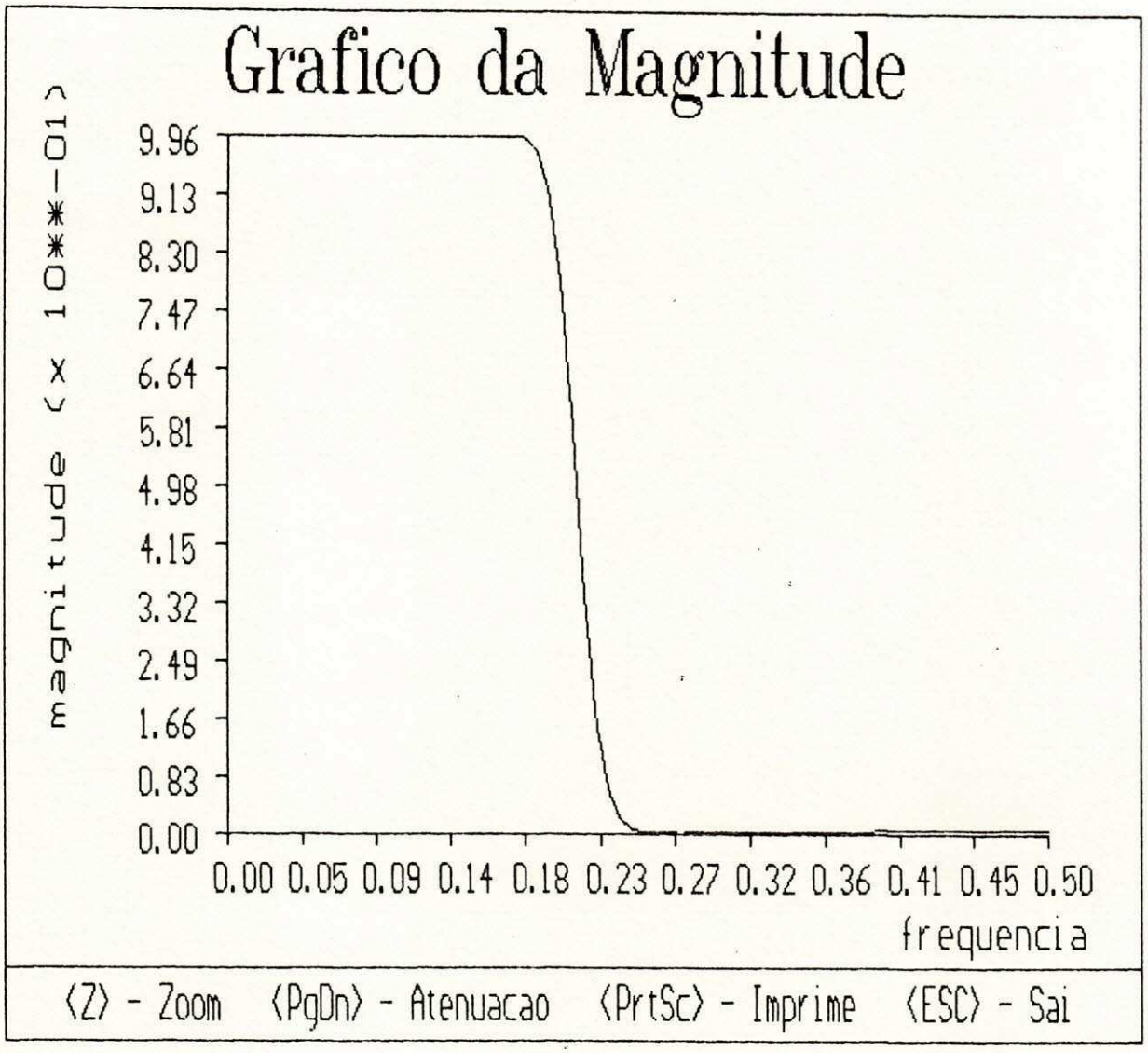

Figura 8.9: Exemplo 4: Magnitude da Resposta do Filtro (Projeto final)

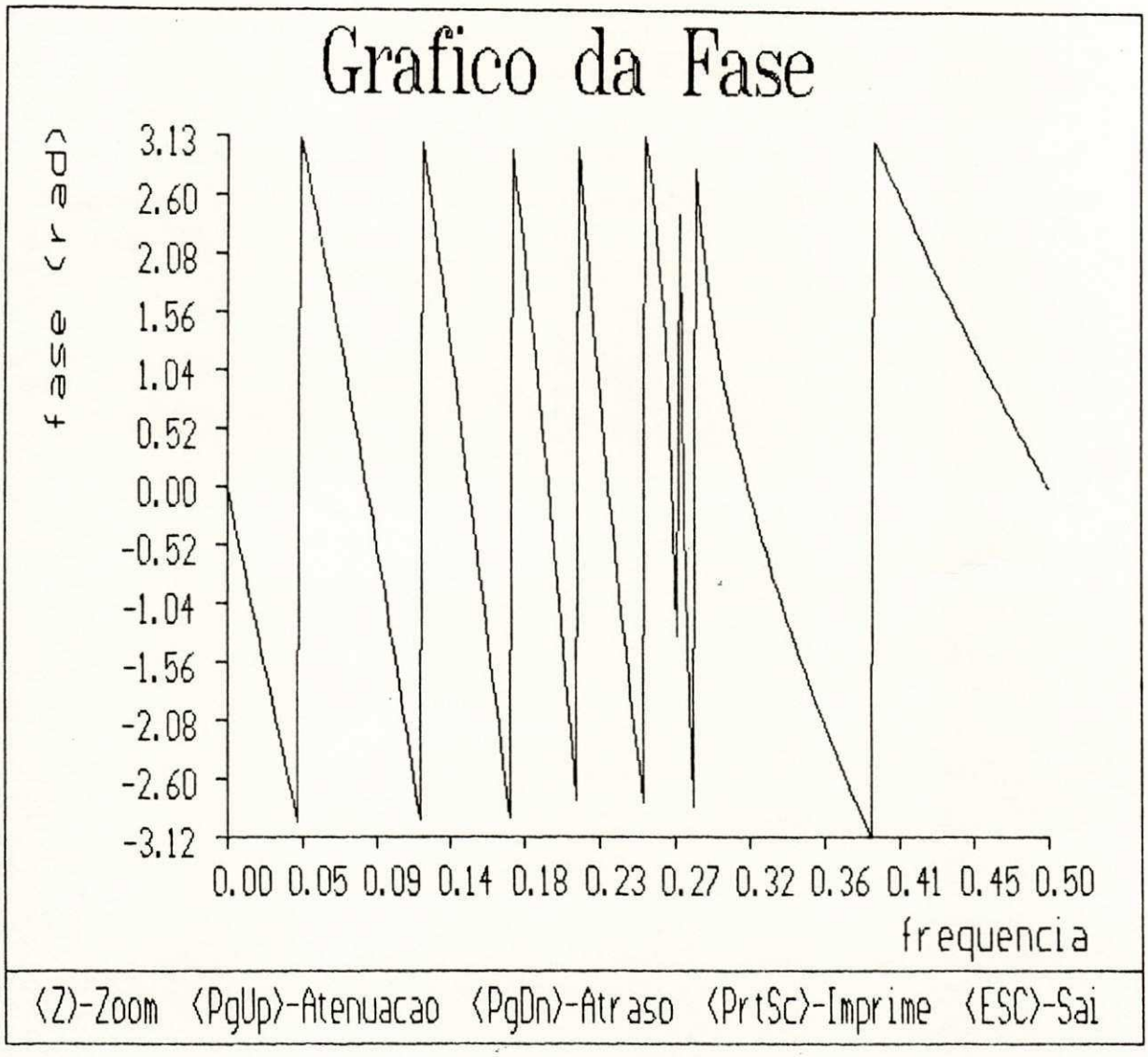

Figura 8.10: Fase do Filtro

8.3 Conclusão

Pelos testes apresentados (e outros processados), o desempenho do sistema foi considerado satisfatório, embora do processo de validação tenha sido completada apenas a etapa inicial. A segunda etapa terá início com a implementação do SIPREX, em caráter experimental, em alguns centros de pesquisa com a finalidade de submeter o sistema a problemas reais, sendo o desempenho do sistema permanentemente acompanhado. O retorno destas informações contribuirá significamente para o ajuste do sistema.

CAPÍTULO 9
CONCLUSÕES

9.1 Introdução

Quando se iniciaram os trabalhos relativos ao projeto SIPREX, em 1987, o conceito de sistemas acoplados era ainda nascente e pouco se conhecia sobre eles. No final daquele ano foi concebido o EXBITAN, o primeiro resultado prático da pesquisa iniciada. O EXBITAN apresentou inovações tanto no seu domínio de aplicação, cálculo numérico básico, como no controle da ativação dos algoritmos e a introdução de alguma análise dos resultados obtidos. A partir do conhecimento adquirido com a construção do EXBITAN, novos estudos foram conduzidos, resultando na construção de um segundo protótipo, o FFTEX, com aplicação no cálculo das transformadas rápidas de Fourier. O FFTEX representou uma evolução em relação ao EXBITAN e aos sistemas acoplados em geral em diversos aspectos, entre eles:

a) Foi introduzida a técnica de simulação como reforço ao processo tradicional de aquisição de conhecimento.
b) A escolha de um determinado algoritmo de cálculo é feita tendo por base valores porcentuais de graus de satisfação fornecidos pelo usuário.
c) Uma técnica original de aprendizado automático baseado no estudo estatístico de casos passados, que torna o sistema mais realista e confiável.

O sistema SIPREX incorpora os resultados das experiências obtidas com a construção desses dois protótipos. Diversas outras inovações foram acrescentadas, que podem ser constatadas na sua estrutura de funcionamento e nos módulos que o compõem, como por exemplo no emprego de interface inteligente com a finalidade de direcionar e facilitar o processo de especificação do problema e no uso de um aprendizado automático seletivo, onde somente usuários classificados (especialistas credenciados) têm permissão de desencadear o processo de aprendizado que resulta na alteração da base de conhecimentos do sistema.

Ao mesmo tempo que se avançava nesta pesquisa, outros trabalhos na mesma linha foram surgindo, tendo se tornado cada vez mais comum a adoção de técnicas de IA para auxiliar a elaboração de projetos de Engenharia [39], o que comprova a relevância de trabalhos como o SIPREX. Como exemplo, podem ser citados dois trabalhos recentemente publicados e que divulgam sistemas construídos com o intuito de auxiliar o desenvolvimento de projetos de sistemas digitais, para aplicação em processamento digital de sinais e controle automático de processos.

O primeiro deles, o sistema ACAI, desenvolvido por Favier et al. [9], é um sistema baseado em conhecimento para aplicações em controle automático de processos. A idéia de Favier é de dotar o sistema ACAI de quatro tipos de funções:

a) Função de escolha - auxilia o usuário a utilizar pacotes existentes, escolhendo os métodos e fornecendo as regras para especificar os parâmetros dos métodos selecionados.
b) Aconselhamento - o usuário toma todas as decisões, mas o SE sempre que solicitado informa sobre as escolhas dos dados e resultados inconsistentes ou mesmo em conflito.
c) Assistência parcial - o usuário e o SE cooperam. O SE faz sugestões e justifica sua coerência. O usuário escolhe as proposições que julgar mais adequadas à situação proposta.
d) Assistência total - o SE toma todas as decisões sem nenhuma intervenção do usuário que é permitido contudo indagar ao SE as razões e as justificativas de suas escolhas.

Atualmente o sistema ACAI é composto de uma quinzena de métodos de identificação (métodos clássicos, adaptativos e robustos) e de alguns métodos de controle (controles antecipativos e proporcional integral derivativo - PID). Apenas o primeiro tipo das funções pretendidas para o sistema ACAI está em funcionamento, ou seja a de escolha dos métodos. As demais estão ainda em desenvolvimento.

Outro sistema com aplicação em projetos de filtros digitais que merece ser citado é o sistema de Nie et al. [28], que já foi mencionado na seção 4.2. É composto de três módulos independentes, sem conexão entre eles, cada um com uma função distinta: o módulo de síntese, o módulo de análise e um módulo de escolha. Os dois primeiros módulos desempenham a função de CAD, efetuando a síntese, a análise e a simulação de filtros FIR e IIR, com precisão infinita e finita em ambos os casos. O terceiro módulo é composto de um sistema especialista que tem a função de baseado nas especificações do projeto fornecidas pelo usuário, auxiliá-lo a escolher o método apropriado de cálculo existente nos outros módulos e a relação de parâmetros a serem fornecidos. Como já foi observado (ver seção 4.2) o acoplamento entre as informações geradas pela parte de computação simbólica do sistema (o SE) e a parte de cálculos numéricos (módulos 1 e 2) é feita sob controle do próprio usuário, o que de certa forma transgride o conceito de sistemas acoplados.

Os dois sistemas apresentam em comum a idéia de uso de CAD na elaboração de projetos e o fato de possuirem um sistema baseado em conhecimentos para a escolha de métodos. O projeto de Favier é mais ambicioso, prevendo o acoplamento total entre as duas partes (numérica e simbólica). O ponto forte do sistema de Nie é a solução adotada para o motor de inferência do SE, utilizando tipos de objetos definidos como operadores ao invés do emprego usual de regras de produção. O conhecimento no

sistema de Nie é representado por cinco tipos de entidades: a entidade chamada parâmetros, a entidade chamada métodos, a entidade chamada operadores contendo a estratégia de resolução, a entidade chamada contexto que realiza o controle da estrutura, e a entidade chamada regra que representa o conhecimento dos especialistas sobre os parâmetros e/ou os métodos.

A análise destes dois projetos, suas capacidades e limitações, permite colocar em pespectiva a contribuição trazida pelo SIPREX. O SIPREX além de ser um sistema mais completo, do ponto de vista de acoplamento e de inteligência, que os anteriores, aborda e apresenta soluções de problemas que nem sequer foram mencionados pelos demais. Um desses problemas é o das especificações dos projetos, cuja quantidade de informações necessária pode resultar numa operação estafante, confusa e dar origem a diversos tipos de erros. A solução adotada para esse problema foi o uso de uma Interface Inteligente que permite o voluntariamento do usuário de maneira clara e amigável, dotada da capacidade de *Help on Line,* podendo esclarecer dúvidas do usuário na definição de algum parâmetro. Simultaneamente, a parte inteligente desta interface, realiza deduções que visam reduzir o questionamento ao usuário, e testa *in loco* a consistência dos dados fornecidos, reduzindo também as possibilidades de ocorrência de erros.

Um segundo problema abordado no SIPREX, é o do gerenciamento e acoplamento entre as diversas partes que compõem o sistema, o que não é uma tarefa trivial, principalmente em se tratando de um sistema acoplado. As informações fornecidas pelo usuário devem satisfazer as necessidades da parte simbólica e da parte numérica do sistema, lembrando que cada um dos métodos que compõem a parte numérica possui uma nomenclatura própria e exige o fornecimento de um conjunto de parâmetros distintos. Visando atingir uma harmonia entre as partes, procurou-se determinar um conjunto mínimo de informações, tal que a partir deste fosse possível gerar novas informações com o intuto de satisfazer todas as partes envolvidas. A determinação desse conjunto mínimo de informações faz com que qualquer que seja o projeto pretendido, o usuário forneça o mesmo tipo de informação, tornando o processo de escolha de um método de projeto totalmente transparente ao usuário. O gerenciamento e a ligação entre os diversos módulos do SIPREX são feitas pelo módulo de controle, que possui ainda a função de gerenciar a saída dos dados oferecendo opções quanto a maneira de fazê-la (numérica, gráfica, tela, impressora).

Um terceiro problema abordado no SIPREX é o do auto-refinamento do sistema através do uso de técnicas de aprendizado automático. Embora sendo um atributo desejável a todos os sistemas inteligentes, raramente esta capacidade está presente, constituindo-se numa ferramenta de refino de sistemas pouco explorada. A técnica particular de aprendizado adotada pelo SIPREX é feita em duas etapas, análise e alteração. Na fase de análise tenta-se refinar a solução com a ajuda do usuário, que é

levado a opinar sobre os pontos críticos do projeto fornecido: ordem do filtro, largura da faixa de transição, e desvio nas faixas de passagem e corte. Uma nova tentativa de cálculo pode resultar desta análise. Se bem sucedido, ou seja, se o projeto atende à exigências do usuário, parte-se para a segunda fase do aprendizado que efetua a alteração da base de conhecimentos, introduzindo a nova regra que permitirá ao sistema um caminho alternativo quando se defrontar com situação semelhante. O SIPREX adota um procedimento de aprendizado seletivo, somente permitindo o ingresso na segunda fase do aprendizado, a usuários especiais (especialistas portadores de um código de acesso). Isto para evitar que um não especialista transmita algum tipo de informação que não contribua (ou pelo contrário, comprometa) o refinamento da solução.

9.2 Estado Atual do Sistema

No Brasil, os computadores do tipo PC/DOS ainda são a ferramenta utilizada em grande parte dos trabalhos de CAE (*Computer Aided Engineering*) realizados em centros de pesquisa e desenvolvimento bem como em indústrias. Os baixos custos de aquisição e manutenção desse tipo de máquinas, aliados à grande quantidade de software disponível no mercado e ao contínuo surgimento de novos processadores mais rápidos garantem a sua permanência no cenário técnico/científico nacional ainda por algum tempo. Essa constatação motivou o desenvolvimento da versão original do SIPREX para essa classe de computadores, mais especificamente para os PC-AT 386/387 e 486, devido ao tempo de computação exigido pelos algoritmos empregados.

Atualmente o SIPREX está com todos os seus módulos implementados, devendo brevemente iniciar-se o processo de validação do sistema com casos reais, e com uma clientela diversificada. Para tanto, versões do SIPREX deverão ser cedidas a alguns centros de pesquisa. Um processo de acompanhamento será realizado, de tal modo que o retorno de informações possa levar ao refinamento do sistema.

Entretanto, limitações foram constatadas na aplicação de alguns algoritmos em situações específicas, o que implicará na necessidade de revisão dos mesmos, a fim de superá-las. São elas:

- O algoritmo que efetua a equalização de fase de um filtro projetado pelo sistema, somente é capaz de fazê-la sem problemas, se o filtro em questão for do tipo passa-baixa. Noutro tipo de filtro, alguns coeficientes calculados ultrapassam a precisão da máquina utilizada, ocasionando erros de *underflow* e *overflow*.

  Uma solução possível para isto é transformar o projeto inicial num projeto passa-baixa. Mas ainda permanece uma limitação, pois isto não é possível para todos os tipos de filtros, como por exemplo, filtros multifaixas.

- O algoritmo de estimativa para o tamanho de um filtro FIR somente é eficiente para filtros passa-baixa, passa-alta, passa- faixa simétrico e rejeita-faixa simétrico.

Para os demais (diferenciadores, transformadores de Hilbert, multifaixas, por exemplo) o valor estimado é ainda bastante impreciso. Vale a pena salientar que a estimativa é feita levando-se em conta fórmulas empíricas, com baixa fundamentação teórica [37].

- Somente é possível projetar-se filtros recursivos do tipo seletivo em freqüência clássico, passa-baixa, passa-alta, passa-faixa e rejeita-faixa. Embora o programa IIR2 que implementa o algoritmo de Deczky [33] seja capaz de fornecer outros tipos de projetos (diferenciador, multifaixas e passa-tudo), exige do usuário uma primeira aproximação para os polos e zeros do filtro desejado, o que não é tarefa trivial, requerendo um dispendioso processo de tentativas e erros.

  A solução para este problema é a incorporação ao módulo numérico do SIPREX de um outro algoritmo com domínio de aplicação semelhante ao algoritmo de Deczky, que embora não produza um projeto final tão eficiente, permita obter o conjunto de polos e zeros que seja o ponto de partida do processo de otimização utilizado pelo programa IIR2. Atualmente o programa IIR1 desempenha este papel, embora restrito a um domínio menor de aplicação.

O sistema SIPREX foi desenvolvido em caracter experimental com aplicação na área de PDS como um todo e a uma área específica, Processamento de Voz. As demais áreas específicas que constam no menu inicial do sistema, Áudio, Telecomunicações e Sinais Biomédicos, ou alguma outra área, precisam ainda ser incorporadas à base de conhecimentos. Obtidas as informações sobre a aplicação de filtros a essas áreas, o processo de incorporação à base de conhecimento é facilitado pela própria estrutura celular utilizada. As novas células são facilmente adicionadas à base sem implicar numa reestruturação do sistema.

A tabela 9.1 fornece uma estimativa do número de linhas e o espaço em disco, em número de bytes, ocupado por cada um dos módulos atuais do SIPREX.

9.3 Perspectiva de Trabalhos Futuros

Embora o SIPREX represente um grande avanço em relação ao estado de arte em termos de ferramentas de CAD para processamento digital de sinais e também na área de sistemas acoplados, acredita-se que ele seja apenas o ponto de partida para investigação de diversos aspectos e questões que foram levantadas durante a sua concepção, implementação e avaliação. Entre os pontos levantados, em torno dos quais futuros trabalhos serão desenvolvidos, destacam-se os seguintes:

Tabela 9.1: O *Tamanho* do SIPREX

Módulo	Linhas de Programa Fonte	Linguagem de Implementação	Programa Executável em Kbytes
Controle	2.100	C	200
Interface	1.900	C	95
Explanação	530	C	93
Aprendizado	430	PROLOG	359
Central	950	PROLOG	359
Numérico	11.600	FORTRAN	700
Gráfico	1.700	C	149
TOTAL	19.210	—	1.705

a) Maior Abrangência da Interface

Investigar o uso de uma linguagem abrangente para especificação de um projeto, que embora seja direcionado ao processamento digital de sinais, se aproxime de uma linguagem natural, de tal modo que o usuário possa de uma maneira livre, descrever as funções e o domínio de aplicação do projeto pretendido. Baseado nessas informações o sistema seria capaz de fornecer os parâmetros de especificação do projeto, como a determinação do tipo de filtro (FIR ou IIR), determinação da faixa de passagem e de corte, características de fase, etc., que fosse capaz de satisfazer a aplicação desejada. Isso exigiria o uso de uma interface inteligente de um ponto de vista mais amplo, dotada de um dicionário de palavras e termos chaves, sendo ainda apoiada por uma base de conhecimentos abrangendo um universo maior da área de PDS. O resultado imediato disto, seria o de tornar o sistema mais poderoso, atendendo a um número maior de usuários, mesmos aqueles com baixo grau de especialização na área de aplicação desejada.

Num nível mais próximo do estado atual, poder-se-ia dotar a interface de capacidade gráfica, de maneira a permitir que o usuário possa definir o projeto através do gráfico da resposta em freqüência do filtro pretendido.

b) Reformulação e Ampliação da Base de Conhecimentos

Dotar o sistema de maleabilidade nas especificações dos problemas, como sugerida na primeira parte da seção anterior, implicaria na ampliação do universo de conhecimentos a ser representado na base de conhecimentos. O tipo de informação a ser armazenada poderia acarretar uma modificação na forma de representação utilizada.

A estrutura celular da base de conhecimentos do SIPREX, em que cada célula contém informações relativas a uma determinada área de PDS, facilita a expansão do

domínio do sistema com o acréscimo de novas células com informações sobre outras áreas de PDS, como Áudio, Telecomunicação, Biomédica, etc.

c) Ampliação do Módulo Numérico

Para tanto, novos algoritmos referentes a identificação de sistemas e projeto de controladores deverão ser incorporados ao Módulo Numérico, visando atender as necessidades da área de Controle Automático de Processos. Logicamente é necessário introduzir células na base de conhecimentos relativas aos novos algoritmos e a área de Controle de Processos.

d) Alteração do Módulo de Controle

O que é natural se se pretende modificar o sistema com a introdução de qualquer uma das modificações citadas.

e) Geração Automática de Código para Processamento de Sinais

A idéia deste trabalho é dotar o SIPREX de uma interface compiladora, capaz de gerar o código necessário para implementação do filtro projetado pelo sistema em qualquer das famílias de processadores de sinais existentes, como o TMS 320, DSP 5600 e AT&T DSP 32. Como complemento, deverá também ser desenvolvida uma interface em hardware que permita o carregamento e execução do código gerado no processador escolhido. Uma evolução natural deste trabalho, seria a realização de testes em tempo real por estes processadores e a utilização do resultado destes testes no processo decisório do SIPREX, para obtenção do melhor projeto final.

f) Elaboração de uma Versão do SIPREX para Ambiente UNIX

O SIPREX, como já justificado na seção anterior, foi desenvolvido para ambiente MS-DOS, mais precisamente para os PC-AT 386/387 e 486, devido ao tempo de computação exigido pelos algoritmos de projeto, e a cultura PC/DOS existente no país. Entretanto, nos centros de pesquisa dos países mais avançados, o papel de ferramenta computacional básica já foi assumido há algum tempo pelas estações de trabalho de 32 bits com sistema operacional UNIX, capacidade gráfica e processadores de alta velocidade. No caso específico do SIPREX, para se obter um projeto de um filtro otimizado, pode ser necessário a ativação de até oito algoritmos de projeto e das rotinas para análise dos resultados fornecidos pelos mesmos, o que implica em um alto custo computacional. Como exemplo, um programa de otimização de um filtro IIR de quarta ordem requer cerca de seis milhões de operações, as quais em um computador PC-AT/386 de 33 MHz consomem um tempo médio de 15 minutos. Este valor é demasiadamente alto para sistemas interativos, como o SIPREX, onde o tempo de resposta a cada decisão do usuário é um fator muito crítico, já que uma longa espera pode acarretar perda de atenção ou de interesse por parte do mesmo [40].

Para se atingir o nível de desempenho ideal para o SIPREX, com um tempo de resposta médio entre 1 e 2 minutos é necessária a utilização de uma máquina com as mesmas características das estações de trabalho mencionadas acima. Um exemplo é a estação SPARC 2 da Sun Microsystems, equipada com relógio de 40MHz, arquitetura RISC, processador de 28.5 MIPS e processador aritmético de 4.2 MFLOPS. Em relação ao AT-386/33MHz, este sistema permitiria uma redução no tempo de projeto e otimização do filtro IIR de quarta ordem por um fator de no mínimo 5.

Além da velocidade de processamento numérico, o uso de estações de trabalho do tipo SPARC como base para o SIPREX permitirá uma maior sofisticação na interface homem-máquina utilizada, deste modo tornando o sistema mais acessível a uma grande faixa de usuários [41].

Adicionalmente, devido aos poderosos recursos gráficos oferecidos pelas estações tipo SPARC, ficará mais fácil apresentar visualmente para exame do usuário os resultados de projetos realizados pelo SIPREX, através de curvas de resposta em freqüência, diagramas de blocos e de testes que simulem as condições a serem encontradas pelos filtros quando em operação real.

Portanto, o desenvolvimento de uma versão do SIPREX para estações de trabalho de 32 bits com sistema operacional compatível com UNIX, resultará em uma ferramenta de trabalho poderosa e versátil, com desempenho equivalente ao estado de arte em termos de sistemas para CAE [26].

REFERÊNCIAS BIBLIOGRÁFICAS

[1] - Waterman, D. A., *A Guide to Expert Systems,* Addison-Wesley Publishing Company, Inc., USA, 1986.

[2] - Yazdani, M., Ed., *Artificial Intelligence - Principles and Applications,* Chapman and Hall Computing, London, 1986.

[3] - Kitzmiller, C. T. and Kowalik, J. S., "Symbolic and Numerical Computing in Knowledge-Based Systems", in *Coupling Symbolic and Numerical Computing in Expert Systems,* Kowalik (Ed.), Elsevier Science Publishers B. V., North-Holland, Amsterdam, 1986.

[4] - DSP Committee, IEEE ASSP Society, (Eds.), *Programs for Digital Signal Processing,* IEEE Press, New York, 1979.

[5] - Talukdar, S. N., Elfes, A., Cardozo, E., Joobbani, R., Rychener, M., e Benares, R., "A System for Distributed Problem Solving", in *Coupling Symbolic and Numerical Computing in Expert Systems,* Kowalik (Ed.), Elsevier Science Publishers B. V., North-Holland, Amsterdam, 1986.

[6] - Sueda, N., "An Expert System for Image Processing", *Image Technology and Information Display,* **17**, No. 9, pp. 19-22, 1985.

[7] - Jacobstein, N., Kitzmiller, C. T. and Kowalik, J. S., "Integrating Symbolic and Numeric Methods in Knowledge-Based Systems: Current Status, Future Prospects, Driving Events", in *Coupling Symbolic and Numerical Computing in Expert Systems, II,* Kowalik and Kitzmiller (Eds.), Elsevier Science Publisher B. V., North-Holland, Amsterdam, 1988.

[8] - Abernathy, M. F., Bernable, M. L., Chandler, L. J., Wilkins, M. G., and Wolfe, M., " A Conceptual Design of Intelligent Front-end to Scientific Application Programs", apresentado durante o *Workshop on Coupling Symbolic and Numerical Computing in Expert Systems,* Washington, 27-29 de agosto, 1985.

[9] - Favier, G., Arruda, L. V. R., Acundeger, E. e Settineri, R., "CAO et Système en Automatique et Traitement du Signal", *XIII Colloque Gretsi,* Juan-Les-Pins, 16 a 20 de setembro, 1991.

[10] - Pequeno, M. C., Hattori, M. T., Cunha, G. M., Feitosa, F. J. e Cysne, R. M., *BITAN - Manual do Usuário,* Relatório Técnico, Universidade Federal do Ceará, Departamento de Estatística e Matemática Aplicada, Campus do Pici, Caixa Postal 3013, Fortaleza, CE, 1985.

[11] - Cooley, J. W. e Tukey, J. W., "An Efficient, One-level, Primitive-equation Spectral Model", *Monthly Weather Review,* No. 100, pp. 683-689, 1965.

[12] - Neto, J .L., e Hattori, M. T., "Transformadas Rápidas de Fourier - Uma Metodologia de Comparação", *XI CNMAC*, Ouro Preto, MG, 1988.

[13] - Gentleman, W. M. e Sande, G., "Fast Fourier Transforms for Fun and Profit", 1966 Fall Joint Computer Conference, *AFIPS Proc.* **29**, pp. 563-578, 1966.

[14] - Pease, M. C., "An Adaptation of the Fast Fourier Transform for Parallel Processing", *Journal of the Association for Computing Machinery* **15**, No. 2, pp. 252-264, 1968.

[15] - Singleton, R. C., "An Algorithm for Computing the Mixed Radix Fast Fourier Transform", *IEEE Transaction Audio and Electroacustics* **AU-17**, No. 2, pp. 93-103, 1969.

[16] - Uhrich, M. L., "Fast Fourier Transforms Without Sorting", *IEEE Transactions on Audio and Electroacoustics* **AU-17**, No. 2, pp. 170-172, 1969.

[17] - Temperton, C., " Mixed Radix Fast Fourier Transforms Without Reordering", Technical Report 3, ECMWE, 1976.

[18] - Ryder, B. G., "The Pfort Verifier", *Software Practice and Experience* **4**, No. 4, pp. 359-378, 1974.

[19] - Smith, B. T., "Portability and Adaptability - What Are the Issues", in *Numerical Software - Needs and Availability*, Jacobs, D. A. H. (Ed.), Academic Press, London, pp. 21-38, 1978.

[20] - Hattori, M. T., Pequeno, M. C., Cunha, G. M., Feitosa, F. J. B. e Gomes Neto, J. B. F., "Uma Disciplina de Programação em FORTRAN, Projeto BITAN", Relatório Técnico, Universidade Federal do Ceará, Departamento de Estatística e Matemática Aplicada, Campus do Pici, Caixa Postal 3013, Fortaleza, CE, 1984.

[21] - Hattori, M. T., Pequeno, M. C., " FFT - Problemática de Testes", *XII CNMAC*, S. J. do Rio Preto, SP, 1989.

[22] - Oppenheim, A. V., e Schafer, R., W., *Digital Signal Processing*, Prentice-Hall, Inc., 1975.

[23] - Rabiner, L. R., e Gold, B., *Theory and Application of Digital Signal Processing*, Prentice Hall, Inc., 1975.

[24] - Monard, M. C., Scarpelli, H. A. C., e Rodrigues, S. R., "Implementação de uma Base de Conhecimento para Apoio a Usuários da Biblioteca NAG", *XII CNMAC*, S. José do Rio Preto, SP, 1989.

[25] - Gandelin, M. H., " Une Approche Système Expert pour la Reconnaissance Automatique d'Organismes Marins a Partir d'Images Monoculaires", Thèse de Doctorat presentée à l'Université de Nice Sophia-Antipolis, 1989.

[26] - Lee, E.A., et al., "Gabriel: A Design Environment for DSP", *IEEE Transactions on Acoustics, Speech and Signal Processing* **37**, No.11, Nov., pp. 1751-1762, 1989.

[27] - Monarch, *Digital Signal Processing Software,* Version 2.0, The Athena Group, Inc., Acte +, Montlignon, France, 1990.

[28] - Nie, L. G., Mahoudeaux, P. M. e Gaillard, P., "Application of AI Techniques to Signal Processing: A Knowledge-Based System for Digital Filters Synthesis", Signal Processing, No. 23, pp. 121-136, 1991.

[29] - Silva Filho, A. M., "Modelagem e Implementação de uma Base de Conhecimento para Projeto de Filtros Digitais", Dissertação de Mestrado, Coordenação de Pós-Graduação em Engenharia Elétrica, Universidade Federal da Paraíba, Campina Grande, PB, 1991.

[30] - Perkusich, M. L. B., "Contribuição para a Construção de Sistemas Baseados em Conhecimento Voltados para o Conhecimento Profundo", Dissertação de mestrado, Coordenação de Pós-Gradução em Engenharia Elétrica, Universidade Federal da Paraíba, Campina Grande, PB, 1990.

[31] - Schuessler, H. W., *Digitale Systeme zur Signalverarbeityng,* Spring-Verlag, Berlin, 1973.

[32] - Gold, B. e Rader, C. M., *Digital Processing of Signals,* McGraw-Hill, New York, 1969.

[33] - Deczky, A. G., "Synthesis of Recursive Digital Filters Using the Minimum-p Error Criterion", *IEEE Transactions on Audio and Electroacoustics* **AU-20**, No. 4, pp. 257-263, 1972.

[34] - Fletcher, R. e Powell, M. J. D., "A Rapidly Convergent Descent Method for Minimization", *Comput. J.* **6**, No. 2, pp. 163-168, 1963.

[35] - Hooke, R. e Jeeves, T. A., "Direct Search Solution of Numerical and Statistical Problems", *Journal of Association for Computing Machinery* **8**, No. 2, pp. 212-229, 1961.

[36] - Cheney, E. W., *Introduction to Approximation Theory,* McGraw-Hill, pp. 77-100, New York, 1966.

[37] - Herrmann, O., Rabiner, L. R. e Chan, D. S. K., "Practical Design Rules for Optimum Finite Impulse Response Low-Pass Digital Filters", *The Bell System Technical Journal* **52**, No. 6, pp. 769-799, 1973.

[38] - Parks, T. W. e McClellan, J. H., "A Program for the Design of Linear Phase Finite Impulse Response Digital Filters", IEEE Transactions on Audio and Electroacustics **AU-20**, No. 3, pp. 195-199, 1972.

[39] - Rycherner, M. D., *Expert Systems for Engineering Design,* Academic Press, New York, 1988.

[40] - Schneider, B., "Response Time and Display Rate in Humam Performance with Computers", *ACM Computing Surveys* **16**, No. 3, pp. 265-285 1984,

[41] - Wexelblat, R. L., "On Interface Requirements for Expert Systems", *AI Magazine,* Fall 1989, pp. 66-78.

APÊNDICE
UM EXEMPLO DE UTILIZAÇÃO DO SIPREX

A.1 Introdução

O problema escolhido para exemplificar a utilização do SIPREX passo a passo é o problema 1 do capítulo 8. O exemplo deverá servir como um guia de utilização do sistema. Todas as janelas exibidas pelo sistema com os respectivos questionamentos serão mostradas a seguir.

A.2 Utilização do Sistema

Quando o usuário aciona o sistema (causando a execução do programa SIPREX), uma tela de apresentação é exibida (figura A.1), para em seguida exibir a janela em que o usuário deverá definir, selecionando uma das opções apresentadas, a área de PDS em que seu problema se enquadra (figura A.2). A escolha pode ser feita movimentando-se o cursor com as teclas de movimento do cursor (setas de direção) ou fornecendo o número correspondente à opção desejada, teclando **Enter** em ambos os casos após a seleção. Note a mensagem explicativa da solicitação no rodapé. Escolhida a área de aplicação, no exemplo a área de VOZ, uma nova janela é exibida (figura A.3), desta feita solicitando a natureza do filtro, se FIR ou IIR, ou ainda se o usuário não deseja determinar a natureza, deixando que o sistema a escolha. Em seguida solicita dados para caracterizar o filtro: o tipo de filtro, o seu tamanho e quantização ou não do projeto final (figura A.4). O usuário poderá responder a estas solicitações em qualquer ordem, mas é imperativo que responda a todas elas. Para o tipo de filtro são apresentados aqueles filtros que podem ser tratados pelo sistema dos quais o usuário deve escolher o que for adequado ao seu projeto (figura A.5). Feita a escolha, no caso passa-baixa, o sistema solicita a especificação das faixas de passagem e corte, dos pesos correspondentes de cada faixa, da freqüência de amostragem e dos desvios normalizados nas faixas de passagem e corte (figura A.6). Para a especificação do tamanho do filtro uma janela é aberta solicitando ao usuário a determinação do tamanho do filtro desejado (figura A.7). Se o usuário não deseja especificar o tamanho do filtro, deverá teclar **Enter** sem definir nenhum valor; o sistema determinará o tamanho. Para a determinação do projeto final quantizado ou não, uma janela é apresentada com as possibilidades **Sim** ou **Não** (figura A.8). No caso foi respondido **Não**. Teclando sucessivos **Esc**'s as janelas são apagadas, com o sistema mostrando as etapas processadas e em processamento (figura A.9). Determinados os coeficientes do filtro, o sistema oferece ao usuário a possibilidade de ver o caminho percorrido para se chegar à solução (figura A.10). Se for o caso, janelas com os fatos e as regras que levaram à solução são exibidas (figura A.11). A seguir, o projeto obtido é fornecido (figura A.12), permitindo ao usuário, se assim o

desejar, examinar o filtro projetado pelo gráfico da sua resposta em freqüência. Para isso, o usuário deverá teclar **G**. O sistema então pergunta se o projeto fornecido é satisfatório ou não (figura A.13). No caso o projeto foi considerado insatisfatório, passa-se então à fase de tomada de decisões para modificar o projeto: modificar a natureza do filtro (no caso, modificar para um projeto IIR), diminuir o desvio (*ripple*), diminuir a largura da faixa de transição ou tentar um projeto com tamanho de filtro menor (figuras A.14 e A.15). No caso optou-se pela diminuição da largura da faixa de transição. O sistema então tenta encontrar uma nova solução. Novamente o projeto obtido é exibido e o usuário deve verificar se é satisfatório ou não (figura A.16). No caso foi julgado satisfatório e o processamento foi encerrado (figura A.17).

S I P R E X

Sistema Especialista para Projeto de Filtros Digitais

E Q U I P E

Mauro C. Pequeno — Ligia Perkusich

Joao M. Carvalho — Antonio M. S. Filho

Mario T. Hattori — Adriano F. Q. Brito

Laboratorio de Automação e Processamento de Sinais

DEE - CCT - UFPb - CAMPUS II

Caixa Postal 10105 - 58100 Campina Grande - Pb - Brasil

Figura A.1: Tela de Apresentação

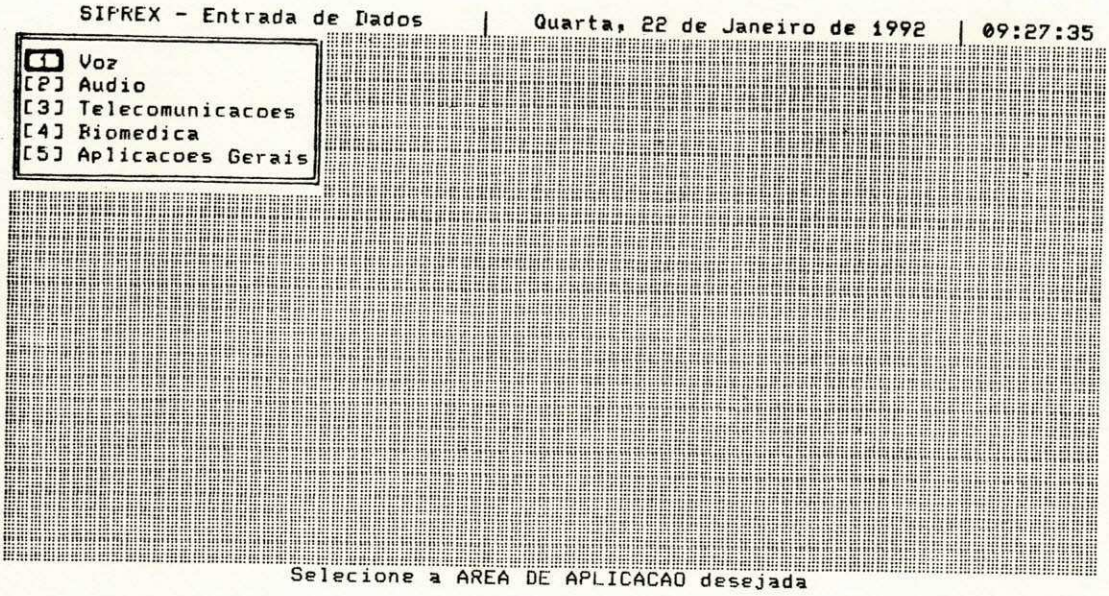

Figura A.2: Janela para Escolha da Área de Aplicação

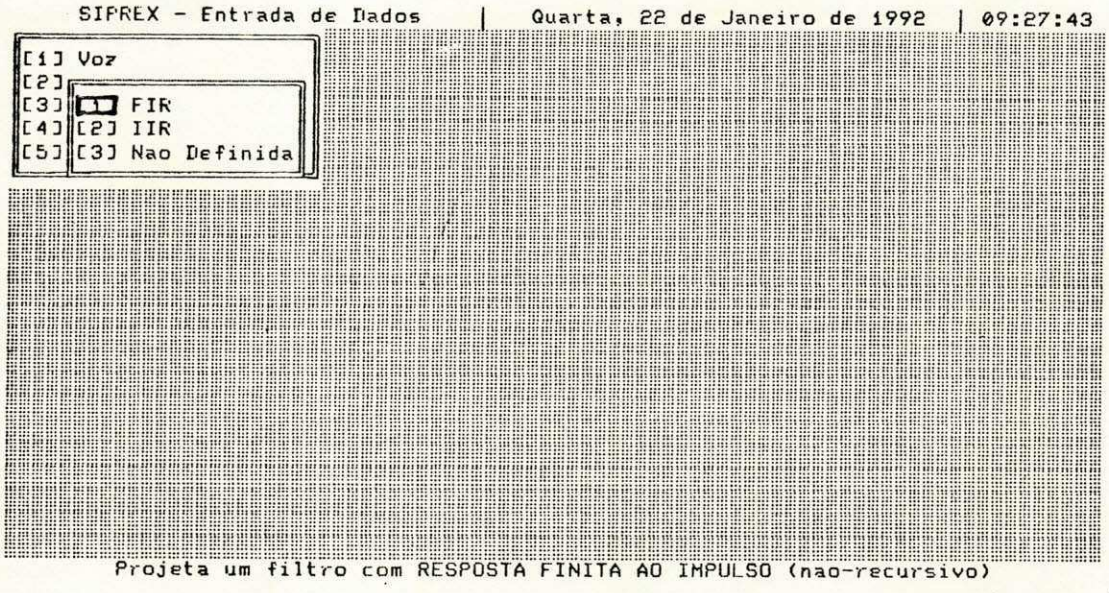

Figura A.3: Determinação da Natureza do Filtro

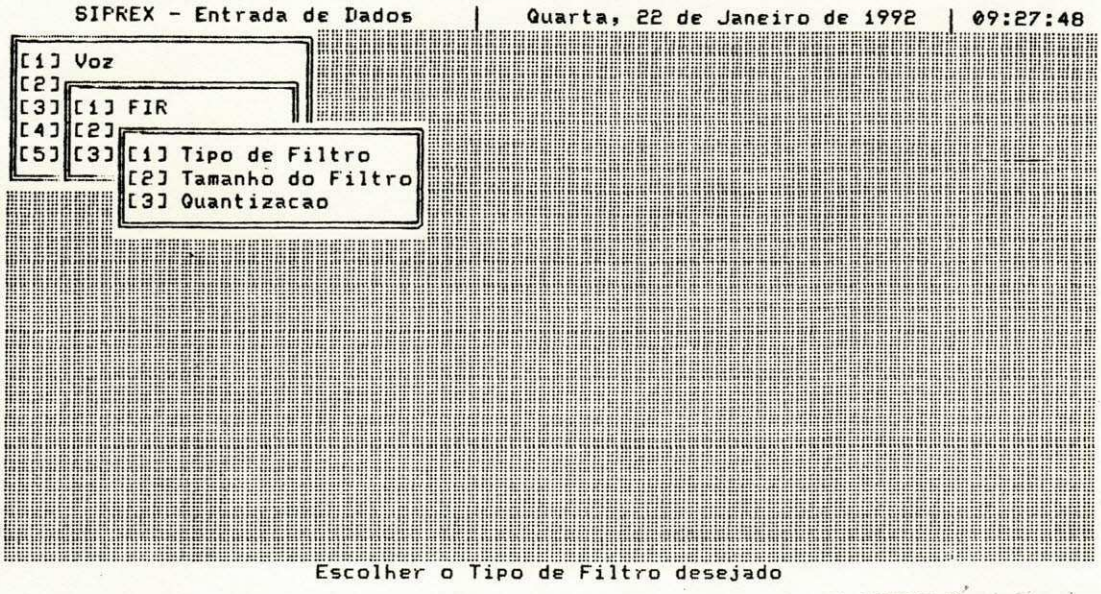

Figura A.4: Janela para Especificação do Filtro

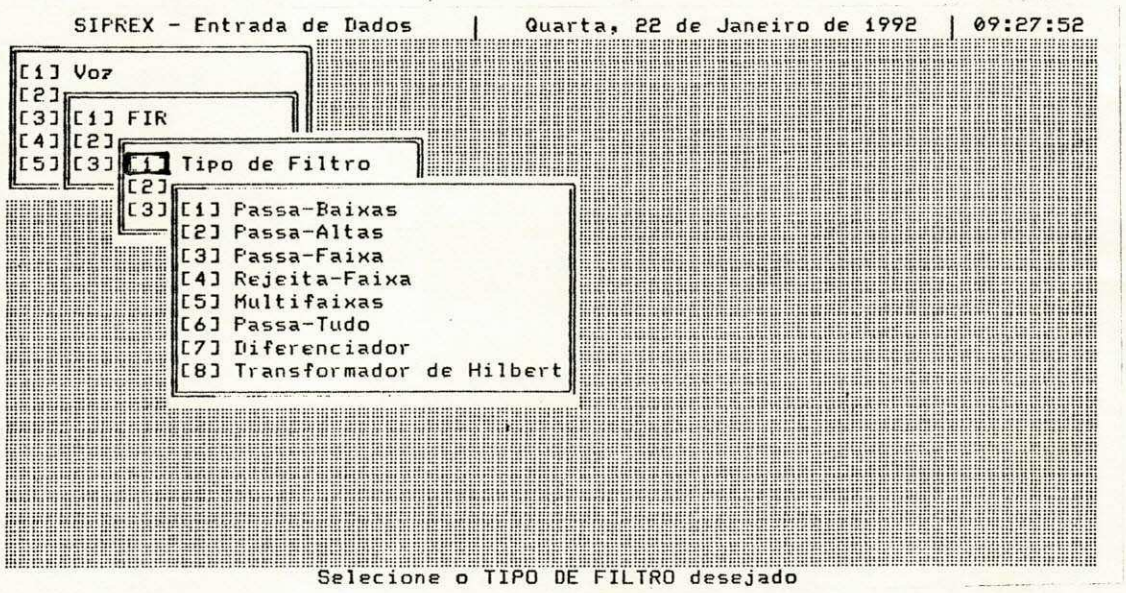

Figura A.5: Janela para Escolha do Tipo de Filtro

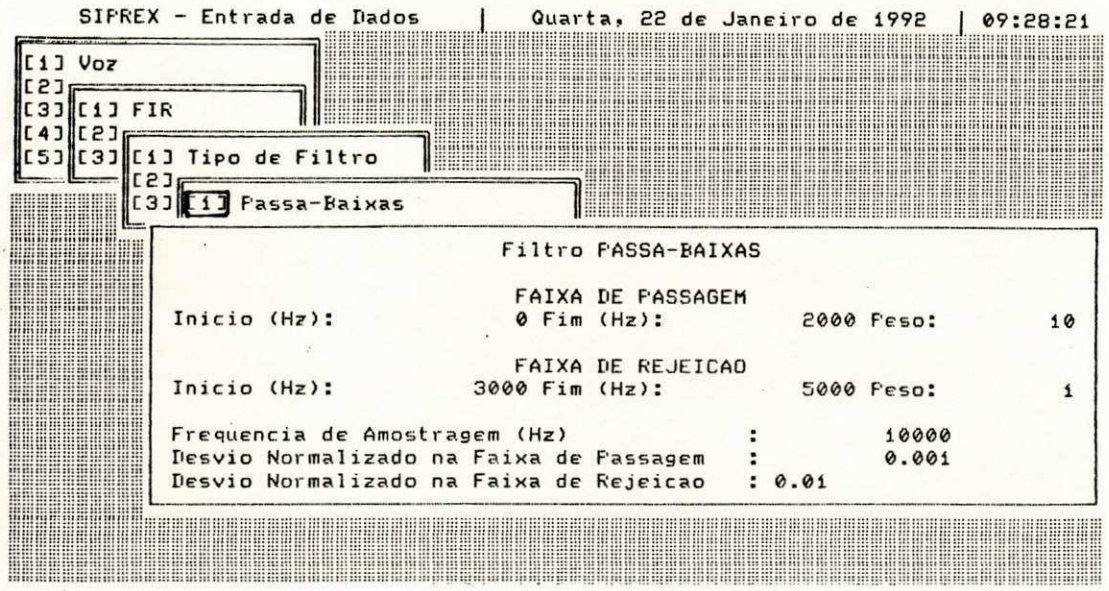

Figura A.6: Determinação das Faixas, Freqüência de Amostragem, Pesos e Desvios

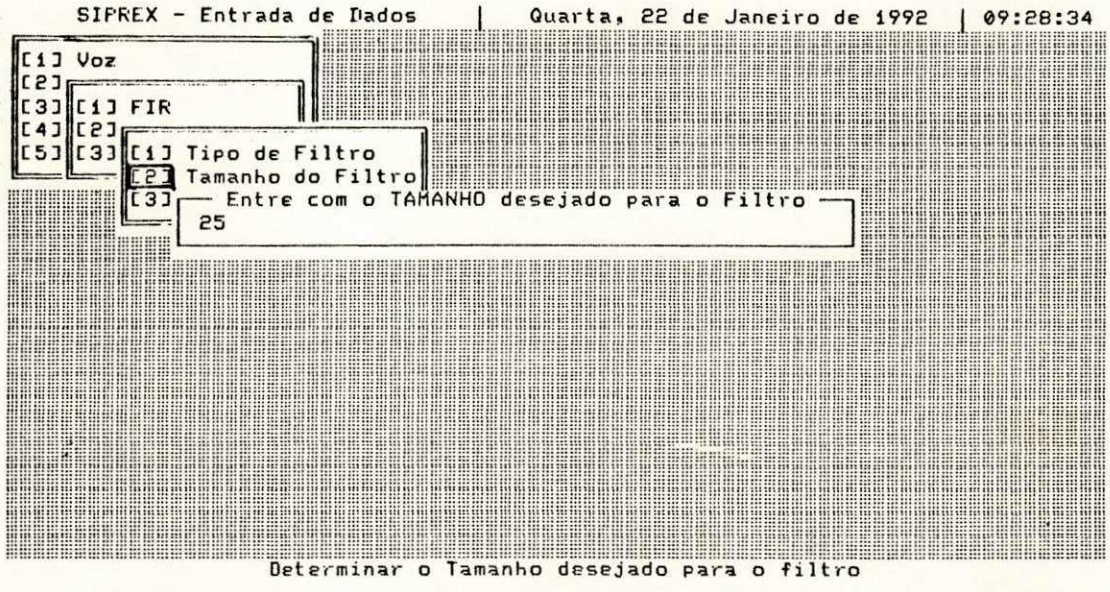

Figura A.7: Determinação do Tamanho do Filtro

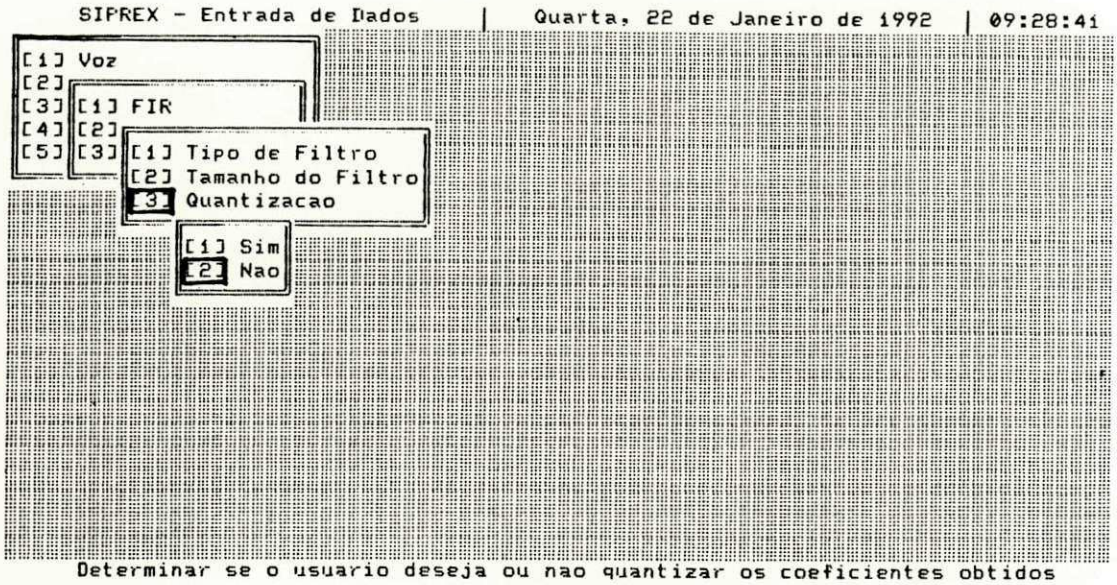

Figura A.8: Janela para Opção de Quantização

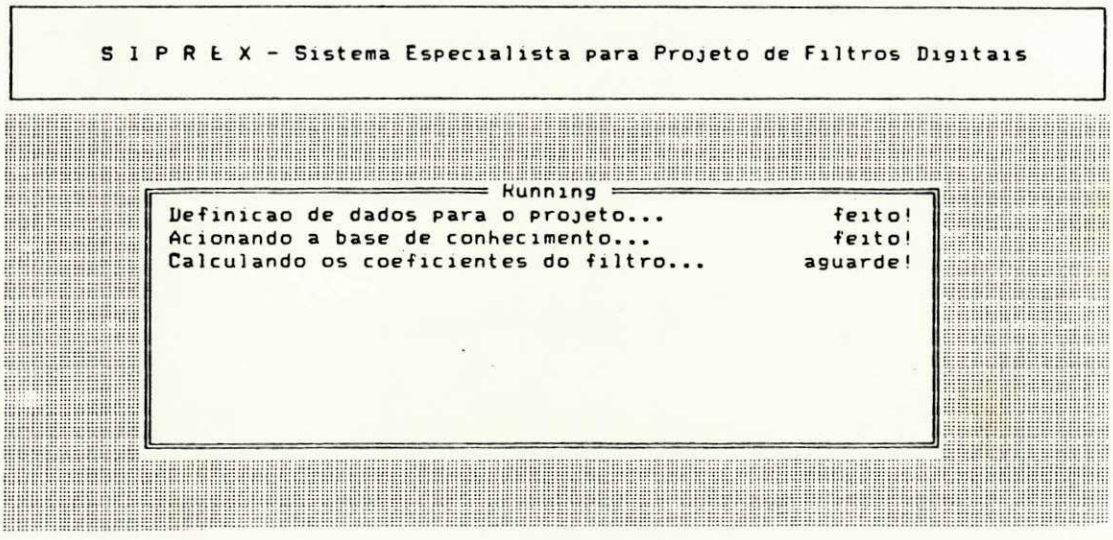

Figura A.9: Janela Indicativa de Processamento

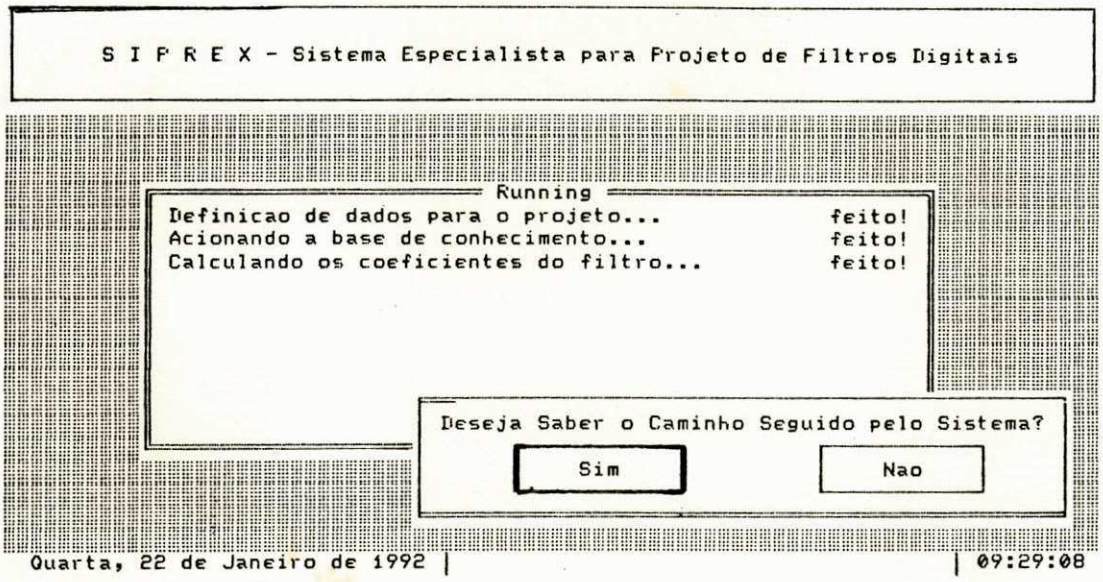

Figura A.10: Opção de Explanação

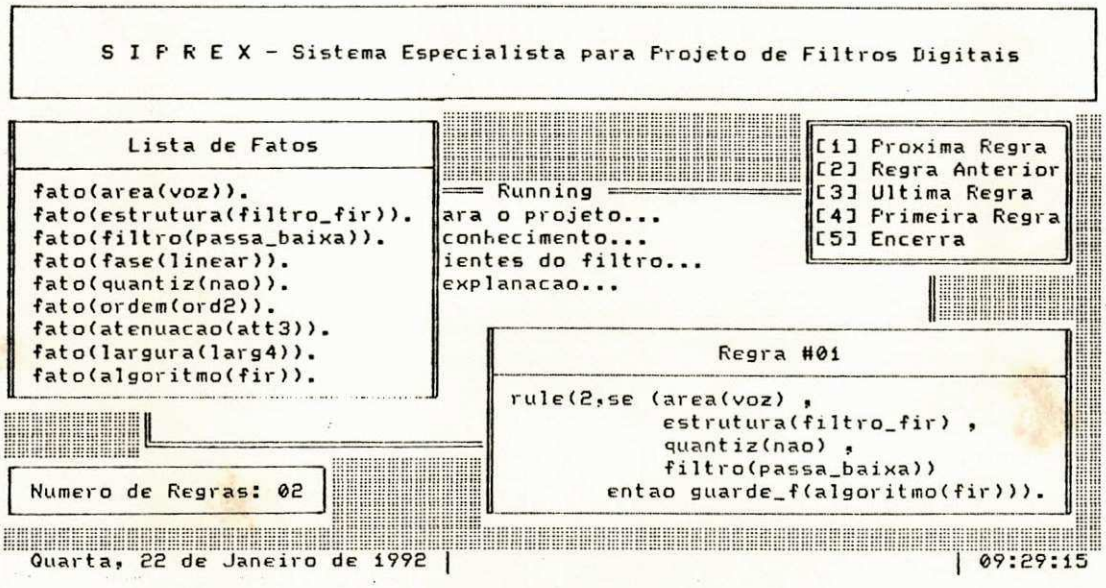

Figura A.11: Janela de Explanação

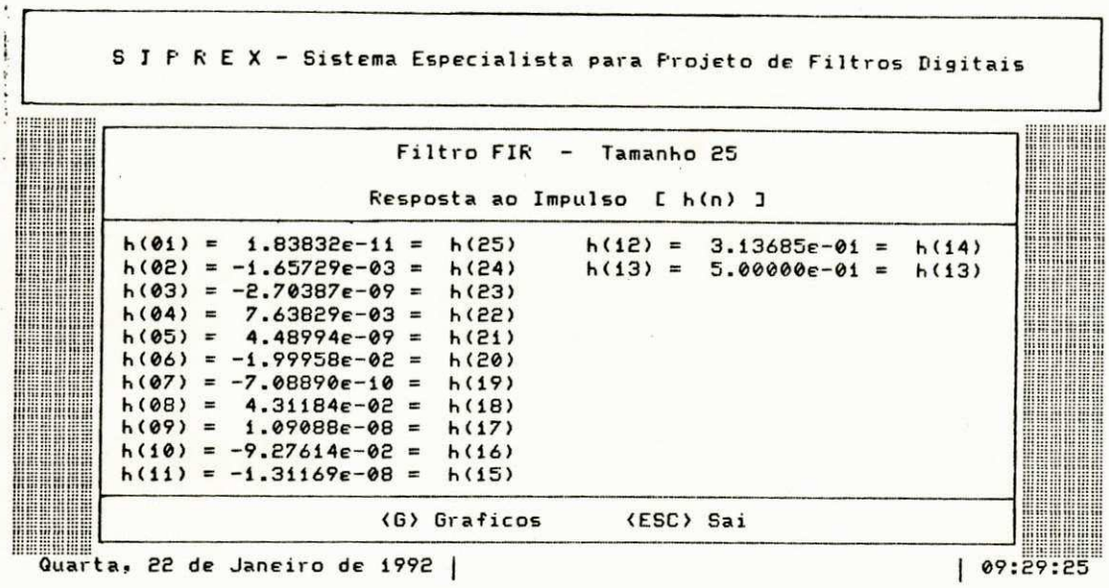

Figura A.12: Coeficientes do Filtro Projetado

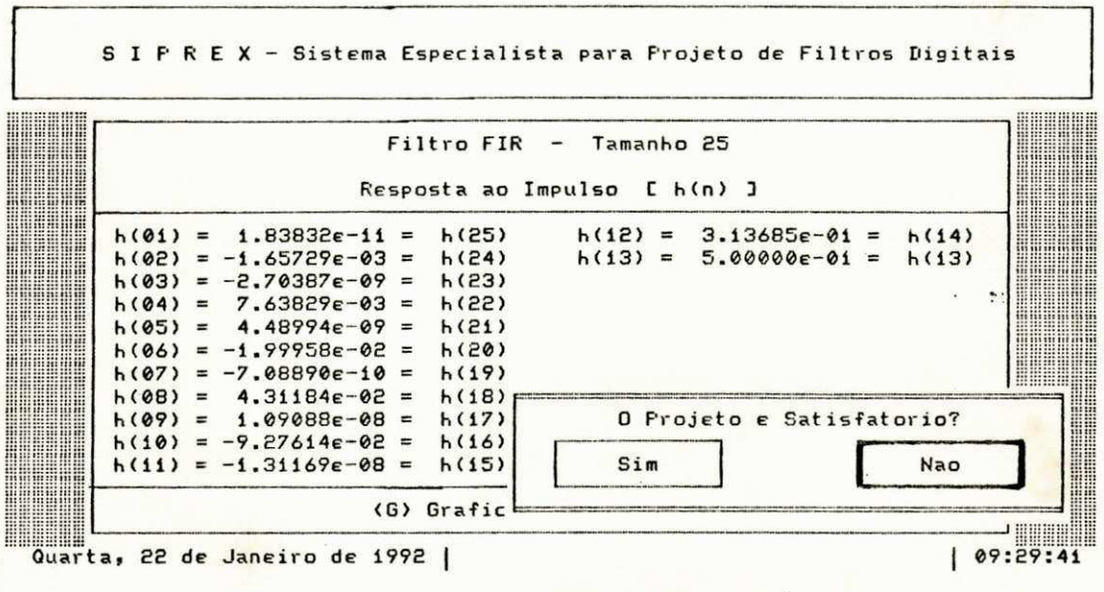

Figura A.13: Usuário Opina sobre Sucesso do Projeto

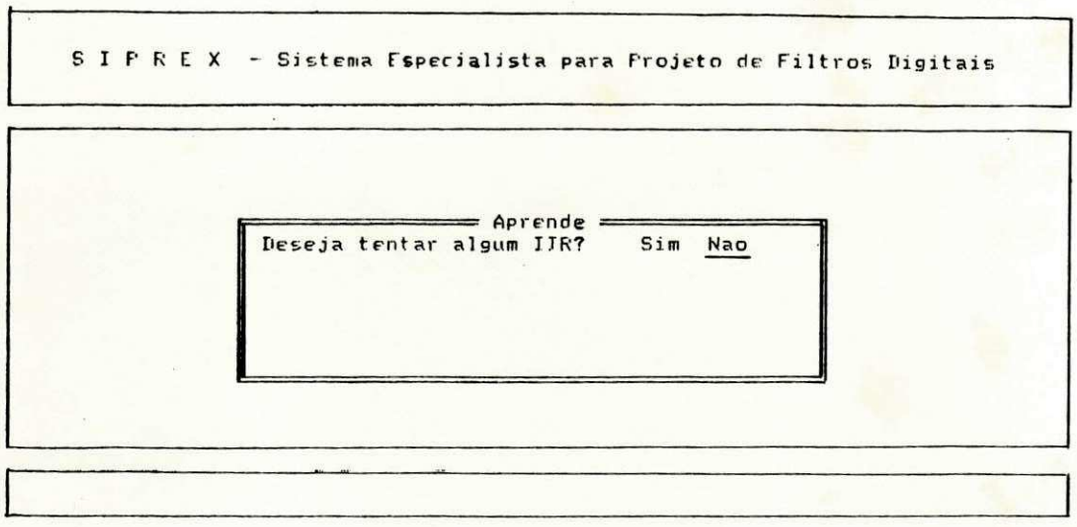

Figura A.14: Busca de Nova Solução

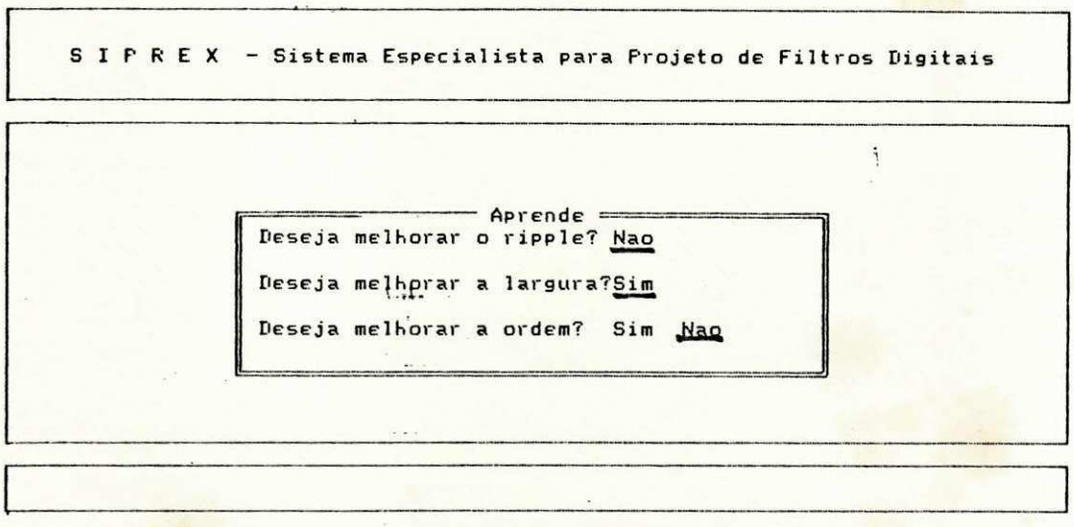

Figura A.15: Análise do Projeto

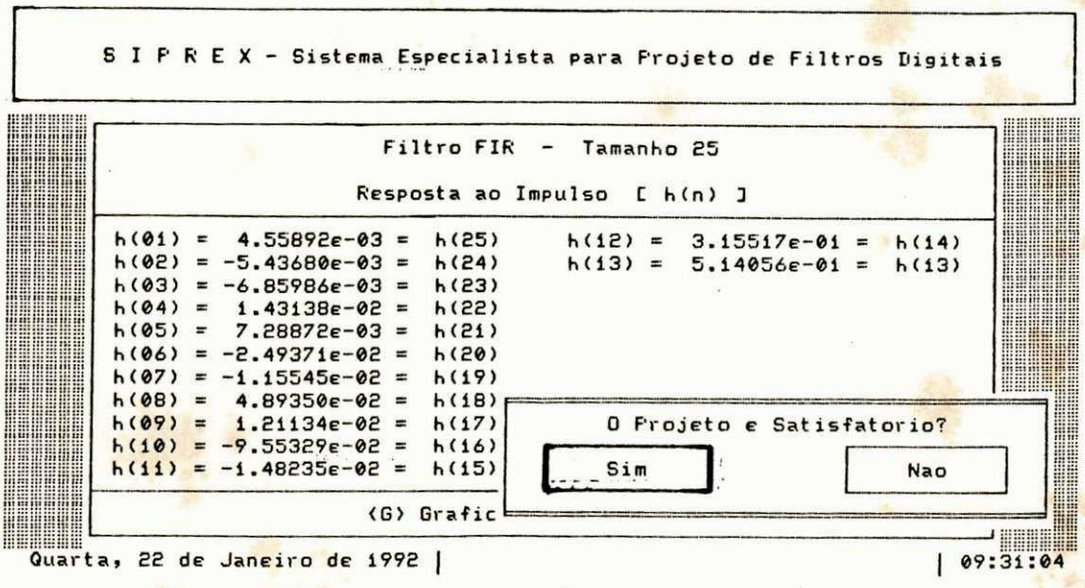

Figura A.16: Novo Projeto

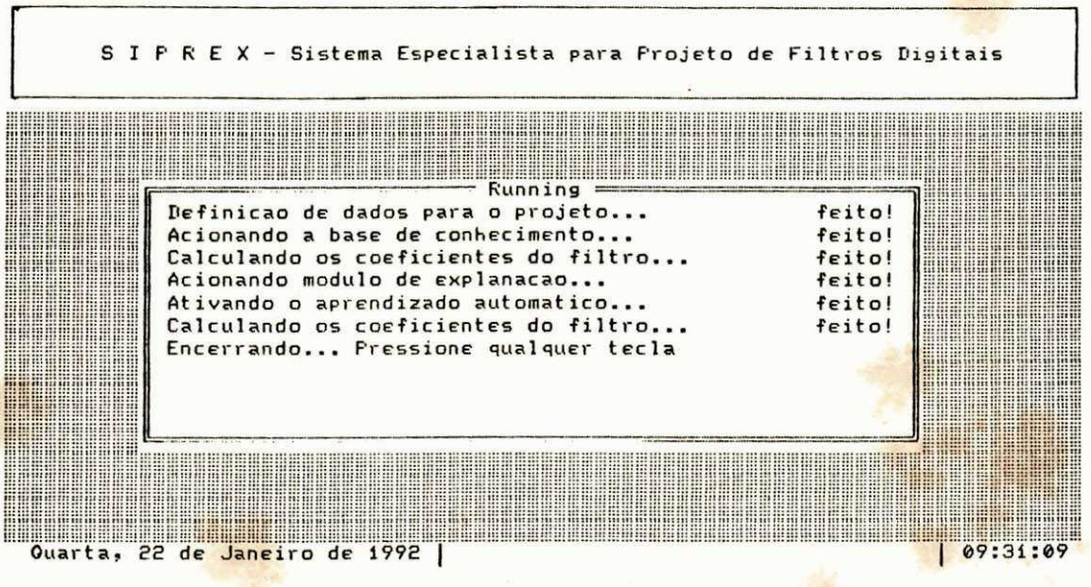

Figura A.17: Encerramento