UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

PROJECTO E ANÁLISE DE FILTROS DIGITAIS

(Relatório de Estágio PRODEP)

VALTER JOSÉ PROENÇA HENRIQUES

PORTO, 1993

Universidade do Porto
Faculdade de Engenharia
Biblioteca
Nº
CDU 621.3/043.3)/LEEC/1992/HENV
Data 01/10/2009

Parecer sobre o estágio PRODEP 4.3 realizado pelo aluno da FEUP Valter José Proença Henriques

O aluno da FEUP ***Valter José Proença Henriques*** foi por mim orientado no estágio que realizou no INESC no âmbito do PRODEP 4.3.

O estágio incluiu a utilização do "package" FDS ("Filter Design System"), que é uma ferramenta avançada de análise e projecto de filtros em computador.

O trabalho do aluno Valter foi de excelente qualidade. Criou blocos novos (de que são exemplo a introdução de pólos e zeros e o bloco "Doctor Filter"), revelando grande originalidade de concepção.

O relatório de estágio, bem escrito, inclui ainda um excelente "tutorial" sobre filtros digitais.

Em face do trabalho desenvolvido sou de opinião que o aluno Valter tem grandes potencialidades e que valeu muito a pena ter realizado este estágio nesta instituição.

INESC, 3 de Dezembro de 1993

Prof. Eurico Carrapatoso
(Responsável técnico pelo estágio)

Parecer sobre o estágio PRODEP 4.3 realizado pelo aluno Valter José Proença Henriques

O aluno finalista ***Valter José Proença Henriques*** realizou um estágio no INESC no âmbito do PRODEP 4.3.

O processamento digital de sinal (em particular filtros digitais) foi o tema dominante do estágio, que consistiu na utilização do "package" FDS ("Filter Design System").

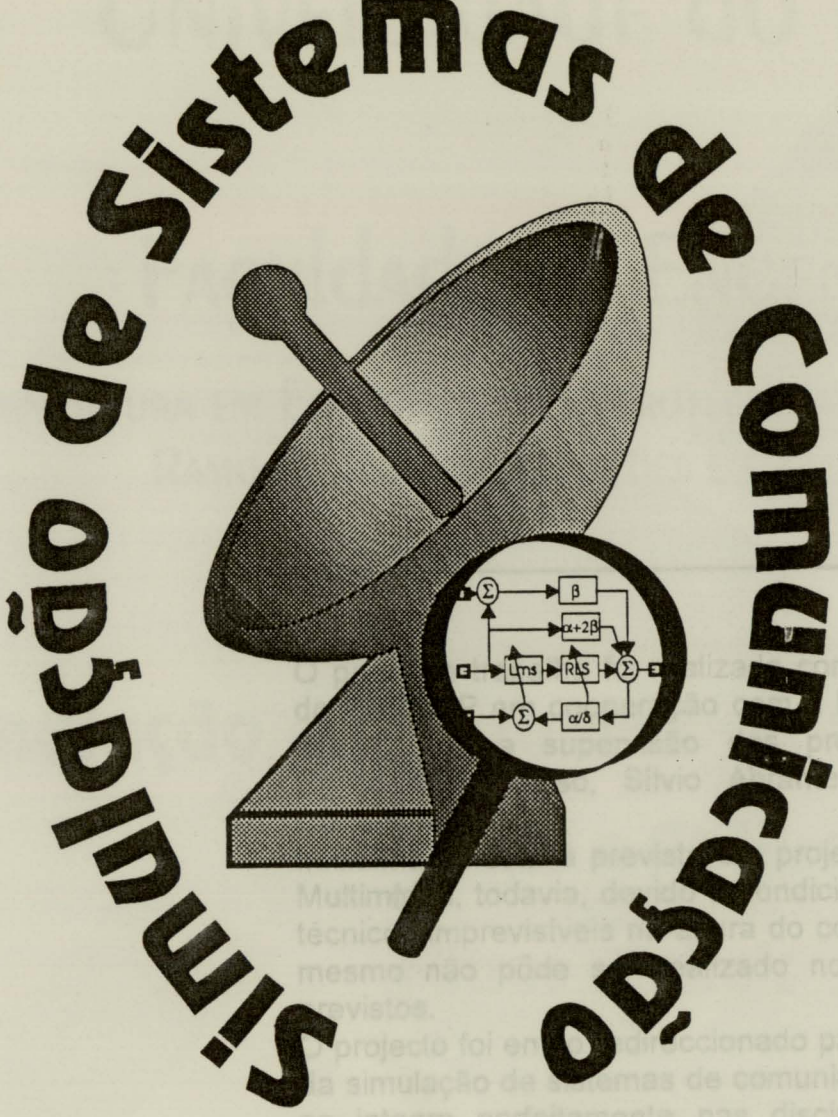
O "package" FDS é uma "ferramenta" pedagógica excelente. Daí que o Relatório de estágio produzido revele uma assinalável consolidação dos conhecimentos anteriormente adquiridos na disciplina de "Processamento de Sinal" (4ª Ano da Licenciatura).

O Relatório revela ainda algum trabalho original, como o bloco "Doctor Filter", e inclui uma exposição ("tutorial") sobre filtros digitais.

Parece-me assim que os objectivos de índole didáctico-pedagógica do estágio foram plenamente conseguidos pelo aluno Valter Henriques.

FEUP, 3 de Dezembro de 1993

Eng. Luís Corte-Real
(Responsável científico pelo estágio)



Projecto e Análise de Filtros Digitais

Signal Processing WorkSystem SPW™

Nota:

O presente trabalho foi realizado com o apoio do PRODEP em cooperação com a FEUP e o INESC sob a supervisão dos professores: Eurico Carrapatoso, Sílvio Abrantes e Luís Côrte-Real.

Inicialmente estava previsto um projecto sobre Multimédia, todavia, devido a condicionalismos técnicos imprevisíveis na altura do concurso, o mesmo não pôde ser realizado nos moldes previstos.

O projecto foi então redireccionado para a área da simulação de sistemas de comunicação que se integra perfeitamente nas disciplinas em investigação por parte destes docentes.

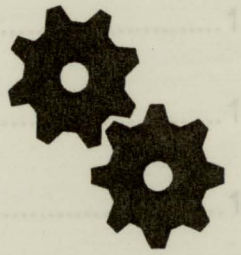
Universidade do Porto

Faculdade de Engenharia

Licenciatura em Engenharia Electrotécnica e de Computadores
Ramo de Telecomunicações e Computadores

Simulação de Sistemas de Comunicação

Projecto e Análise de Filtros Digitais



Valter José Proença Henriques

Trabalho coordenado
pelo professor
Sílvio Abrantes

Porto, 1993

1. ÍNDICE

1. Índice 1

2. Prefácio 3

3. Introdução 3

3.1. Breve Descrição do SPW 4

3.2. Breve Descrição do FDS 5

4. Filtros Digitais 6

4.1. Filtros IIR 9

4.2. Filtros FIR 9

5. Representação de Filtros no FDS 10

5.1. Forma Polinomial (POLY) 11

5.2. Forma de Segunda Ordem (SOS) 11

6. O Ambiente de Trabalho 12

6.1. Área de Projecto 13

6.2. Área dos Coeficientes 13

6.3. Área de Resposta em Frequência 13

6.4. Área de Resposta Impulsional 14

6.5. Área do Mapeamento Pólos-Zeros 14

6.6. Menus e Área de Comando 14

7. Métodos de Concepção de Filtros Digitais 15
7.1. Síntese de Filtros Digitais 15
7.2. Métodos Standard do FDS 16
7.2.1. Métodos para Filtros IIR 16
7.2.1.1. Método Bessel 16
7.2.1.2. Método Butterworth 17
7.2.1.3. Método Chebychev I 17
7.2.1.4. Método Chebychev II 17
7.2.1.5. Métodos Elípticos 18
7.2.1.6. Métodos SDE 18
7.2.1.7. Resumo dos Métodos IIR 18
7.2.2. Métodos para filtros FIR 19
7.2.2.1. Método Equiripple 19
7.2.2.2. Métodos de Janela (Windowed_sinc Methods) 20
7.2.2.3. Métodos SDE 20
7.2.2.4. Resumo dos Métodos FIR 20
7.3. Métodos Definidos pelo Utilizador 21
7.3.1. Procedimentos Gerais 21
7.3.1.1. O Ficheiro ".params" 22
7.3.1.2. O Ficheiro ".C" ou ".F" 22
7.3.1.3. O Ficheiro ".flags" 22
7.3.2. Exemplos de Aplicação 22
7.3.2.1. Exemplo 1: MANUAL 23
7.3.2.2. Exemplo 2: CARTESOIDE 23
7.3.2.2.1. Pólos-Zeros versus Resposta em Frequência 23
7.3.2.2.2. Solução Adoptada 24
7.3.2.2.3. Exemplos de Utilização 25
7.4. Filtros versus Ficheiros 27
8. Utilização de Filtros "FDS" no SDE 28
9. Utilização de Filtros "FDS" no BDE 28
9.1. O Doctor Filter 29
10. Resumo 30
10.1. A versão 3.0 31
11. Bibliografia 31
12. ANEXOS 32

2. PREFÁCIO

O presente trabalho visa dar a conhecer as potencialidades dum *package* de *software*, o **FDS**[1], cuja finalidade é assistir o engenheiro nas fases de concepção e análise dum determinado filtro digital (quer **IIR**[2], quer **FIR**[3]). Esta aplicação pode ser utilizada de uma forma autónoma ou em conjunto com o ambiente de projecto que é o **SPW**. Será este último caso, o alvo privilegiado da nossa atenção.

Assim sendo, iniciaremos este relatório com uma breve descrição da envolvente contextual do **FDS**, propositadamente reduzida pois interessa apenas focar o modo como o *Filter Design System* interage com os seus correligionários.

Seguidamente, apresentaremos alguns dos fundamentos que regem os filtros digitais, nomeadamente os tipos e seus campos de aplicação. A teoria subjacente à explanação poderá ser encontrada em qualquer livro de processamento digital de sinal, pelo nos limitaremos a focar apenas os aspectos práticos em questão.

Em sequência com o raciocínio anterior, estudaremos a forma como o **FDS** representa essas realidades tecnológicas, bem como do ambiente de trabalho e potencialidades disponibilizadas pelo sistema.

O núcleo do trabalho será dedicado à descrição dos métodos de concepção de filtros digitais usados nesta aplicação, bem como à forma de criar novos métodos definidos pelo utilizador. Como corolário, serão apresentados alguns exemplos a propósito deste tema.

Finalmente, e dado que no nosso caso o **FDS** é um dos módulos do **SPW**[4], debruçaremo-nos sobre os métodos disponíveis para proceder à troca de informação entre as diversas aplicações. Mais concretamente, a utilização de filtros recém criados no **SDE**[5] e, é claro, no **BDE**[6].

3. INTRODUÇÃO

Antes de nos concentrarmos na síntese e análise de filtros com base nesta nova e poderosa ferramenta, vamos apresentar (de forma muito sucinta) a envolvente contextual deste trabalho.

Começaremos pelo ambiente mais abrangente, o **SPW**, localizando de seguida um seu componente, o **FDS**, que é no fundo o alvo deste relatório.

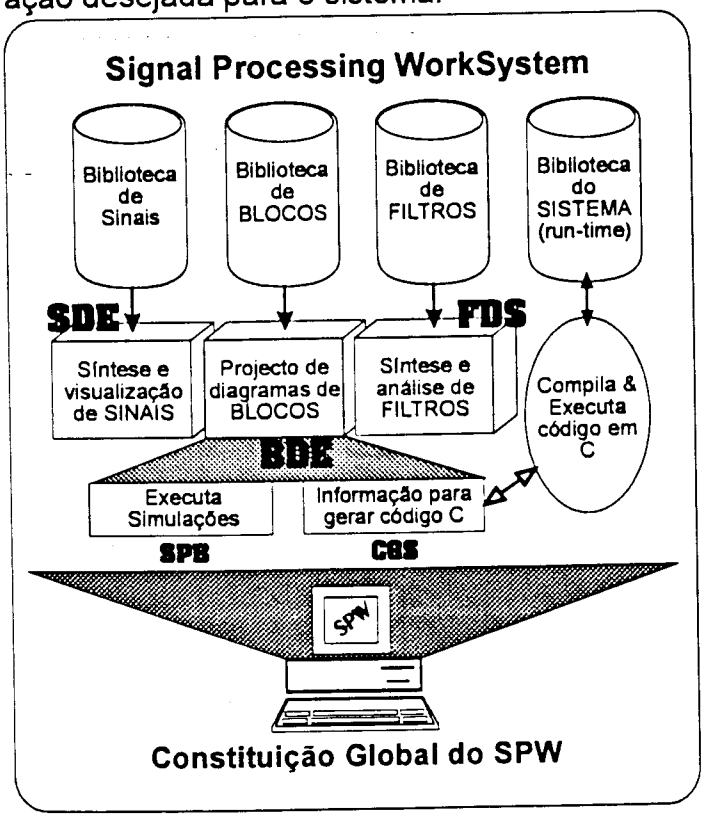
---

[1]FDS: *Filter Design System*; versão 2.7/FDS1000
[2]IIR: *Infinite-Impulse Response*
[3]FIR: *Finite-Impulse Response*
[4]SPW: *Signal Processing Work system;* versão 2.8.2.
[5]SDE: *Signal Display Editor*
[6]BDE: *Block Diagram Editor*

3.1. Breve Descrição do SPW

Como referimos anteriormente, o **SPW** é um *package* de *software* bastante complexo vocacionado para a concepção e simulação de sistemas de comunicações, cuja "linguagem" tem por base uma estrutura hierárquica, construída quer à custa de sub-blocos funcionais, quer em blocos em que se concebeu um algoritmo dedicado, normalmente em C. Trata-se de uma ferramenta bastante poderosa, pois liberta o projectista das tarefas menos criativas, permitindo-lhe uma maior dedicação à fase de concepção de um sistema descrito por um diagrama de blocos, os quais podem ser construídos de base, ou simplesmente retirados de uma vasta biblioteca que contém os módulos mais frequentes na área do processamento de sinal. Existe também a possibilidade de criar uma paleta de blocos, definidos ou não pelo utilizador, facilitando bastante a recolha das unidades mais utilizadas.

O **SPW** em si mesmo, não é um programa, mas antes um conjunto de componentes que podem ser utilizados isoladamente e interligados sempre que se desejar. Por esta razão existem alguns módulos cuja instalação depende da configuração desejada para o sistema.



Constituição Global do SPW

Todavia, podemos considerar como fulcrais o **BDE** e o **SDE**. No primeiro, para além de se poderem conceber e simular sistemas, engloba também os módulos **CGS**[7] e o **HDS**[8]. O **SDE** permite analisar e gerar sinais e comunica com o **BDE** ou directamente (no caso de se acompanhar o evoluir de uma simulação) ou por intermédio de ficheiros. O **FDS** é um bloco à margem vocacionado para o projecto de filtros, gerando ficheiros compatíveis com o resto do sistema.

Dados os objectivos deste pequeno item introdutório, remetemos para a figura acima o modo de como interagem estes blocos num todo, que é o **SPW**.

[7]CGS: *Code Generation System*
[8]HDS: *Hardware Design System*

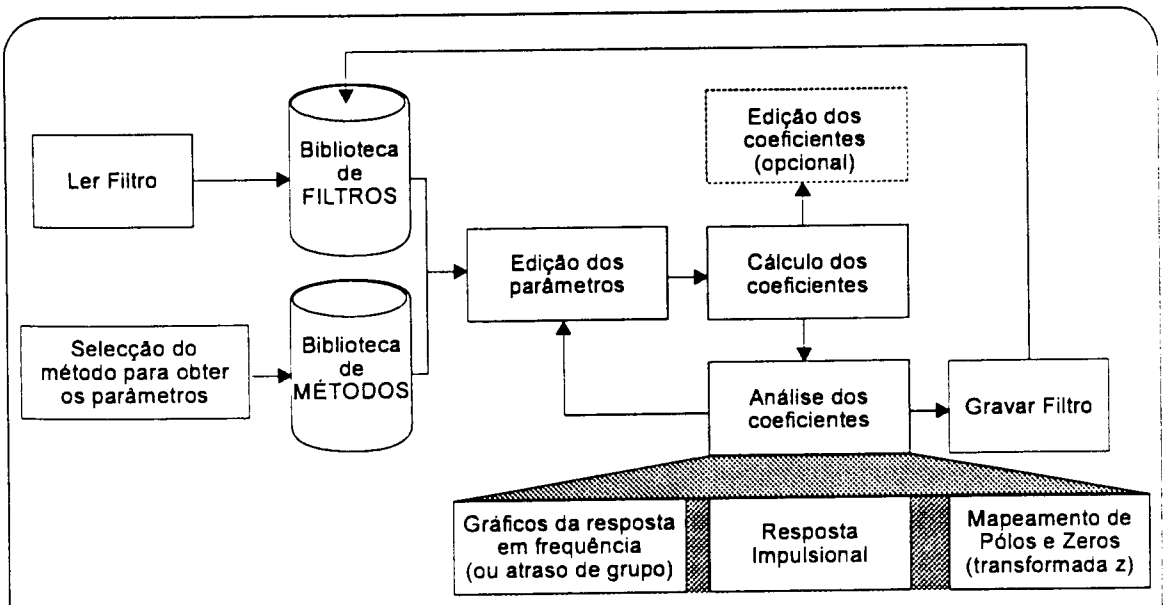
3.2. BREVE DESCRIÇÃO DO FDS

Como referimos anteriormente, o **FDS** é um módulo algo independente do resto do sistema, permitindo a concepção e análise interactiva de filtros **IIR** e **FIR** digitais.

Estão disponíveis diversos métodos de projecto de filtros, escolhendo-se um deles consoante os dados considerados mais relevantes, nomeadamente: frequências de corte, número de coeficientes, tipo de resposta, etc. Caso a aplicação seja mais restrita, o utilizador poderá conceber um método de projecto de filtros, podendo faze-lo quer em C standard, quer em *Fortran*.

As ferramentas de análise mais comuns como sejam a resposta em frequência, a resposta impulsional e os diagramas de pólos e zeros (ou seja, a transformada z do filtro), estão implementadas nesta aplicação permitindo um total controlo sobre as características do filtro em estudo.

Os filtros poderão ser gravados juntamente com toda a informação referida no parágrafo anterior, ou simplesmente no formato ASCII podendo assim serem utilizados pelos restantes blocos do **SPW**, nomeadamente o **BDE** e o **SDE**. Os coeficientes que assim são preservados, são utilizados no **FDS** quer na sua forma polinomial (POLY), quer na forma de coeficientes de segunda ordem (SOS), como é apanágio da teoria dos sistemas. Este assunto merecerá a nossa atenção num dos futuros pontos.



Esquema Funcional do FDS

O esquema funcional interno do **FDS**, encontra-se representado na figura acima, sendo que cada um dos seus blocos terá a sua merecida explicação no ponto mais adequado.

4. FILTROS DIGITAIS

Um filtro digital é, na sua essência, um sistema que implementa um determinado algoritmo, em que os dados de entrada são constituídos pelas amostras passadas de um sinal (ou também futuras, em aplicações em que se dispõe de muita informação simultaneamente, como é o caso do vídeo) e, opcionalmente, por amostras da própria saída do filtro. Essas amostras são multiplicadas por coeficientes que, juntamente com a configuração do filtro, estão intimamente relacionados com uma equação às diferenças, ou resposta impulsional, ou transformada z que se pretende implementar.

Podemos considerar que a saída dum filtro pode ser descrita por uma equação convolucional:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k) \cdot x(n-k) \qquad (1)$$

Uma outra representação possível é a da equação às diferenças a qual, dada a sua natureza, constitui uma das formas mais comuns de representar os filtros pois, os coeficientes duma determinada configuração (dita *Direct Form I*) aparecem explicitamente na expressão:

$$y(n) = \sum_{k=1}^{M} a_k \cdot y(n-k) + \sum_{k=-N_f}^{N_p} b_k \cdot x(n-k) \qquad (2)$$

Por outro lado, se pretendermos que o nosso sistema tenha uma determinada transformada z, temos duas formas clássicas. A primeira é a chamada "soma dos produtos":

$$H(z) = \frac{\sum_{k=-N_f}^{N_p} b_k z^{-k}}{1 - \sum_{k=1}^{M} a_k z^{-k}} \qquad (3)$$

A segunda é a do "produto dos termos":

$$H(z) = A \cdot z^{N_f} \frac{\prod_{k=1}^{N_p+N_f} (1 - c_k z^{-1})}{\prod_{k=1}^{M} (1 - d_k z^{-1})} \qquad (4)$$

Neste último caso teremos:

- Zeros em $z=c_k$ para $1 \le k \le N_p+N_f$, e N_p+N_f pólos em $z=0$.
- Pólos em $z=d_k$ para $1 \le k \le M$, e M zeros em $z=0$.
- N_f pólos em $z=\infty$ e N_f zeros em $z=0$.

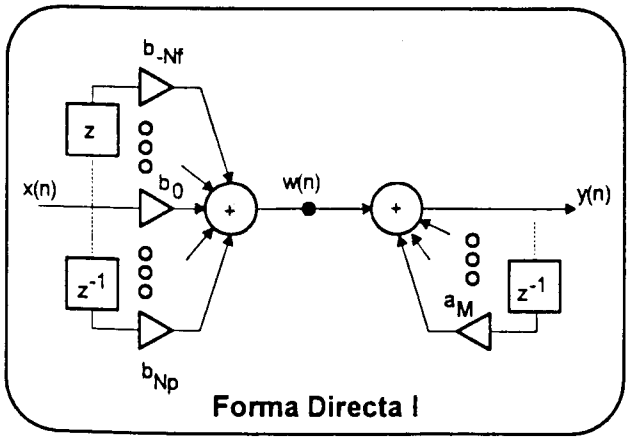
(o cancelamento pólo/zero ocorrerá normalmente em $z=0$).

Os filtros digitais podem ser caracterizados quer pela duração da sua resposta impulsional, quer pela sua estrutura. Assim, se um filtro produz uma saída de duração infinita, quando um impulso real (ou seja, de amplitude e durações limitadas) é colocado à sua entrada, designamo-lo por **IIR** (*infinite-impulse response*). Se tal não suceder, estamos na presença do chamado filtro **FIR** (*finite-impulse response*).

Quanto à sua estrutura, os filtros podem ser denominados por recursivos ou não recursivos. Se a saída depender, entre outros factores, de valores anteriores dessa mesma saída, o filtro diz-se recursivo (existe pelo menos uma malha de *feedback*). Dissemos anteriormente que as equações que apresentámos estavam directamente ligadas à implementação dos filtros, e um bom exemplo disso é que podemos saber se um filtro é recursivo ou não por simples inspecção do valor dos seus coeficientes. Assim, se nas equações (2) e (3) existir pelo menos um a_k não nulo, então existe pelo menos uma malha de *feedback*. O mesmo se passa se pelo menos um dos d_k na equação (4) for diferente de zero. É óbvio que se nenhuma das condições anteriores se verificar é porque estamos na presença de um sistema não recursivo. Um filtro deste tipo diz-se que tem uma transformada z constituída só por zeros pois os pólos não são relevantes (ou estão em z=0 ou em z=∞, não contribuindo para a magnitude da resposta).

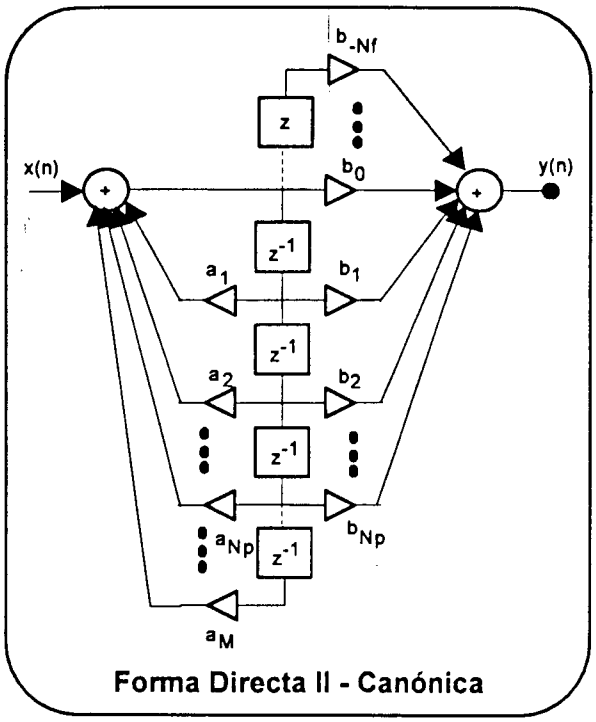
Para que a resposta de um sistema tenha uma duração infinita, é necessário que exista uma estrutura recursiva (para que haja um sinal na entrada) pelo que os termos "**IIR**" e "recursivo" costumam designar uma mesma propriedade. Analogamente, o mesmo se passa com os termos "**FIR**" e "não-recursivo", embora neste caso existam algumas excepções.

A forma mais expedita de passar directamente das equações à implementação prática (e por isso mesmo designada por *Direct Form I*) está representada na próxima figura.



Forma Directa I

Apesar das enormes vantagens no que diz respeito à transição do modelo matemático para o diagrama de blocos que esta estrutura possibilita, nem sempre esta é a opção acertada pois não é assegurada utilização mais eficiente de recursos, nomeadamente em termos de memória. Todavia, se exprimirmos as funções de transferência em relação ao ponto *w(n)*, considerando o filtro como dois sistemas em cascata, e após algum tratamento matemático, encontramos outra configuração, que curiosamente se designa por *Direct Form II*, representada na figura da página seguinte.



Forma Directa II - Canónica

Esta estrutura utiliza o menor número de unidades atrasadoras requeridas para implementar uma determinada função de transferência, sendo por isso conhecida por forma canónica. Um caso muito importante surge quando N_f é nulo e M=2, pois essa configuração pode ser agrupada em cascata ou em paralelo constituindo uma forma prática de conceber filtros de ordem elevada. Estamos a falar das secções de segunda ordem (SOS).

Apresentamos em seguida, e de forma muita reduzida, as equações de que falamos, bem como da estrutura a que dão origem.

Assim, o primeiro passo é decompor *H(z)* na forma:

$$H(z) = A \cdot z^{N_f} \cdot \prod_{k=1}^{(\frac{M}{2})} H_k(z) \qquad (5)$$

onde,

$$H_k(z) = \frac{1 + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}} \qquad (6)$$

- *A* é o ganho.
- z^{Nf} é a componente não causal em avanço.

[M/2] é o menor inteiro superior a M/2. Se $N_p+N_f>M$, então o limite superior do produtório será calculado com base em $[(N_p+N_f)/2]$.

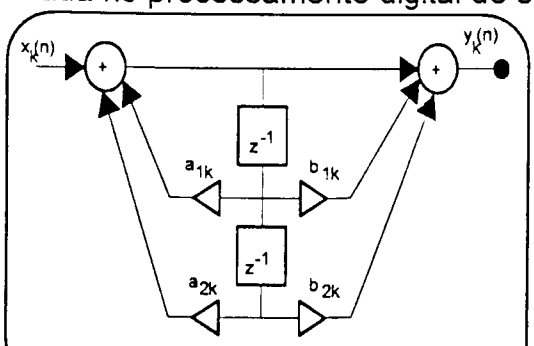
Em termos temporais,

$$y_k(n) = a_{1k} y_k(n-1) + a_{2k} y_k(n-2) + x_k(n) + b_{1k} x_k(n-1) + b_{2k} x_k(n-2) \quad (7)$$

No domínio z,

$$Y_k(z) = a_{1k} z^{-1} Y_k(z) + a_{2k} z^{-2} Y_k(z) + X_k(z) + b_{1k} z^{-1} X_k(z) + b_{2k} z^{-2} X_k(z) \quad (8)$$

Finalmente, apresenta-se a configuração da secção canónica, bastante usada no processamento digital de sinal (e também no **FDS**).



Estrutura de Segunda Ordem (SOS)

De facto, normalmente concentra-se a maior parte do esforço projectista na concepção de uma implementação eficiente desta estrutura, pois um filtro pode ser decomposto numa associação em cascata destas unidades. Mais ainda, os coeficientes do filtro podem ser pode ser acedidos de uma memória, o que possibilita que uma nova configuração pode ser facilmente conseguida por simples alteração do conteúdo dos registos. Por outro lado, como o sistema

é constituído por um número mais reduzido de componentes, também existem melhorias em termos de tempo de processamento.

No caso geral, e aquando da necessidade de implementar um filtro que esteja de acordo com determinadas especificações, o projectista é confrontado com três situações típicas:

1. Escolha do tipo de filtro (**IIR** e **FIR**).
2. Determinação dos coeficientes adequados à especificação.
3. Verificação do respeito dessa mesma especificação (no caso negativo, deve voltar ao ponto anterior - processo interactivo).

A primeira decisão é de fulcral importância, pelo que lhe dedicaremos alguma da nossa atenção.

4.1. FILTROS *IIR*

Os filtros de resposta de duração infinita foram os primeiros a surgir pois são extensões naturais dos seus homólogos analógicos. Não é por acaso que são normalmente os eleitos quando se pretende substituir filtros analógicos por digitais. Naturalmente, as técnicas para a sua síntese, baseiam-se em procedimentos clássicos tendo por base a transformada de *Laplace*, a qual é usada como passo intermédio, pois o produto final é descrito através da transformada z (transição em que se faz uso das transformações bilineares). Os métodos mais comuns para conceber filtros passa-baixo são: *Butterworth*, *Chebychev* e técnicas elípticas. Se se pretenderem filtros de outro tipo (nomeadamente passa-alto, passa-banda e rejeição de banda) é necessário empreender algumas transformações no domínio das frequências. Por vezes encontram-se associações de filtros **IIR** com filtros "passa-tudo", que tem por função "linearizar" a resposta em fase na banda passante.

Voltando aos procedimentos de síntese, podemos dizer que a técnica defendida por *Chebychev* é a mais popular pois enquanto que o método de *Butterworth* (apesar de simples) conduz a filtros de ordem elevada, o elíptico (proporciona os filtros de mais baixa ordem para a mesma especificação) exige um apurado esforço analítico. Escusado será dizer que, com o advento do **FDS**, o problema da complexidade do projecto não se coloca.

4.2. FILTROS *FIR*

Os filtros de resposta impulsional de duração finita, apresentam à partida uma enorme vantagem em relação aos seus correligionários pois a sua resposta em frequência pode ter um perfil qualquer. Por outro lado, também se pode forçar um filtro a ter uma resposta em fase exactamente linear com a frequência, o que também possibilita a síntese de estruturas com um reduzido (metade) número de coeficientes. Contudo, e como não há filtro sem senão, o tipo **IIR** é bastante mais competitivo quando o tempo de processamento e os recursos de memória são factores preponderantes.

Existem basicamente três processos típicos para a síntese de filtros **FIR**:

- Método da janela.
- Método da DFT (*DFT/windowing method*).
- Método da amostragem em frequência.

O primeiro é o mais adequado quando se conhece exactamente o perfil desejado para a sua resposta impulsional. É usual serem utilizados como substitutos dos seus homólogos analógicos.

A transformada de *Fourier* discreta é utilizada na segunda técnica em que a resposta impulsional do filtro é determinada à sua custa, e também com o uso de janelas (rectangulares, triangulares...) que tem por função permitir o uso de "vectores" truncados sem que as especificações sejam desrespeitadas.

A terceira, aproveita directamente as amostras da resposta em frequência que se pretende, e a sua estrutura é constituída por duas secções: um *comb filter* e uma bateria de *parallel resonators*. É o método mais poderoso quando se pretendem sintetizar filtros com um perfil arbitrário.

Uma outra técnica que não foi citada (isto porque foge um pouco ao conceito clássico de "filtro"), consiste em efectuar as operações no domínio das frequências, o que teoricamente permite atingir qualquer especificação que se coloque. Não nos adiantaremos mais sobre esta alternativa, pois o seu ambiente natural será o **BDE** e não o **FDS**.

5. REPRESENTAÇÃO DE FILTROS NO *FDS*

Após este pequeno intrelúdio sobre o reino dos filtros digitais, impõe-se que nos debrucemos sobre a forma de como esse universo é tratado no seu "palácio" natural: o **FDS**.

Para facilidade de compreensão (e também para adoptar a notação usada no **FDS**), repetimos aqui a equação às diferenças que se pretende implementar:

$$y(n) = \frac{\sum_{k=0}^{M-1} b(k)x(n-k) - \sum_{k=1}^{N-1} a(k)y(n-k)}{a(0)} \qquad (9)$$

Estamos na presença de um filtro **IIR**. Se se desejar um de resposta finita, basta colocar todos os a(k) a zero, com excepção de a(0) que normalmente é unitário. Para o **FDS** esta representação é irrelevante pois o importante é conhecer a transformada z do filtro. Esta poderá assumir uma de duas formas: a polinomial (POLY); e a de secções de segunda ordem em cascata (SOS).

5.1. FORMA POLINOMIAL (*POLY*)

No primeiro caso, a transformação é imediata pois não é necessário recalcular os coeficientes:

$$H(z)=\frac{b(0)+b(1)z^{-1}+...+b(M-1)z^{-(M-1)}}{a(0)+a(1)z^{-1}+...+a(N-1)z^{-(N-1)}} \qquad (10)$$

De notar que os b(k) são designados por coeficientes do numerador e os a(k) por coeficientes do denominador (isto aquando da apresentação de resultados). A ordem do filtro pode ser determinada tomando o maior dos valores de N e M.

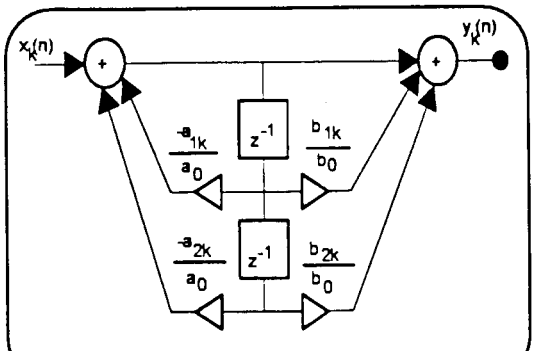
5.2. FORMA DE SEGUNDA ORDEM (*SOS*)

Já nos referimos às diversas virtudes do uso da associação em cascata de secções de segunda ordem, todavia existe ainda outro ponto importante por focar. Com efeito, os filtros digitais são normalmente implementados com **DSP**'s cujos multiplicadores tem uma precisão finita (obviamente), o que origina problemas de *overflow* e arredondamento.

A minimização destes contratempos poderá ser conseguida com a associação em cascata de que falamos. Neste caso, a equação descritiva no domínio z será:

$$H(z)=\frac{b_1(0)z^2+b_1(1)z+b_1(2)}{a_1(0)z^2+a_1(1)z+a_1(2)}\times\frac{b_2(0)z^2+b_2(1)z+b_2(2)}{a_2(0)z^2+a_2(1)z+a_2(2)}\cdots\frac{b_L(0)z^2+b_L(1)z+b_L(2)}{a_L(0)z^2+a_L(1)z+a_L(2)} \qquad (11)$$

Existem portanto, *L* secções de segunda ordem, sendo que os índices indicam a que secção dizem respeito.



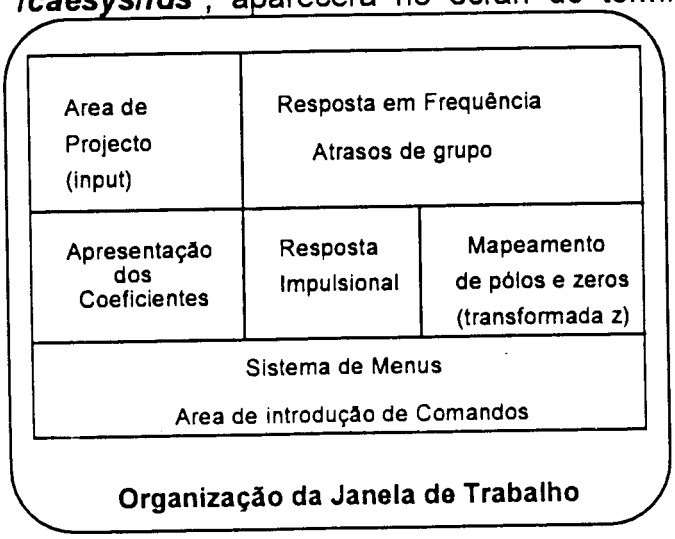
A primeira Secção SOS

O **FDS** permite a permutação entre as duas formas (opção ***EDIT COEFS CONVERT***) de um modo completamente transparente para o utilizador. Para além disso, e a propósito dos problemas da precisão limitada, o **FDS** possibilita também a "simulação" desse fenómeno pois podemos atribuir uma determinada precisão aos coeficientes (numa opção em tudo análoga à anterior).

Fazendo uma analogia com o estudo precedente, podemos apresentar uma das possíveis estruturas para uma das secções da cascata. Assim consideramos apenas a primeira na figura acima.

6. O AMBIENTE DE TRABALHO

Após a entrada no sistema (ou seja, no *Newton*), e a digitação do comando "***/caesys/fds***", aparecerá no écran do terminal um sistema de janelas que irão constituir o nosso ambiente de trabalho. A sua fiel representação pode ser encontrada no anexo a este relatório. Todavia, podemos apresentá-lo aqui também, se bem que de uma forma esquemática, como é o caso da figura apensa.



Organização da Janela de Trabalho

No topo, e discretamente, encontram-se algumas informações muito úteis naquela que é designada por *Title Line*. Nesta barra estão discriminadas, para além da versão do *package*, as bibliotecas a que o **FDS** irá aceder sempre que necessário. Existem três destas bibliotecas (*libraries*):

- Biblioteca de Filtros.
- Biblioteca de Métodos.
- Biblioteca de Sinais.

A primeira deverá ser criada pelo utilizador (usando um programa próprio para o efeito), que deverá indicar ao sistema que a deseja utilizar como *default*. Nela serão preservados os filtros que se vão sintetizando, por forma a poderem ser utilizados mais tarde quer pelo mesmo projectista, quer por outro elemento do grupo de trabalho.

Como será detalhado posteriormente, o engenheiro poderá conceber filtros segundo um método seu (isto apesar de a versão original lhe colocar à disposição uma enorme panóplia de possibilidades). Estes "algoritmos" também poderão ser registados numa biblioteca própria, em tudo análoga à que nos referimos no parágrafo anterior.

Embora o **FDS** não permita processar sinais, se for instalado como um módulo do **SPW**, este poderá comunicar com o **SDE** que é uma excelente ferramenta para se efectuarem testes sobre o filtro incipiente. Surge assim a necessidade de se indicar ao sistema qual a biblioteca que deverá usar (que deverá ser a mesma no dois módulos). Assim, o **FDS** pode comunicar com o **SDE** através de ficheiros que tem um formato compatível com os dois interessados. Todavia, esta não é a única forma de comunicação entre as duas aplicações pois, o **SDE** disponibiliza uma opção em que se pode utilizar o filtro em uso no momento no seu vizinho.

Passamos agora a uma breve digressão pelas diversas áreas da janela, a qual nunca se deverá substituir a um conhecimento muito mais sólido que se adquire pela sua utilização directa (através do *tutorial*, por exemplo).

6.1. ÁREA DE PROJECTO

Esta é a zona reservada à introdução de parâmetros. De facto, após a selecção do método de síntese, surgem nesta secção três tipos distintos de informação: indicação das bibliotecas do filtro e do método; um espaço reservado para comentários (editável); a terceira área é a dos parâmetros propriamente ditos acompanhados dos seus valores (editáveis) de *default* e de comentários (editáveis).

A alteração de dados é efectuada directamente no local (com o rato e teclado), após o surge, apensa ao local, a expressão "*Modified*". A visualização não é tão flexível pois é necessário usar as *Pan Keys*, o que atendendo a que no nosso caso é precedida de uma sequência de *escape*, não se torna muito agradável.

6.2. ÁREA DOS COEFICIENTES

A apresentação dos coeficientes do filtro é feita neste local, numa das duas formas possíveis e já nossas conhecidas: *POLY* e *SOS*. A visualização da totalidade dos resultados segue os mesmos trâmites que a do ponto precedente. Os coeficientes e comentários associados poderão ser modificados, aparecendo a palavra "*Modified*" e também a expressão "*Not current with parameters*" nas áreas de análise de resultados, pois essas alterações ainda não se reflectiram num recalcular do processo de síntese.

Após a alteração manual, podem-se constatar as suas implicações, uma a uma, se seleccionarmos uma das ferramentas de análise.

6.3. ÁREA DE RESPOSTA EM FREQUÊNCIA

A actualização deste *viewport* é executada aquando da digitação de um comando explícito ou de uma forma implícita quando se requisita uma determinada operação (por exemplo: quando se pede ao **FDS** para projectar um filtro, podemos também forçar o sistema a actualizar as janelas de análise).

A apresentação é feita num gráfico cuja escala vertical é graduada em dB. O utilizador pode manipular esse gráfico quer na presença ou não de *grids*, transformação da escala horizontal em logarítmica bem como na indicação dos seus limites (bastante útil no caso de se pretender analisar apenas a banda passante, por exemplo).

Existe também um gráfico onde se pode visualizar a resposta em fase do filtro ou então o atraso de grupo, permitindo ver o atraso médio de cada componente do sinal de entrada em função da frequência.

Como auxiliar de análise, podemos utilizar um cursor que pode ser movimentado com o rato ou teclado, e que em cada posição nos fornece toda a informação possível sobre o ponto seleccionado.

6.4. ÁREA DE RESPOSTA IMPULSIONAL

Também aqui nos é fornecido um gráfico em que se descreve a resposta impulsional do filtro em análise, em função do tempo. Em termos de manipulação e visualização, é em tudo análogo ao caso anterior.

6.5. ÁREA DO MAPEAMENTO PÓLOS-ZEROS

Neste caso, o **FDS** apresenta-nos um gráfico cartesiano onde se representa a transformada z do "paciente" em observação. Os pólos são representados por uma pequena cruz (x) e os zeros (quando aparecem...) por uma circunferência (o). No caso de termos pólos coincidentes a notação adoptada é a dum asterisco (*) enquanto que para o caso dos zeros é utilizado círculo negro (•).

Uma circunferência auxiliar de raio unitário é também desenhada com o objectivo de nos permitir uma rápida avaliação sobre a estabilidade do filtro (ou seja, se todos os pólos estiverem contidos na área por ela delimitada então o filtro é estável).

A visualização é partidária da mesma política que os restantes elementos havendo neste caso que destacar duas facetas importantes: a deslocação do cursor de análise centra-se sempre no ponto crítico mais próximo; é possível efectuar o *zoom* de uma região definida pelo utilizador localmente.

6.6. MENUS E ÁREA DE COMANDO

Para um utilizador do **SDE** ou dos *expert menus* do **BDE** não há muito a dizer sobre o sistema de menus do **FDS**. É naturalmente um modelo hierárquico, em que as opções possíveis vão surgindo à medida que se tomam decisões. Existem também *switches* que se podem manipular conforme o desejo do projectista. Um exemplo clássico é o da automatização da apresentação dos resultados das análises de cada vez que é concebido um novo filtro.

Á medida que se selecciona um comando no menu, este aparece digitado na sua linha própria, podendo aí, o utilizador alterar os parâmetros que bem entender. Junto a esta linha de comandos existe um *viewport* do sistema onde aparecem informações sobre operações em curso e também, as sempre indesejáveis mensagens de erro.

7. MÉTODOS DE CONCEPÇÃO DE FILTROS DIGITAIS

Em pontos precedentes referenciámos algumas das técnicas utilizadas na síntese de filtros digitais. Algumas delas envolvem uma elevada complexidade analítica pelo que se tornaria fastidioso e pouco produtivo, implementá-las artesanalmente. É aqui que o **FDS** interpreta o papel principal.

De facto, aquilo que ele nos proporciona é um ambiente agradável em que o projectista é confrontado com um vasto leque de menus, em que dispõe de diversas alternativas para solucionar o seu problema. Mais concretamente, dado um conjunto de dados que se possuem e também um pacote de especificações que se pretende implementar, o engenheiro terá ao seu alcance várias formas de encarar o problema. Resta-lhe optar por aquela que obtiver a melhor relação qualidade *vs* recursos envolvidos (neste caso já não falamos de recursos em termos de esforço de projecto, mas sim de implementação: número de coeficientes, precisão necessária, tipo de estrutura, etc...).

Os métodos podem ser catalogados em dois grandes grupos: pré-definidos e definidos pelo utilizador. No primeiro caso ainda poderemos incluir mais um rótulo por forma a identificar o filtro quanto à sua resposta impulsional: **IIR** ou **FIR**. Um pouco à margem destes considerandos teóricos, surge ainda outro grupo: os filtros **SDE** (esta família de filtros pertence a uma das outras duas classes conforme o tipo de filtro que se pretende). Esta "etnia" não perfilha nenhuma religião revolucionária mas tão somente consiste num conjunto de filtros que utiliza o mesmo formato que os seus congéneres do *Signal Display Editor*.

Os parágrafos anteriores são apenas a justificação da organização adoptada na apresentação que segue.

7.1. SÍNTESE DE FILTROS DIGITAIS

A utilização do programa é extremamente fácil e pode ser totalmente apreendida após algumas experiências (ou através do *tutorial*).

O primeiro passo é o de seleccionar um dos métodos de concepção disponíveis. Para tal, basta activar o comando: **METHOD SELECT** *método tipo*. O "método" (pormenores nos próximos pontos) está relacionado com o algoritmo utilizado, estrutura pretendida, dados disponíveis, qualidade desejada, etc. O "tipo", definirá (de notar que nem todos os métodos permitem projectar o mesmo "tipo" de filtros) se se trata de um passa-baixo, passa-alto, passa-banda, etc.

Após a execução do comando, surgem no *viewport* de parâmetros uma lista de especificações inerentes ao método escolhido (normalmente, existem comentários apensos a estes parâmetros com o intuito de aclarar o que se pretende ou então, o de fornecer uma lista de possíveis respostas). A alteração dos valores de *default* é feita *in place*.

Finalmente, resta executar o comando: **METHOD MAKE_FILT**. Os coeficientes do filtro surgem no seu *viewport* (no formato POLY ou SOS em função do método seleccionado) e serão levadas a cabo as análises de resposta em frequência, resposta impulsional e/ou diagrama de pólos e zeros, conforme se activaram ou não, os *switches* próprios.

O projectista poderá efectuar todas as análises que lhe aprouver, bem como alterar o valor dos parâmetros (artesanalmente), ou mesmo, as sua propriedades (precisão dos coeficientes). As novas análises terão de ser feitas uma a uma, tendo o utilizador de indicar especificamente, aquela que pretende ver realizada.

7.2. MÉTODOS *STANDARD* DO FDS

Dada a grande complexidade envolvida na teoria subjacente à maior parte dos métodos, enveredaremos pela sua vertente voltada para o utilizador, remetendo os mais cépticos para um qualquer livro de processamento de sinal.

Apresentaremos os métodos pela sua ordem original, pois esta pode ser alterada pelo utilizador (no caso de este estar interessado em ter uma biblioteca de métodos mais utilizados) através do comando: **METHOD** liblist=*padrão*.

7.2.1. Métodos para Filtros IIR

O procedimento básico efectuado pelo **FDS** na síntese destes filtros, é a seguinte:

1. Determinar os pólos e zeros do filtro analógico normalizado.
2. "Des-normalizar" o filtro normalizado (Passa-baixo).
3. Digitalizar o filtro obtido.
4. Aplicar uma transformação geométrica para mudar do domínio de *Laplace* para o domínio z.

Os métodos garantem que o ganho máximo seja unitário (0dB).

No final desta secção, juntamos um pequeno resumo por forma auxiliar o utilizador nas decisões mais frequentes.

7.2.1.1. Método Bessel

O filtro passa-baixo normalizado tem um atraso de grupo máximo na origem e é caracterizado pela seguinte função de transferência (só pólos):

$$H(s) = \frac{d_0}{B_n(s)} \tag{12}$$

$B_n(s)$ é o polinómio de *Bessel* de ordem *n*, e d_0 é uma constante. Estes filtros tem uma resposta em frequência, desde a banda passante até à de corte, monótona. A frequência analógica correspondente ao limiar de passagem é sujeita a um pré-ênfase (*pre-warping*) segundo a lei:

$$\Omega_p = \frac{2}{T}\tan(\frac{\omega_p}{2}) \tag{13}$$

Ω_p é a frequência onde ocorrerá a atenuação requerida na especificação, *T* é o instante de amostragem e ω_p é o limiar de passagem da frequência digital.

7.2.1.2. Método Butterworth

Esta técnica proporciona uma banda passante bastante regular e horizontal, e uma banda de corte monotonamente decrescente. Um filtro de ordem *n* tem uma resposta quadrática dada pela expressão:

$$|H(j\omega)|^2 = \frac{1}{1+\Omega^{2n}} \tag{14}$$

A localização dos pólos provém de:

$$s_k = \begin{cases} e^{j\frac{(2k-1)\pi}{2n}}, & \text{para n par} \\ e^{j\frac{(k-1)\pi}{n}}, & \text{para n ímpar} \end{cases} \tag{15}$$

7.2.1.3. Método Chebychev I

Este tipo de síntese baseia-se numa configuração à base de pólos, e cuja resposta quadrática é expressa por:

$$|H(j\omega)|^2 = \frac{1}{1+\varepsilon^2 C_n^2(\Omega)}, \text{ e } C_n(\Omega) = \begin{cases} \cos(n\cos^{-1}\Omega), & |\Omega| \le 1 \\ \cosh(n\cosh^{-1}\Omega), & |\Omega| > 1 \end{cases} \tag{16}$$

ε é um parâmetro relacionado com o *ripple* que este filtro exibe na banda passante. O seu comportamento durante o corte é semelhante ao anterior.

7.2.1.4. Método Chebychev II

Os filtros projectados segundo esta técnica costumam designar-se por "filtros *Chebychev* inversos", pois têm uma banda passante monótona e uma banda de corte "modulada" por um *ripple* (*equiripple*). O quadrado da magnitude da sua resposta em frequência é dado por:

$$|H(j\omega)|^2 = \frac{1}{1+\varepsilon^2\left(\frac{C_n(\Omega_s)}{C_n(\frac{\Omega_s}{\Omega})}\right)^2} \tag{17}$$

Ω_S é a frequência superior de corte para o filtro normalizado.

7.2.1.2. Método Butterworth

Esta técnica proporciona uma banda passante bastante regular e horizontal, e uma banda de corte monotonamente decrescente. Um filtro de ordem *n* tem uma resposta quadrática dada pela expressão:

$$|H(j\omega)|^2 = \frac{1}{1+\Omega^{2n}} \tag{14}$$

A localização dos pólos provém de:

$$s_k = \begin{cases} e^{j\frac{(2k-1)\pi}{2n}}, & \text{para n par} \\ e^{j\frac{(k-1)\pi}{n}}, & \text{para n ímpar} \end{cases} \tag{15}$$

7.2.1.3. Método Chebychev I

Este tipo de síntese baseia-se numa configuração à base de pólos, e cuja resposta quadrática é expressa por:

$$|H(j\omega)|^2 = \frac{1}{1+\varepsilon^2 C_n^2(\Omega)}, \text{ e } C_n(\Omega) = \begin{matrix} \cos(n\cos^{-1}\Omega), & |\Omega| \le 1 \\ \cosh(n\cosh^{-1}\Omega), & |\Omega| > 1 \end{matrix} \tag{16}$$

ε é um parâmetro relacionado com o *ripple* que este filtro exibe na banda passante. O seu comportamento durante o corte é semelhante ao anterior.

7.2.1.4. Método Chebychev II

Os filtros projectados segundo esta técnica costumam designar-se por "filtros *Chebychev* inversos", pois têm uma banda passante monótona e uma banda de corte "modulada" por um *ripple* (*equiripple*). O quadrado da magnitude da sua resposta em frequência é dado por:

$$|H(j\omega)|^2 = \frac{1}{1+\varepsilon^2 \left(\frac{C_n(\Omega_s)}{C_n(\frac{\Omega_s}{\Omega})}\right)^2} \tag{17}$$

Ω_S é a frequência superior de corte para o filtro normalizado.

7.2.1.5. Métodos Elípticos

Estes procedimentos proporcionam filtros cujas respostas em frequência, para a mesma ordem *n*, possuem a transição mais íngreme entre a banda passante e a de corte. Um parâmetro característico deste método o factor de selecção *k*, que nada mais é que a razão entre a frequência superior de corte e a frequência limiar da banda de corte, no domínio analógico.

Um filtro elíptico de ordem *n* é caracterizado pela seguinte função de transferência (normalizado e analógico):

$$H_s = \frac{H_0}{D_0(s)} \prod_{i=1}^{r} \frac{s^2 + A_{0i}}{s^2 + B_{1i}s + B_{0i}} \qquad (18)$$

Em que,

$$r = \begin{cases} n-1 \text{ para } n \text{ par} \\ \frac{n}{2} \text{ para } n \text{ ímpar} \end{cases} \quad D_0(s) = \begin{cases} s + \sigma_0 \text{ para } n \text{ ímpar} \\ 1 \text{ para } n \text{ par} \end{cases} \qquad (19)$$

As constantes A_{0i}, B_{0i} e B_{1i} são determinadas usando funções Jacobianas elípticas. O projecto deste tipo de filtros necessita que se proceda ao pré-ênfase nos limiares das bandas passante e de corte, de modo em tudo análogo ao do método proposto por *Bessel*.

7.2.1.6. Métodos SDE

Esta biblioteca contém métodos de geração de filtros que operam da mesma forma que os filtros pertencentes à biblioteca *standard* de filtros (pode ser usada no **BDE**).

Assim, podem ser projectados e analisados os seguintes blocos (de segunda ordem):

- Filtro passa-banda: *filt/bpf.symbol*.
- Filtro tapa-banda: *filt/notch.symbol*.

O nome dado à biblioteca deriva do facto do **SDE** utilizar exactamente estes métodos para proceder a filtragem de sinais.

7.2.1.7. Resumo dos Métodos IIR

Este grupo de técnicas produzem resultados em termos de coeficientes de segunda ordem (com a excepção dos filtros **SDE**). Em geral fornecem resultados bastante agradáveis mesmo para uma ordem reduzida.

Uma análise comparativa poderá ser feita observando uma série de resultados que juntamos em anexo[9]. São os frutos de uma bateria de experiências em que procurámos projectar filtros passa-baixo usando uma quantidade semelhante de recursos. Cada um dos teste foi documentado com duas folhas de análise por forma a se poder observar a morfologia da banda passante e de corte (e também a resposta em fase e atraso de grupo).

Desta análise podemos concluir que se pretendermos um filtro sem *ripple* em todo o espectro então, a hipótese mais válida é a do método de *Butterworth*. Se no entanto, podermos conviver com esse "parasita" na banda de corte então, a melhor

[9]Ver anexo B: "Análise comparativa de métodos".

performance é alcançada pela técnica *Chebychev* II. Se a aplicação suportar o *ripple* mesmo na banda passante, não existem dúvidas: deve-se optar por um método elíptico.

Este estudo foi alargado aos outros tipos de filtro disponíveis na biblioteca *standard* pelo que os seus diagnósticos foram também arquivados para que o leque de observação possa ser o mais alargado possível.

Com o objectivo de facilitar a escolha de um dos métodos tendo em conta os dados disponíveis ao projectista, elaborámos um pequeno ábaco para os filtros **IIR** que o conduzirá rapidamente à melhor alternativa (em anexo[10]).

7.2.2. Métodos para filtros FIR

Estes métodos caracterizam-se por produzir filtros com respostas de fase linear (ou melhor, rectilínea) e pelos coeficientes serem fornecidos na forma polinomial.

7.2.2.1. Método Equiripple

O **FDS** baseia-se na teoria de *Parks-McClellan* para projectar esta família de filtros. Para além da fase linear, é também minimizado o erro de *Chebychev*[11], aproximando o filtro da sua resposta ideal. Os fundamentos teóricos, têm os seus direitos reservados para *Parks-McClellan et al.*, e como tal não podem ser divulgados.

Este método permite projectar os seguintes tipos de filtro:

- ***auto_BandPass*** (**FIR**, passa-banda, número de coeficientes automático).
- ***auto_BandStop*** (**FIR**, pára-banda, número de coeficientes automático).
- ***auto_HighPass*** (**FIR**, passa-alto, número de coeficientes automático).
- ***auto_LowPass*** (**FIR**, passa-baixo, número de coeficientes automático).
- ***BandPass*** (**FIR**, passa-banda).
- ***BandStop*** (**FIR**, pára-banda).
- ***Differentiator*** (**FIR**, diferenciador).
- ***Hilbert*** (**FIR**, filtro de *Hilbert*).
- ***HighPass*** (**FIR**, passa-alto).
- ***LowPass*** (**FIR**, passa-baixo).
- ***multi_BandPass*** (**FIR**, passa-banda de múltiplas bandas).

O número de coeficientes do filtro deverá ser inferior a 128 e todos os valores para as frequências de limiar deverão ser inferiores a metade da frequência de amostragem. Existe um parâmetro "*weight*" que representa o peso relativo dos desvios em cada banda.

O último tipo de filtro é bastante útil pois permite congregar uma associação de filtros num único elemento com diversas bandas passantes.

---

[10]Ver anexo A: "Ábaco IIR".
[11]Na bibliografia: "*minimum weighted Chebychev error*".

7.2.2.2. Métodos de Janela (*Windowed_sinc Methods*)

Esta sub-biblioteca contém procedimentos capazes de conceber filtros do tipo:

- Passa-baixo.
- Passa-alto.

Estes métodos de projecto de filtros **FIR** baseiam-se no facto de que a transformada de Fourier inversa de um impulso rectangular (resposta ideal no domínio das frequências) ser um seno cardinal de comprimento infinito. Naturalmente que um filtro digital não pode ter um número de coeficientes indeterminadamente grande pelo que se torna inevitável a truncagem do seno cardinal. A esse processo chama-se método da janela a qual representará um sub-tipo deste género de operações. O caso mais simples é o de termos uma janela rectangular (truncagem pura e simples) e origina uma resposta em frequência com alguns "*overshoots*" (em vez do natural degrau rectilíneo). Outras janelas mais elaboradas minimizam este fenómeno.

As opções disponíveis são:

- Rectangular.
- Hamming.
- Hanning.
- Blackman.
- Bartlett.
- Kaiser.

Estes métodos sintetizam filtros de ordem ímpar e fase linear. Mais detalhes quanto a estas janelas podem ser encontrados no manual.

7.2.2.3. Métodos SDE

Já os apresentámos antes como sendo métodos **IIR**, todavia, para certo tipo de filtros, estas técnicas são abrangidas no presente grupo. Os casos a que nos referimos são os seguintes:

- Filtro passa-baixo: *filt/lpf.symbol.*
- Filtro passa-alto: *filt/hpf.symbol.*

Mais informações no ponto homólogo anterior.

7.2.2.4. Resumo dos Métodos FIR

Como se disse estes métodos são indicados para implementações da estrutura *Direct Form I* pois os coeficientes já estão na forma polinomial. Para além disso, a garantia de uma fase linear é sempre um bom augúrio do funcionamento óptimo.

O estudo comparativo é uma sequência do anterior e dele podemos reter algumas constatações.

Assim, o método *equiripple* foi aquele conseguiu a melhor *performance*, seguido de perto pelo da janela rectangular. Todavia este resultado não pode ser considerado como regra geral pois a condicionante de não utilizar filtros com um número de coeficientes superior a 10 (isto porque o estudo pretendia envolver todos os métodos pelo que adoptamos o mesmo critério sempre que possível), viciava um pouco os resultados pois estes filtros atingem o rendimento óptimo a partir dos 25 coeficientes.

A confirmação destas constatações pode ser conseguida no mesmo anexo[12].

7.3. MÉTODOS DEFINIDOS PELO UTILIZADOR

Para além de todos os métodos citados nos pontos precedentes, o **FDS** proporciona a faculdade de o projectista definir um algoritmo "personalizado". Ou seja, ele, melhor que ninguém, sabe dos dados de que dispõe, os quais podem não ser directamente utilizáveis em nenhum dos métodos *standard*. Logo, poderá desenvolver uma rotina de tratamento desses dados e de seguida aplicar um algoritmo válido para a síntese de filtros. Esta é uma das principais virtudes deste *package* pelo que lhe dedicaremos a nossa atenção de seguida.

7.3.1. Procedimentos Gerais

Se se trata da primeira vez em que se vai conceber um método, ou se se pretende que o mesmo seja arquivado numa biblioteca separada, temos que criar essa biblioteca. Tal deverá ser realizado com o programa próprio para o efeito[13]. De seguida, o **FDS** tem de a reconhecer através do comando: "**FILES CREATE_LIB METHOD...**". Se nesta biblioteca se quiserem colocar métodos já existentes poderá usar-se o comando de estabelecimento de *defaults*: "**FILES SET_DEFAULTS...**".

Preparado o cenário podemos passar à acção. O primeiro acto é o de informar o sistema de que queremos desenvolver um novo método: "**METHOD UTILITES CREATE method...**". A esta deixa, o **FDS** criará um novo directório (pendurado no da biblioteca) com o mesmo nome do método, o qual contém um ficheiro com uma máscara de parâmetros. Este é um dos mais importantes adereços que o actor principal (o engenheiro) deverá utilizar, pois é nele que se definem as especificações do filtro. Nesta altura poderá optar-se por uma linha trágica (editar o ficheiro com o *vi*) ou por uma romântica (utilizando um outro qualquer editor de texto).

Após o intervalo, e estabelecido o enredo, deve-se desenvolver o argumento, ou seja, o algoritmo. Para isso deve-se utilizar o comando base "**METHOD UTILITES MAKEFILES...**" seguido de dois *switches*: o primeiro especificando se se trata de um método com "notação" POLY ou SOS; o segundo indicará ao sistema se queremos um molde em C ou em Fortran. Após a sua execução, a atmosfera reinante adensa-se ainda mais, pois surgem mais três personagens (ficheiros): "main.c", "*.flags" e "*.C" ou "*.F". O pano cai e é chegada a altura de fazermos um novo intervalo (que podemos aproveitar para ir até à nossa janela de sistema).

É assim, nos "bastidores" que se desenrola parte substancial da obra a que vimos assistindo, visto que o engenheiro terá que implementar o método que pretende sobre um molde que lhe é fornecido. No caso de se pretender temperar a intriga um pouco mais, podemos sempre alterar o ficheiro responsável pelo *link* de outras bibliotecas: "*.flags".

Eis-nos chegados ao último acto, em que o suspense absorve toda a plateia: a compilação toma conta do palco ("**METHOD UTILITES COMPILE method...**"). Dependendo da qualidade dos intervenientes, esta técnica poderá ter de ser figura de cartaz durante mais algum tempo.

---

[12]Ver anexo B: "Análise comparativa de métodos".
[13]Através do comando: "*/caesys/sys_admin/lib_admin*".

O comando "**METHOD SELECT...**" entra em cena e atinge-se o clímax: surgem os parâmetros no *viewport* respectivo (no caso de um final feliz).

Finalmente, e com o público completamente enterrado nos seus assentos, o engenheiro poderá declamar (digitar) a frase heróica (comando): "**METHOD MAKE_FILT...**". Em lugar das reticências poderão ser colocados alguns argumentos infalíveis (instruções para que determinadas análises sejam automaticamente efectuadas) como sejam: "**[FREQ]**", "**[IMPULSE]**" e "**[POLE_ZERO]**".

7.3.1.1. O Ficheiro ".params"

Este ficheiro é constituído basicamente por um comentário (in)formativo e uma única declaração/inicialização de variável: a imprescindível frequência de amostragem.

O projectista poderá acrescentar os parâmetros que entender, escrevendo primeiro o nome da variável, o seu valor de *default* e um comentário adicional (entre chavetas). Está definido o *input* do método.

7.3.1.2. O Ficheiro ".C" ou ".F"

Para além de variar com a linguagem decidida para a sua implementação, esta máscara depende também da estrutura escolhida para os coeficientes (POLY ou SOS). Grande parte do seu aspecto formal é moldado com base no ficheiro anterior (parâmetros de entrada da função, por exemplo), sendo o restante salpicado com comentários (que orientam a organização do programa), testes de alocação de memória e algumas mensagens para o sistema.

Os coeficientes correspondem àqueles a que nos referimos no início deste trabalho (usamos a mesma notação).

7.3.1.3. O Ficheiro ".flags"

Este ficheiro é uma espécie de *header*, pois é nele que se especificam os *includes* (de directorias) e *links* necessários. A biblioteca matemática é agrupada automaticamente pelo que este ficheiro raramente é utilizado.

7.3.2. Exemplos de Aplicação

Vamos apresentar alguns modestos (mas eficazes) exemplos daquilo que podem ser os métodos definidos pelo utilizador. Como exemplos ilustrativos, decidimos conceber algo de simples mas útil (ao contrário do *tutorial*).

Assim, e como o **FDS** só permitia fazer a síntese de filtros a alto nível, fez-se notar a necessidade de uma ferramenta que pudesse manipular filtros a mais baixo nível (isto também nos permitiria perceber melhor a relação entre alguns fenómenos).

A primeira aplicação em que nos empenhámos era desavergonhadamente simples: síntese de filtros por introdução directa dos coeficientes (POLY).

A segunda, um pouco mais rebuscada, deveria permitir projectar filtros por manipulação de pólos e zeros (fins didácticos, portanto).

7.3.2.1. Exemplo 1: MANUAL

Neste método, os parâmetros de entrada são o número de coeficientes para o numerador e denominador da função de transferência do filtro (ver eq.[10]), para além da frequência de amostragem.

O ficheiro "manual.c" (no formato POLY C), praticamente só tem a declaração, alocação e inicialização de variáveis, pois visto que o **FDS** permite a alteração *in place* do valor dos parâmetros, só era necessário colocá-los à disposição do utilizador.

As listagens encontram-se reunidas no anexo[14].

7.3.2.2. Exemplo 2: CARTESOIDE

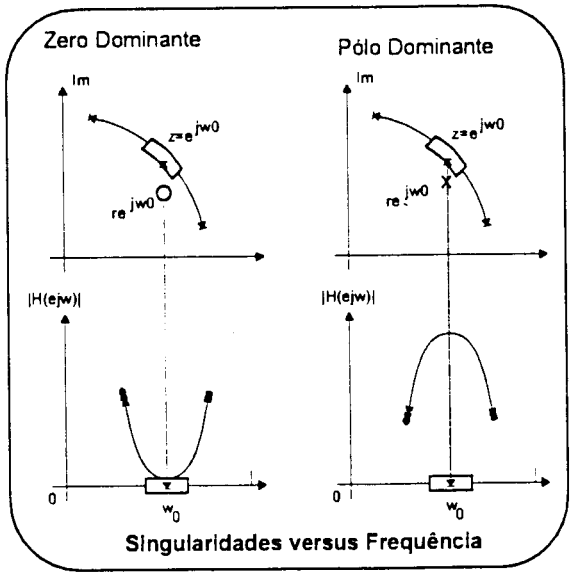
Um dos primeiros métodos de síntese interactiva de filtros digitais era baseada na disposição cuidada de pólos e zeros no seu diagrama cartesiano as quais se reflectiam fortemente na resposta em frequência (ou não fossem elas duas formas de representar a mesma coisa).

Assim, aquilo a que nos propusemos foi conceber um algoritmo que aceitasse a localização das singularidades e as transferisse para os coeficientes do filtro.

Antes de mais, apresentaremos uma pequena revisão (na sua vertente prática) sobre este método.

7.3.2.2.1. Pólos-Zeros *versus* Resposta em Frequência

Na transformada z, os pólos e zeros da sua função de transferência estão intimamente ligados à resposta em frequência pois quando se avalia do seu valor para uma dada frequência, a singularidade que dela estiver mais perto torna-se dominante. Se atentarmos ao facto de que o círculo unitário no domínio z é a transformada de Fourier desse sistema, então essa relação ainda se torna mais evidente.



Singularidades versus Frequência

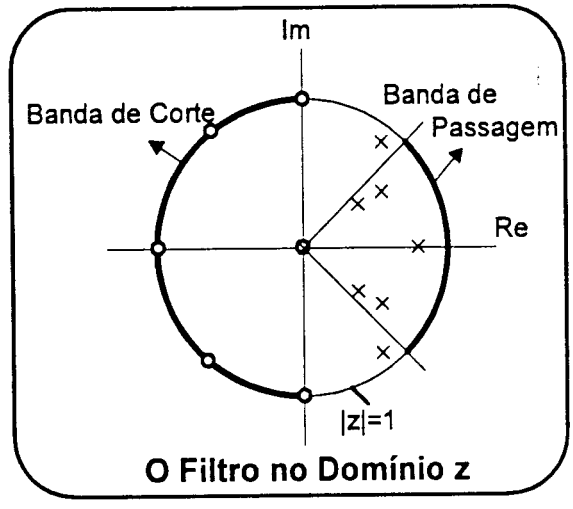
O efeito das singularidades dominantes pode ser facilmente apreendido pela figura seguinte.

Normalmente, nunca se colocam pólos cujo raio esteja fora do intervalo: $0.60<|z|<0.95$. O limite inferior previne que se coloquem pólos cujo efeito seja demasiado anémico e se torne um recurso subaproveitado. O limite superior garante que os problemas de *overflow* e arredondamento tenham alguma "folga" e não se tornem instáveis (ou seja, devem estar sempre contidos no círculo unitário).

[14]Ver anexo C: "Listagem de Métodos".

Dissemos que o círculo unitário correspondia à transformada de *Fourier* do sistema, e como tal, cada um dos seus pontos é uma imagem duma frequência. Deste modo, podemos marcar nessa circunferência as zonas em que pretendemos que o filtro se comporte como um curto-circuito e como um circuito aberto (entendemos o filtro como um dispositivo que tem um comportamento binário ao nível das frequências).

Este fenómeno poderá melhor ser entendido se atentarmos na próxima figura, em que temos a representação de um filtro passa-baixo no domínio z. Normalmente, a representação do espectro é apresentada num gráfico cujo valor máximo das abcissas é π, daí que a figura seja simétrica em relação ao eixo dos x (e é claro que todas as singularidades não coincidentes com um dos eixos, têm o seu complexo conjugado).



O Filtro no Domínio z

7.3.2.2.2. Solução Adoptada

Demonstrada a utilidade do método que se pretende desenvolver, resta somente isso mesmo: desenvolvê-lo.

Uma primeira hipótese seria a introdução das coordenadas das singularidades uma a uma e ir construindo os polinómios do denominador e numerador. Esta solução acarretava alguma complexidade e também levantava problemas quanto à validade dos dados introduzidos (por exemplo: os coeficientes dos polinómios não podem ser complexos). Por outro lado, iria exigir alguns algoritmos que se poderiam tornar perigosos quando necessitassem de técnicas mais elaboradas. O que pretendemos dizer, é que não poderíamos estar tão à vontade na programação como numa situação normal, pois o C do **FDS** não é tão *standard* como o apregoado e a nossa pouca documentação residia num único exemplo do manual.

Outra alternativa seria o aproveitamento da "notação" SOS pois reduziria o problema anterior a uma série de pequenos (tratáveis) problemas. Ou seja, uma secção de segunda ordem representa, no máximo, um pólo conjugado e um zero conjugado. Outras configurações serão: secção de primeira ordem (pólo e zero coincidentes com um dos eixos), um pólo (conjugado) e um zero (conjugado). Curiosamente, a representação destas configurações (na sua forma final) é a mesma que a utilizada pelo **FDS** para os coeficientes do tipo SOS. Assim, conhecendo o tipo de secção e as coordenadas das singularidades restava-nos determinar o valor dos coeficientes oficiais.

Por estas razões, adoptamos a hipótese SOS.

Um grave problema se colocou quando pensámos na forma de introduzir os dados para a execução do método. Isto porque a declaração de parâmetros no ficheiro ".params" é estática e como tal não permitia nenhuma flexibilidade para a introdução de dados (ou seja, seria a ideal para a concepção de filtros sempre com a mesma estrutura mas de diferentes coeficientes). Em condições normais nada nos impediria de utilizar um "scanf()", mas do exemplo do manual só sabíamos que o equivalente do "printf()" era um estranho "fd_message()".

O caminho que seguimos foi o de efectuar as declarações de parâmetros estáticos de início (como sejam a frequência de amostragem e ganho do filtro) e só depois "perguntar" o número de secções pretendidas para o filtro. Seguidamente, apareceria uma lista de conjuntos de parâmetros e em cada um especificar-se-ia o tipo de secção e as coordenadas do pólo e zero correspondente (é óbvio que no caso de se ter uma secção do tipo "*All_Pole*" poderá ignorar-se o zero que aparece nos parâmetros). Os restantes conjuntos (de índice superior ao número de secções serão desprezados. Dado que a lista de conjuntos teria de ter um comprimento fixo, limitámo-la a um máximo de 10, número que nos pareceu razoável (poderemos ter até 20 pólos e 20 zeros).

O ficheiro "cartesoide.c" é constituído por uma função com uma enorme lista de argumentos, seguida da declaração de variáveis essenciais (como é o caso dos coeficientes e *array* de tipos-STYLES[ ]) e também auxiliares (neste caso utilizamos *arrays* para conter os dados de entrada por forma a tornar o programa mais racional). O corpo do algoritmo é constituído por um ciclo (dependente do número de secções) dentro do qual existe um "*switch() case*" por forma a tratar separadamente cada tipo de secção. Em cada caso específico resta fazer o cálculo dos coeficientes bem como testar a validade dos dados introduzidos.

As listagens encontram-se agrupadas no anexo[15], pelo que nos absteremos de tecer mais considerações sobre o programa propriamente dito.

7.3.2.2.3. Exemplos de Utilização

Para demonstrar a utilidade do método, bem como do seu interesse didáctico, decidimos testar a sua validade com alguns exemplos retirados de um livro de processamento de sinal listado na bibliografia[16].

O primeiro coincide com o caso apresentado na figura anterior e trata-se da síntese de um filtro passa-baixo. As especificações são as seguintes:

$$\begin{aligned} &\text{Passa-banda:} \quad -1 < \left|H(e^{j\omega})\right|_{dB} \leq 0 \quad \text{para} \quad 0 \leq \omega \leq \frac{\pi}{4} \\ &\text{Banda de corte:} \quad \left|H(e^{j\omega})\right|_{dB} < -50 \quad \text{para} \quad \frac{\pi}{2} \leq \omega \leq \pi \end{aligned} \qquad (20)$$

O processo de concepção será interactivo, ou seja, introduzem-se algumas singularidades e analisa-se o resultado. Este último, apoiado com o conhecimento anterior, conduzirá à inserção de novas secções até se atingir a *performance* desejada. Podemos resumir o processo realizado nas próximas linhas, o qual poderá ser acompanhado pela verificação no anexo[17] em que, para além de alguns comentários, temos duas folhas de análise para cada passo com o objectivo de permitir supervisionar as condições das bandas passante e de corte.

---

[15]Ver anexo C: "Listagem de métodos".
[16]"Introduction to Digital..." de Roman Kuc.
[17]Ver anexo D: "Exemplo de interacção".

Cada passo é repetido uma vez para se ajustar o valor do ganho (para que o filtro tenha um ganho máximo unitário), através da expressão:

$$A = 10^{-\frac{H_{max}}{20}} \qquad (21)$$

Resumo da sequência de operações:

1. Coloca-se um pólo no centro da banda passante e um zero no centro da banda de corte. Isto "inicializa" o cumprimento das duas restrições. Trata-se de uma secção de primeira ordem pois as duas singularidades só possuem uma componente.
2. Um zero (e seu conjugado) é colocado no limiar da zona de corte com vista a gerar um foco de grande atenuação a essa frequência. Estamos na presença de uma secção do tipo "*All-Zero*".
3. Um pólo(e seu conjugado) é colocado no limiar da banda passante com o objectivo de incrementar o ganho nessa zona (tentar com a banda passante não seja decrescente com a frequência). O tipo de secção a seleccionar é o inverso do anterior: "*All-Pole*".
4. Um par de zeros (conjugados) são introduzidos perto do limiar de corte para assegurar a atenuação exigida para esse intervalo. Novamente o tipo: "*All-Zero*". Todavia, este passo levou a uma diminuição bastante sensível no limiar da zona passante.
5. Introduzem-se um par de pólos (conjugados) no interior da banda passante para vincar o cumprimento do limiar da banda passante. Tipo: "*All-Pole*".
6. Colocam-se mais dois pólos (conjugados), um pouco mais para o interior na banda passante com o objectivo de assegurar a primeira especificação (e dar alguma homogeneidade a essa faixa). Tipo: "*All-Pole*".

Estes resultados podem ser acompanhados no anexo[18].

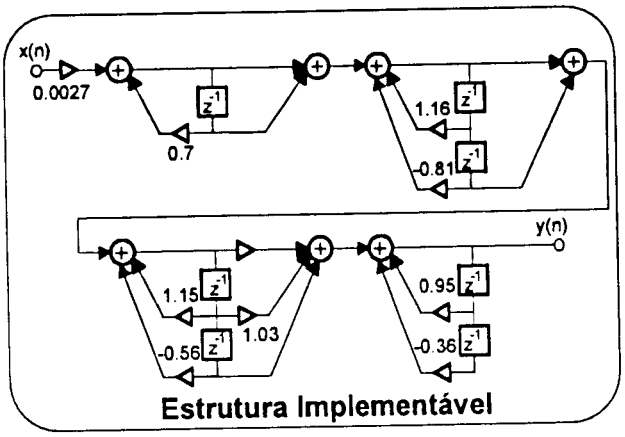
Lamentável e inexplicavelmente, o **FDS** mostrou-se bastante reticente em fazer um apresentação condigna do mapeamento da transformada z, pois raramente desenhava o círculo unitário e nem sempre os zeros eram visíveis. Em contrapartida, os poucos com que nos presenttava nunca eram removidos de livre vontade...

Após a síntese do filtro, podemos alterar a ordem de introdução de singularidades e procurar um agrupamento óptimo com o objectivo de aproveitar ao máximo as potencialidades de cada secção. Por exemplo: em vez de termos uma secção do tipo "*All-Pole*" e outra "*All-Zero*" podemos utilizar como sua substituta uma secção mais poderosa, ou seja, do tipo "Segunda Ordem".

Se o fizermos, podemos reduzir o número de secções necessárias de 6 para 4, o que é uma economia apreciável. O sistema óptimo está representado na figura seguinte, sendo a equação descritiva:

$$H(z) = \frac{0.0027(1+z^{-1})(1+z^{-2})(1+1.03z^{-3}+z^{-2})}{(1-0.7z^{-1})(1-1.16z^{-1}+0.81z^{-2})(1-1.15z^{-1}+0.56z^{-2})(1-0.95z^{-1}+0.36z^{-2})} \qquad (22)$$

[18]Ver anexo D: "Exemplo de interacção".



Estrutura Implementável

Concebemos ainda um filtro passa-banda utilizando técnicas semelhantes pelo que reduzimos a sua presença às folhas de análise juntas no anexo[19].

Ainda em relação a estes exemplos, experimentámos utilizar pólos de módulo perto do unitário e em seguida limitar a precisão dos coeficientes segundo técnicas já apresentadas. O resultado foi reconfortante já que nunca se atingiram situações de instabilidade. Tal facto deve-se a que redução no número de bits utilizados na representação do valor dos coeficientes, gera sempre aproximações por defeito. Logo, existe uma certa tendência para que os raios se "encurtem".

7.4. FILTROS *VERSUS* FICHEIROS

Para que os filtros possam ser preservados em ficheiros, há que criar uma biblioteca à imagem do que se disse para o caso dos métodos. O **FDS** proporciona uma gestão de acessos às suas bibliotecas de modo análogo ao processo utilizado pelo sistema UNIX na protecção das contas individuais (também se contemplam os grupos de trabalho).

Existem duas grandes alternativas quando se coloca a questão de gravar um filtro: utilizar o modo "*filter*" ou o modo "*TO_SIGNAL*".

No primeiro caso, toda a informação relativa aos parâmetros e coeficientes (e até mesmo comentários e avisos do tipo: "*Modified*"), é preservada. Se não se tiverem problemas de memória em disco nada nos impede de gravar também as análises que se pretenderem. O ficheiro respeita o formato ASCII e é bastante auto-explicativo. Aliás, foi apartir dele que conseguimos reunir informação suficiente para realizar o método "cartesoide" descrito em pontos precedentes.

A hipótese "*TO_SIGNAL*" serve para gravar o filtro segundo um formato compatível com o **SDE**. Neste caso só se podem preservar os coeficientes os quais devem ser separados em dois ficheiros: um conterá o polinómio do numerador e o outro o do numerador. Por esta razão não se podem gravar filtros desta forma no formato SOS, impedimento que não existe na primeira alternativa.

[19]Ver anexo D: "Exemplo de interacção".

8. UTILIZAÇÃO DE FILTROS "FDS" NO SDE

Na sequência do último parágrafo, em que já nos debruçámos sobre a primeira parte deste problema (da inteira responsabilidade do **FDS**) resta-nos dedicar uma linha para o **SDE**.

No "processador de sinais", e partindo do princípio que a biblioteca de sinais do **FDS** e do **SDE** são as mesmas, basta invocar um filtro do tipo pretendido. Nessa linha de comando surgem dois campos em que se devem especificar os ficheiros relativos ao polinómio do numerador e do denominador, e utilizar o filtro tão candidamente como qualquer outro.

Uma forma muito mais imediata de comunicação entre os dois módulos é também possível no caso de querermos utilizar no **SDE** o filtro "activo" (corrente) no **FDS**. Para tal, basta utilizar um comando próprio para essa situação.

9. UTILIZAÇÃO DE FILTROS "FDS" NO BDE

É perfeitamente legítimo que se deseje utilizar um filtro tão carinhosamente concebido e "educado" no **FDS**, como um dos blocos utilizáveis no *Block Diagram Editor*. Isto apesar de estarem já disponíveis numa vasta panóplia de blocos *standard*.

O processo para o conseguir é bastante simples, embora não seja muito dedúzível sem a ajuda do manual.

Depois de projectar um filtro digital deve-se gravá-lo no formato "*filter*" na biblioteca usual. Tudo o que havia para ser feito no **FDS** se resume à última linha.

Em seguida, e já no **BDE**, deve-se digitar o comando "**ADD INSTANCE *model=filt/filter.symbol***" (deve-se garantir que a opção "*expert menus*" no menu "*TOOLS*" está activada), independentemente do filtro sintetizado. A este comando, o **BDE** responde com um símbolo, também ele, sempre igual.

O busílis da questão está na manipulação dos parâmetros deste bloco. Fazendo o "**PUSH NEXT**", aparecem uma lista de parâmetros, os quais não são todos válidos simultaneamente. Assim, existe um deles (a que se pode atribuir "*yes*" ou "*no* ") que permite seleccionar a fonte de atributos do filtro. Ou seja, pode-se pretender que ele implemente um filtro baseado numa lista de coeficientes (alterável e tratável como parâmetros normais) ou que os recolha de um determinado ficheiro. É claro que é esta a situação que nos interessa. Resta-nos pois, digitar no campo correspondente, o nome da biblioteca e do ficheiro fonte.

Por forma a evitar confusões entre os diversos filtros construídos por este método (já que o seu aspecto exterior se mantém inalterado) podemos optar por uma de duas soluções. A primeira alternativa é pura e simplesmente digitar algum comentário *in place* no próprio símbolo. É uma opção bastante rápida e eficaz, mas que traz problemas quanto à mobilidade do mesmo pois o **BDE** não reconhece a ligação entre o símbolo e o comentário. A outra alternativa baseia-se na construção de um novo símbolo para o filtro. Para isso basta acrescentar os "portos" de entrada e saída e gravar o novo "*.detail*". O inconveniente desta técnica reside na introdução de mais uma fronteira entre o símbolo do filtro e os seus parâmetros.

9.1. O DOCTOR FILTER

Até agora, e após o estudo que vem sendo apresentado, deve-se ter notado que a síntese e análise de filtros é algo unidireccional. Por outras palavras, temos total controlo sobre as características de um filtro que se projecta desde a sua fase inicial, mas não podemos analisar um filtro desconhecido (assumindo que este não dispões de ficheiros compatíveis com o **FDS**).

Para além disso, e mesmo que os filtros "não identificados" possam ter ficheiros de formato reconhecível, não nos é permitido estudar o comportamento duma associações de diversos filtros.

Por sermos da opinião de que este assunto era merecedor de alguma atenção, dedicámos-lhe alguma da nossa juventude.

Visto que deveríamos ser livres de analisar um qualquer sistema, decidimos desenvolver este micro-projecto no **BDE**.

A característica que escolhemos como mais relevante para ser determinada foi a resposta em frequência do sistema desconhecido. A técnica mais elementar, e cientes de que a saída do sistema no domínio de *Fourier* é fruto da multiplicação dos espectros desse mesmo sistema (a determinar) e do sinal injectado, concluímos que o nosso sinal de teste teria de ter um espectro constante e unitário. Deste modo, o espectro do sistema e da saída do mesmo seriam coincidentes.

A transformada inversa de *Fourier* dum degrau de *Heaviside* (o nosso objectivo) é tão simplesmente um impulso de *Dirac*. Por outro lado, também sabemos que à multiplicação dos espectros no domínio das frequências, corresponde a convolução no domínio dos tempos. Nada mais fácil! Basta-nos ligar à entrada do sistema desconhecido um gerador de impulso de *Dirac* (existente na biblioteca *standard*) e um *Signal Sink* à saída do sistema.

Por forma a tornar o processo ainda mais transparente para o utilizador, desenvolvemos um novo bloco designado por "*Doctor Filter*" que possuem uma entrada e uma saída, sendo constituído unicamente pelos dois elementos citados. Em anexo[20] encontram-se alguns detalhes deste novo companheiro de projecto.

Em termos de utilização, o projectista deverá construir o sistema a ser analisado e colocar o *Doctor Filter* de modo a fechar o circuito. Em seguida deverá proceder ao "**CREATE/RUN**" do programa de simulação do modelo, escolhendo para o número de iterações um valor não inferior a 5000 (para ter boa resolução nos resultados). O sinal produzido deverá ser recolhido com o "**GET SIM**" do **SDE** e é nada mais nada menos que a resposta impulsional do sistema. Se se calcular a FFT deste sinal (a 2048 pontos e janela rectangular) obtemos a resposta em frequência do sistema, a menos de um factor de escala. De facto, os espectros obtidos por este método e os reais, coincidem na forma mas diferem no *off-set* e também ligeiramente na escala. É possível que tal facto esteja relacionado com o valor de amplitude do impulso de *Dirac* que só pode aproximar o seu valor ideal, ou seja, o infinito. O mesmo se aplica à duração do impulso (inversamente).

Em anexo[21] temos dois exemplos de utilização.

Em primeiro lugar, sintetizámos dois filtros passa-banda, para os quais criámos dois blocos no **BDE**. Em seguida, construímos dois sistemas: uma associação em série e outra em paralelo. Aplicamos o *Doctor Filter* e respeitamos todos os

[20]Ver anexo E: "Doctor Filter".
[21]Ver anexo E: "Doctor Filter".

trâmites da sua deontologia. Nesse anexo encontram-se os espectros individuais e os resultantes das associações.

De referir que a versão 3.0 dispõe de uma ferramenta algo semelhante incluída no **FDS**, a qual nos apressamos a utilizar para confirmar a validade dos resultados obtidos. A confirmação não se fez esperar e juntamos a documentação devida a esse anexo.

A princípio, e apesar de não duvidarmos das suas virtudes, o *Doctor Filter* pareceu-nos um tanto ou quanto primitivo. Assim, começamos a desenvolver um novo protótipo que efectuasse directamente a transformada de *Fourier* pretendida. Todavia, o sinal só poderia ser visualizado no **SDE**, e tratando-se de um sinal complexo normal, não mereceria um tratamento privilegiado em relação aos seus pares. Isto é, não disporíamos das escalas adequadas para a correcta observação dos dados recolhidos. Concluímos assim que a acrescida complexidade do sistema não se iria reflectir numa melhor qualidade na utilização do mesmo.

Mais uma vez se demostrou que a simplicidade é a alma do engenho.

10. RESUMO

Após o estudo apresentado não será difícil de perceber que o **FDS** é hoje em dia uma ferramenta indispensável nos "laboratórios" de engenharia actuais.

Apesar de termos feito uma abordagem algo superficial sobre um tema tão vasto quanto este, pensamos ter aberto uma larga diversidade de caminhos que se abrem em direcção a um universo em que os conceitos de engenho e de implementação se encontraram cada vez mais indissociáveis. Tudo isto porque o fenómeno do "pronto-a-vestir" chegou à engenharia electrotécnica e não parece mostrar sinais de se esvaecer.

Como pontos sensíveis deste *package* temos a considerar alguns *bugs* relativos à apresentação do mapeamento da transformada z (visível, ou melhor, invisível nos exemplos do anexo) e também à pouca informação contida nos manuais. De facto, se não fomos mais longe é apenas porque não consta do *curriculum* nenhuma disciplina de ciências ocultas ou sobre a arte da adivinhação.

10.1. A VERSÃO 3.0

A chegada, algo turbulenta, do *upgrade* do *software* foi uma das razões pelas quais não podemos apresentar mais alguns exemplos por forma argumentar as nossas afirmações. Isto porque, com a instalação da nova versão, a antiga foi enterrada com todos os seus mistérios.

Todavia, o **FDS 3.0**, não é apenas um *remake* do seu antecessor com um novo visual. Esta versão, no curto espaço de tempo em que tivemos oportunidade de a conhecer, demonstrou algumas virtudes não só no que diz respeito à eliminação dos *bugs* referidos, como também em algumas ferramentas novas.

Uma delas foi já referenciada no ponto anterior e é de extrema utilidade para o estudo da associação de filtros. Existem também novos métodos de concepção de filtros digitais e mesmo os antigos têm algumas alterações ao nível (pelo menos) da introdução de dados.

Existem também ferramentas para auxiliar o engenheiro na síntese interactiva como é um bom exemplo o comando *target*. Esta nova opção permite efectuar o mapeamento do espectro em zonas críticas (definidas pelas especificações que se pretendem respeitar) as quais são um guia útil ao projectista.

Para além destas (e de muitas outras) inovações, estão disponíveis muitas ferramentas de edição (segundo o ambiente *Windows*) e visualização que tornam o trabalho muito mais flexível e agradável.

Esta nova versão também terá um merecido relatório logo que tenhamos tempo para uma familiarização mais profunda (e esperemos que fecunda também).

11. BIBLIOGRAFIA

Roman Kuc, "*Introduction to Digital Signal Processing*", McGraw-Hill 1988.

A. Bruce Carlson, "*Communication Systems*", Third Edition, McGraw-Hill 1986.

"SPW™ *Filter Design System User's Guide*", 2.7 Revision, 1990.

"SPW™ *Signal Processing Worksystem User's Guide & Tutorial*", 2.7 Rev., 1990.

12. ANEXOS

ANEXO A: "ÁBACO IIR"

TABELA PARA FILTROS *IIR*

TIPO	Center frequency	Passband bandwidth	Passband attenuation	Passband edge	Passband Ripple	Stopband bandwidth	Stopband edge	Ordem	SOLUÇÃO
Low-Pass	x	x	♥	♥	x	x	x	<20	Bessel
	x	x	♥	♥	x	x	x	<40	Butterworth
	x	x	x	♥	♥	x	x	<40	Chebychev I
	x	x	♥	♥	x	x	♥	<40	Chebychev II
	x	x	x	♥	♥	x	♥	<40	Eliptic
High-Pass	x	x	♥	♥	x	x	x	<20	Bessel
	x	x	♥	♥	x	x	x	<40	Butterworth
	x	x	x	♥	♥	x	x	<40	Chebychev I
	x	x	♥	♥	x	x	♥	<40	Chebychev II
	x	x	x	♥	♥	x	♥	<40	Eliptic
Band-Pass	♥	♥	♥	x	x	x	x	<40	Bessel
	♥	♥	♥	x	x	x	x	<40, par	Butterworth
	♥	♥	x	x	♥	x	x	<40, par	Chebychev I
	♥	♥	♥	x	x	♥	x	<40, par	Chebychev II
	♥	♥	x	x	♥	♥	x	<40, par	Eliptic
	♥	♥	x	x	x	x	x	x	SDE_Filter
Notch	♥	x	x	x	x	♥	x	x	SDE_Filter
Band-Stop	♥	♥	♥	x	x	x	x	<40, par	Butterworth
	♥	♥	x	x	♥	x	x	<40, par	Chebychev I
	♥	♥	♥	x	x	♥	x	<40, par	Chebychev II
	♥	♥	x	x	♥	♥	x	<40, par	Eliptic

Nota:

Em todos os casos é necessário indicar qual a frequência de amostragem.

♥ — ?

x — ?

ANEXO B: "ANÁLISE COMPARATIVA DE MÉTODOS"

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=<null>

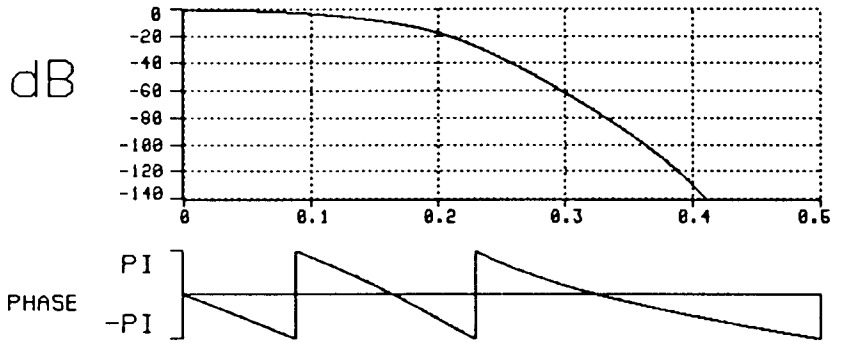
Filter =
Method = Bessel/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
passband_atten = 3 {passband attenuation in
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -36.004
Phase = 2.25798

SOS FILTER COEFFICIENTS

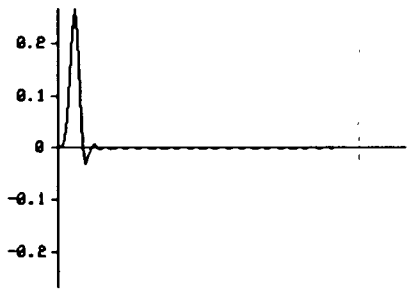
Num. of sections = 5 Gain = 0.000263919

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 2
b[0][2] = 1
a[0][0] = 1
a[0][1] = -0.3308136523
a[0][2] = 0.4876150913

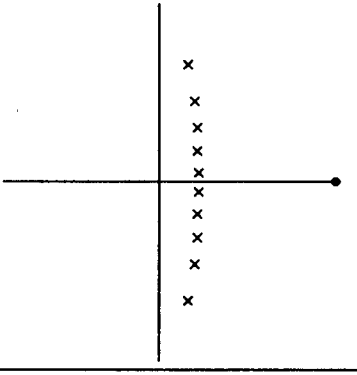
Section 1: second order
b[1][0] = 1
b[1][1] = 2
b[1][2] = 1
a[1][0] = 1
a[1][1] = -0.4046082527
a[1][2] = 0.2619515143

IMPULSE RESPONSE



Point = 64 of 128
Time = 64 secs
Value = 2.7716E-11

POLE ZERO



# of Poles = 10
# of Zeros = 10
Zero
Real = -1
Imag. = -0
Radius = 1
Angle = -180

FDS 2.8.2 I Filter Lib=valterFilts Method Lib=valtermeths Signal Lib=(null)

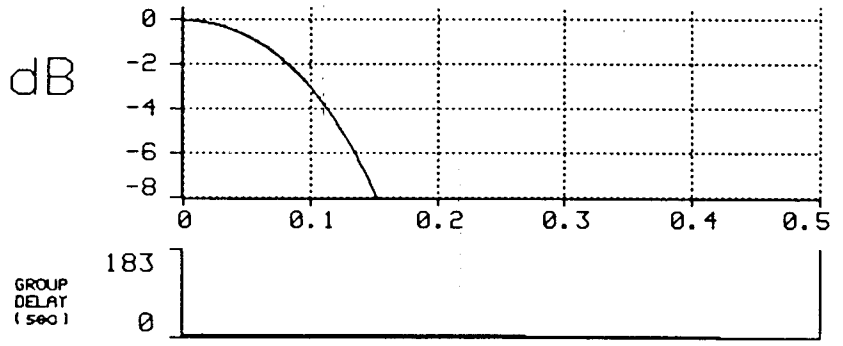
Filter =
Method = Bessel/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
passband_atten = 3 {passband attenuation in
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -36.004
Delay = 6.27867

SOS FILTER COEFFICIENTS

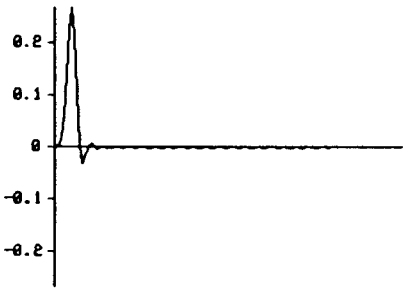
Num. of sections = 5 Gain = 0.000263919

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 2
b[0][2] = 1
a[0][0] = 1
a[0][1] = -0.3308136523
a[0][2] = 0.4876150913

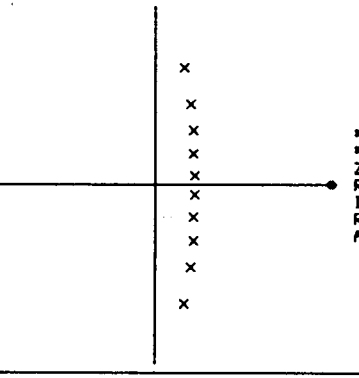
Section 1: second order
b[1][0] = 1
b[1][1] = 2
b[1][2] = 1
a[1][0] = 1
a[1][1] = -0.4046082527
a[1][2] = 0.2619515143

IMPULSE RESPONSE



Point = 64 of 128
Time = 64 secs
Value = 2.7716E-11

POLE ZERO



# of Poles = 10
# of Zeros = 10
Zero
Real = -1
Imag. = -0
Radius = 1
Angle = -180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

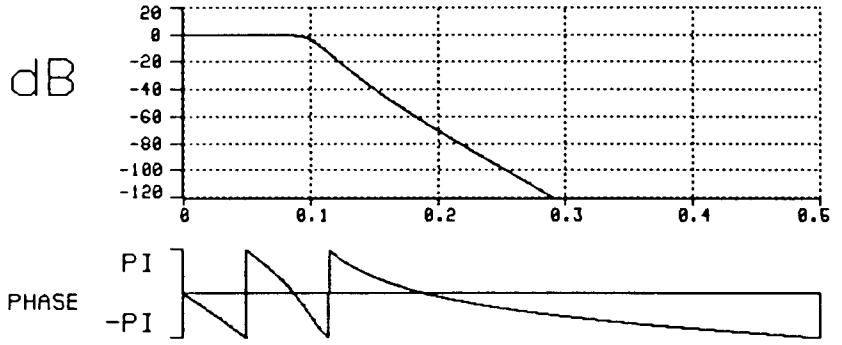
Filter =
Method = Butterworth/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
passband_atten = 3 {passband attenuation in
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -97.7607
Phase = -1.04116

SOS FILTER COEFFICIENTS

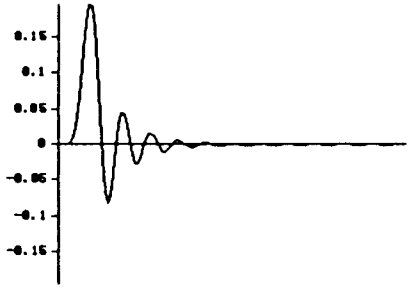
Num. of sections = 5 Gain = 1.68678e-06

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 2
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.481609876
a[0][2] = 0.8315562821

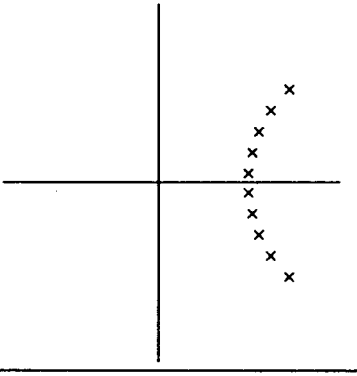
Section 1: second order
b[1][0] = 1
b[1][1] = 2
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.277030271
a[1][2] = 0.5786563339

IMPULSE RESPONSE



Point = 64 of 128
Time = 64 secs
Value = 0.0010168

POLE ZERO



# of Poles = 10
# of Zeros = 10
Zero
Real = -1
Imag. = -0
Radius = 1
Angle = -180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

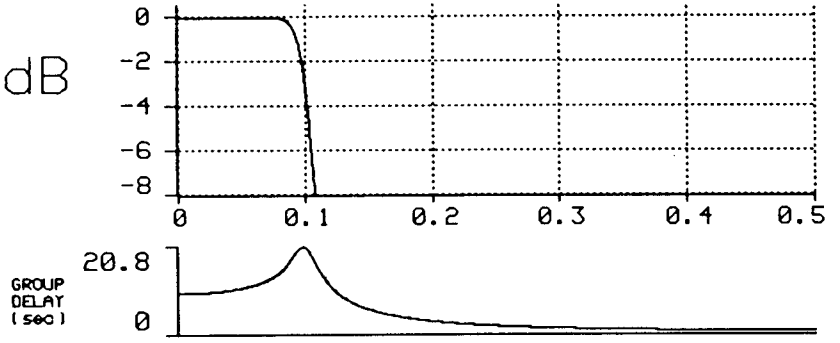
Filter =
Method = Butterworth/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
passband_atten = 3 {passband attenuation in
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -97.7607
Delay = 2.15508

SOS FILTER COEFFICIENTS

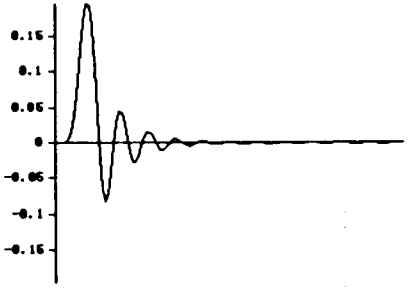
Num. of sections = 5 Gain = 1.68678e-06

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 2
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.481609876
a[0][2] = 0.8315562821

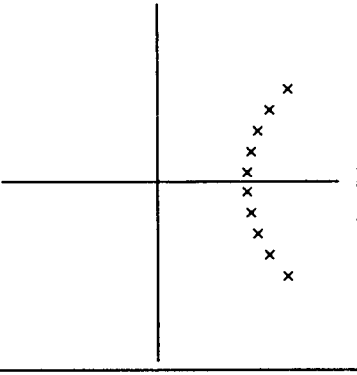
Section 1: second order
b[1][0] = 1
b[1][1] = 2
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.277030271
a[1][2] = 0.5786563339

IMPULSE RESPONSE



Point = 84 of 128
Time = 84 secs
Value = 0.0010168

POLE ZERO



# of Poles = 10
# of Zeros = 10
Zero
Real = -1
Imag. = -0
Radius = 1
Angle = -180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

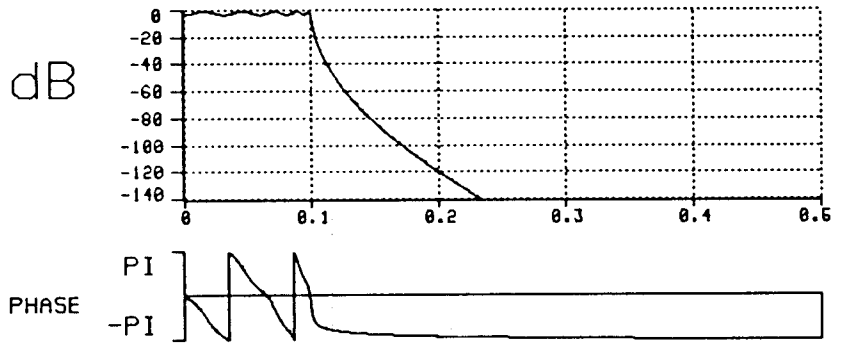
Filter =
Method = Chebyshev_1/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
passband_ripple = 3 {passband ripple in dB
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 200 of 1000
Freq. = 0.1001
dB = -3.47108
Phase = -1.16182

SOS FILTER COEFFICIENTS

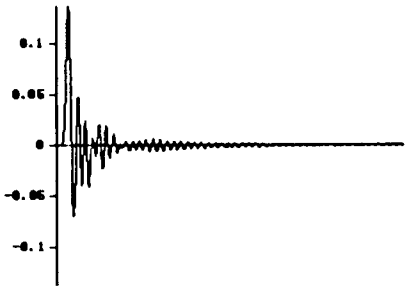
Num. of sections = 5 Gain = 1.66762e-08

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 2
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.610680784
a[0][2] = 0.9838452052

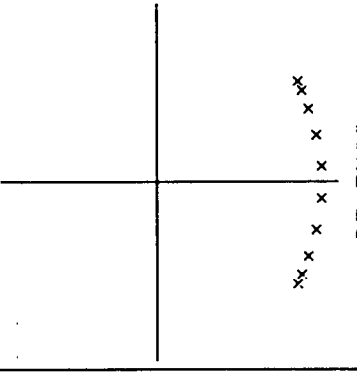
Section 1: second order
b[1][0] = 1
b[1][1] = 2
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.648223329
a[1][2] = 0.9530292254

IMPULSE RESPONSE



Point = 16 of 512
Time = 16 secs
Value = 0.13349

POLE ZERO



# of Poles = 10
# of Zeros = 10
Zero
Real = -1
Imag. = -0
Radius = 1
Angle = -180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

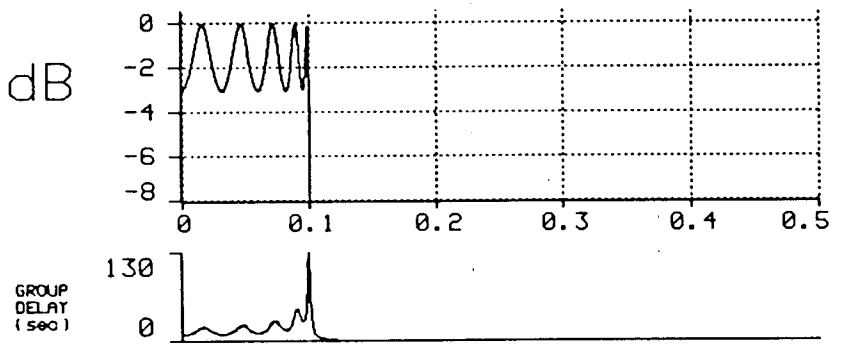
Filter =
Method = Chebyshev_1/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
passband_ripple = 3 {passband ripple in dB
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 200 of 1000
Freq. = 0.1001
dB = -3.47108
Delay = 90.779

SOS FILTER COEFFICIENTS

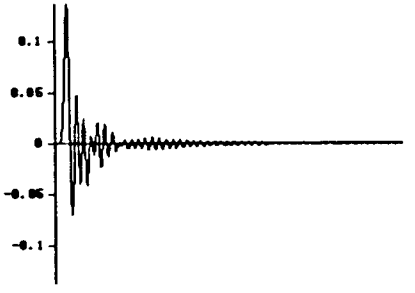
Num. of sections = 5 Gain = 1.66762e-08

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 2
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.610680784
a[0][2] = 0.9838452052

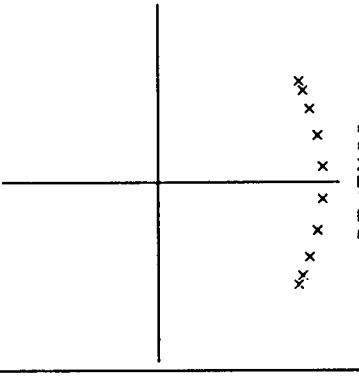
Section 1: second order
b[1][0] = 1
b[1][1] = 2
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.648223329
a[1][2] = 0.9530292254

IMPULSE RESPONSE



Point = 16 of 512
Time = 16 secs
Value = 0.13349

POLE ZERO



# of Poles = 10
# of Zeros = 10
Zero
Real = -1
Imag. = -0
Radius = 1
Angle = -180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

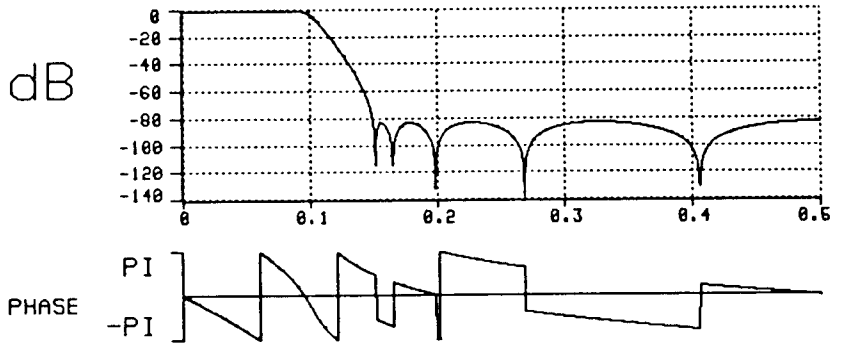
Filter =
Method = Chebyshev_2/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
stopband_edge = 0.15 {Stopband edge frequen
passband_atten = 3 {passband attenuation in
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 200 of 1000
Freq. = 0.1001
dB = -3.0606
Phase = -0.752344

SOS FILTER COEFFICIENTS

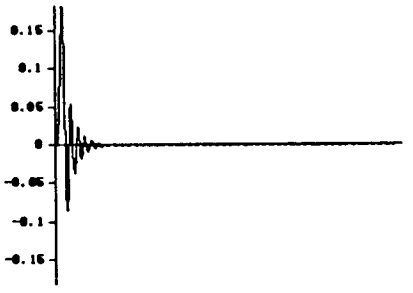
Num. of sections = 5 Gain = 0.000535011

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = -1.1592362
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.50674376
a[0][2] = 0.8665884903

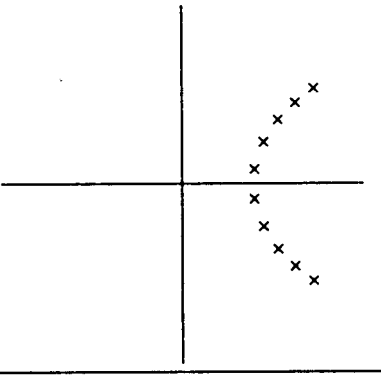
Section 1: second order
b[1][0] = 1
b[1][1] = -1.014279971
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.297875394
a[1][2] = 0.6360936953

IMPULSE RESPONSE



Point = 256 of 512
Time = 256 secs
Value = -1.5E-09

POLE ZERO



# of Poles = 10
# of Zeros = 10
Zero
Real = -0.82772
Imag.= 0.56115
Radius = 1
Angle = 145.865

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

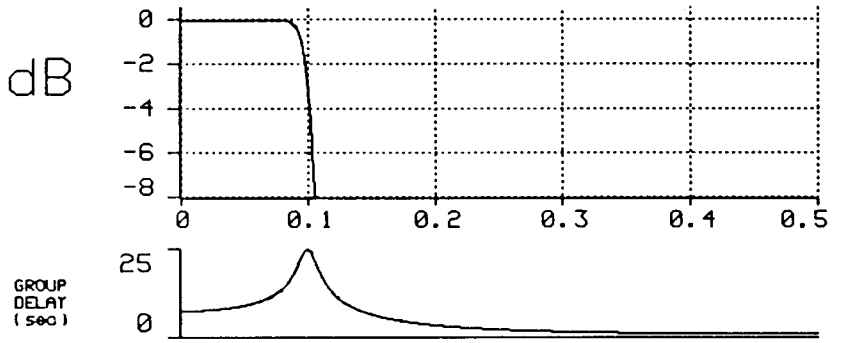
Filter =
Method = Chebyshev_2/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
stopband_edge = 0.15 {Stopband edge frequen
passband_atten = 3 {passband attenuation in
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 200 of 1000
Freq. = 0.1001
dB = -3.0606
Delay = 24.9723

SOS FILTER COEFFICIENTS

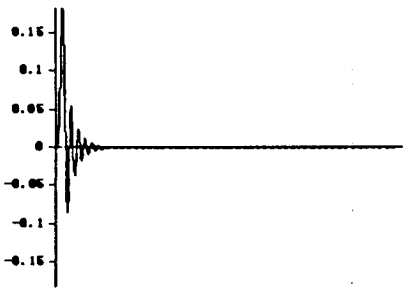
Num. of sections = 5 Gain = 0.000535011

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = -1.1592362
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.50674376
a[0][2] = 0.8665084903

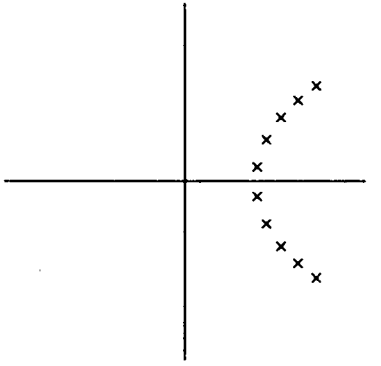
Section 1: second order
b[1][0] = 1
b[1][1] = -1.014279971
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.297875394
a[1][2] = 0.6360936953

IMPULSE RESPONSE



Point = 10 of 512
Time = 10 secs
Value = 0.18121

POLE ZERO



# of Poles = 10
# of Zeros = 10
Pole
Real = 0.41352
Imag. = 0.084505
Radius = 0.42208
Angle = 11.5497

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

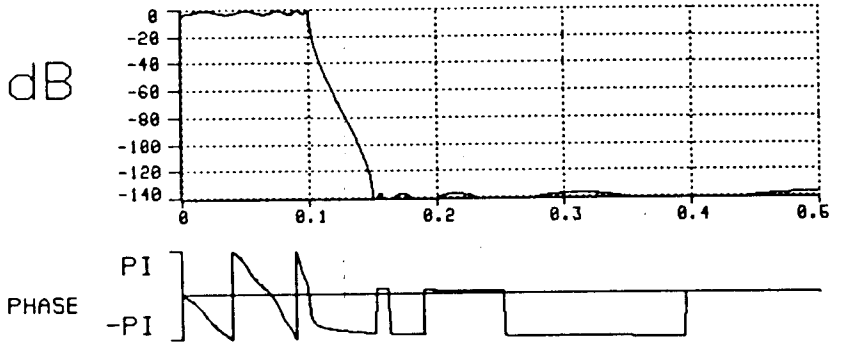
Filter =
Method = Elliptic/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
passband_edge = 0.1 {passband edge frequenc
stopband_edge = 0.15 {Stopband edge frequen
passband_ripple = 3 {passband ripple in dB
filter_order = 10 {desired filter order}

FREQUENCY RESPONSE



Point = 200 of 1000
Freq. = 0.1001
dB = -3.61975
Phase = -1.10306

SOS FILTER COEFFICIENTS

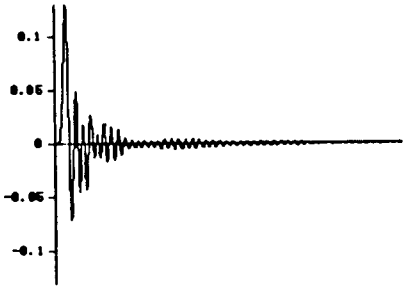
Num. of sections = 5 Gain = 1.30254e-06

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 1.56971781
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.864165718
a[0][2] = 0.8805219784

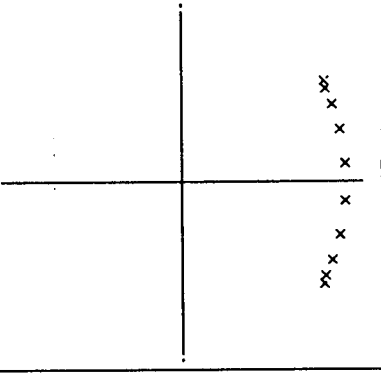
Section 1: second order
b[1][0] = 1
b[1][1] = 0.03320573278
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.799157438
a[1][2] = 0.90109754

IMPULSE RESPONSE



Point = 16 of 512
Time = 16 secs
Value = 0.12715

POLE ZERO



# of Poles = 10
# of Zeros = 10
Pole
Real = 0.89958
Imag. = -0.30308
Radius = 0.94926
Angle = -18.8192

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

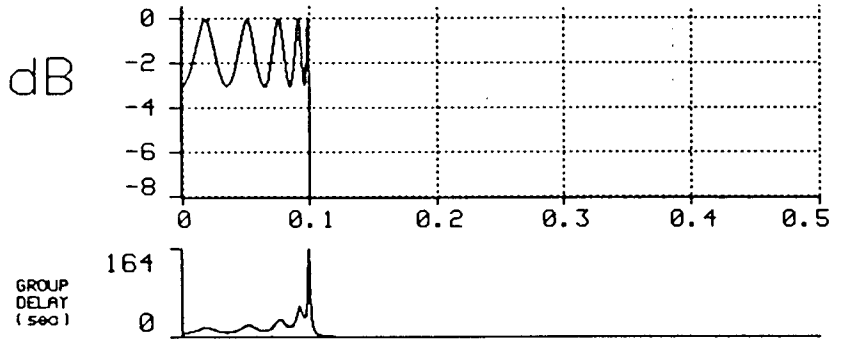
Filter =
Method = Elliptic/LowPass

COMMENT

PARAMETERS

```
sample_freq = 1   {sample frequency}
passband_edge = 0.1   {passband edge frequenc
stopband_edge = 0.15   {Stopband edge frequen
passband_ripple = 3   {passband ripple in dB'
filter_order = 10   {desired filter order}
```

FREQUENCY RESPONSE



Point = 200 of 1000
Freq. = 0.1001
dB = -3.61975
Delay = 116.336

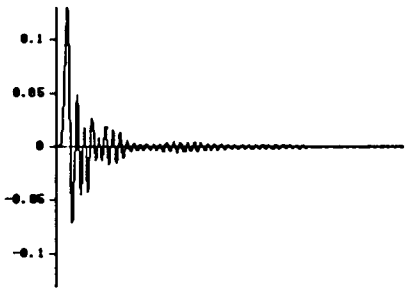
SOS FILTER COEFFICIENTS

Num. of sections = 5 Gain = 4.30254e-06

Sections

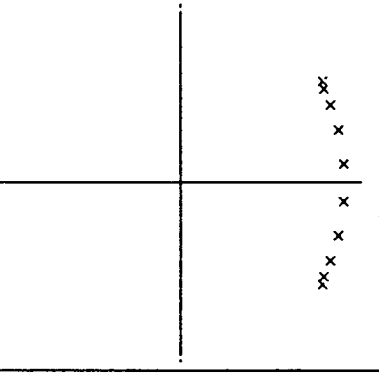
```
Section 0: second order
b[0][0] = 1
b[0][1] = 1.56971781
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.864165718
a[0][2] = 0.8805219784

Section 1: second order
b[1][0] = 1
b[1][1] = 0.03320573278
b[1][2] = 1
a[1][0] = 1
a[1][1] = -1.799157438
a[1][2] = 0.90109754
```

IMPULSE RESPONSE



Point = 18 of 512
Time = 18 secs
Value = 0.12715

POLE ZERO



# of Poles = 10
# of Zeros = 10
Pole
Real = 0.89958
Imag. = -0.30308
Radius = 0.94928
Angle = -18.6192

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

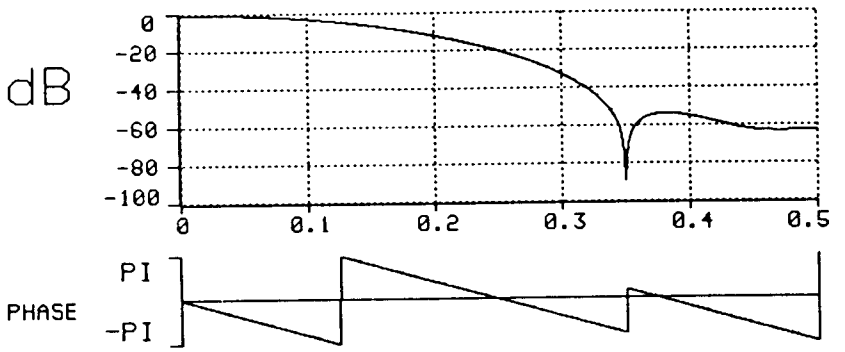
Filter =
Method = Sde_filters/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.13 {-6 dB cut off frequency
num_coefs = 10 {number of coefficients}

FREQUENCY RESPONSE



Point = 301 of 1000
Freq. = 0.150651
dB = -6.3577
Phase = 2.49692

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 9

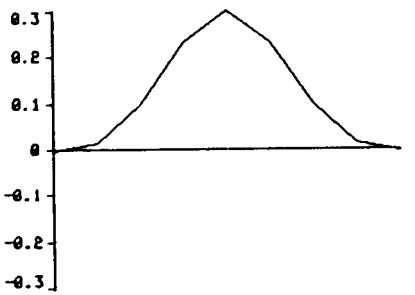
Numerators

b[0] = -0.0009271947999
b[1] = 0.01687626755
b[2] = 0.09967372378
b[3] = 0.2333107714
b[4] = 0.3021328642
b[5] = 0.2333107714
b[6] = 0.09967372378
b[7] = 0.01687626755

Denominators

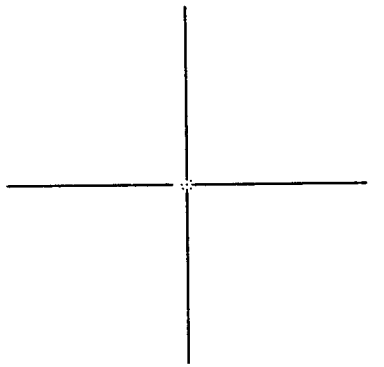
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.30213

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -1.2987
Imag.= 0.42557
Radius = 1.3591
Angle = 161.752

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

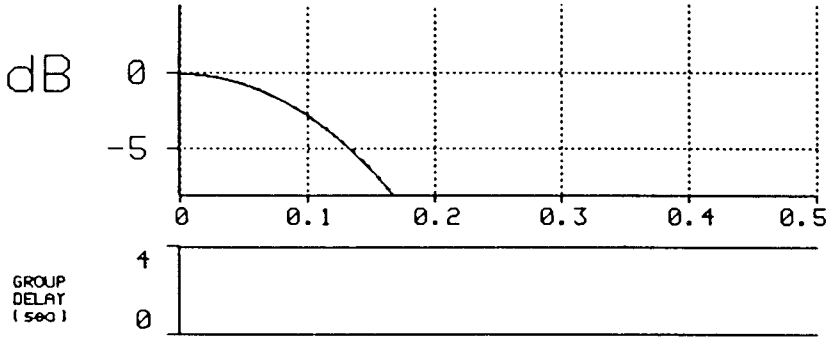
Filter =
Method = Sde_filters/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.13 {-6 dB cut off frequency
num_coefs = 10 {number of coefficients}

FREQUENCY RESPONSE



Point = 211 of 1000
Freq. = 0.105606
dB = -3.02721
Delay = 4

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 9

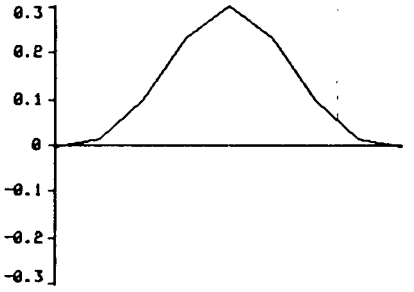
Numerators

b[0] = -0.0009271947999
b[1] = 0.01687626755
b[2] = 0.09967372378
b[3] = 0.2333107714
b[4] = 0.3021328642
b[5] = 0.2333107714
b[6] = 0.09967372378
b[7] = 0.01687626755

Denominators

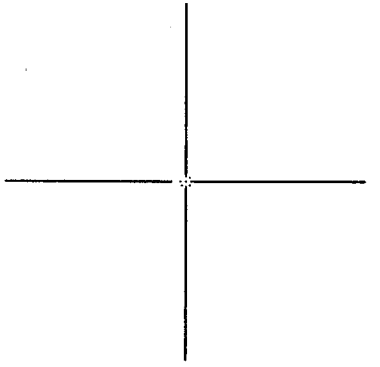
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.30213

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -1.2907
Imag.= 0.42557
Radius = 1.3591
Angle = 161.752

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

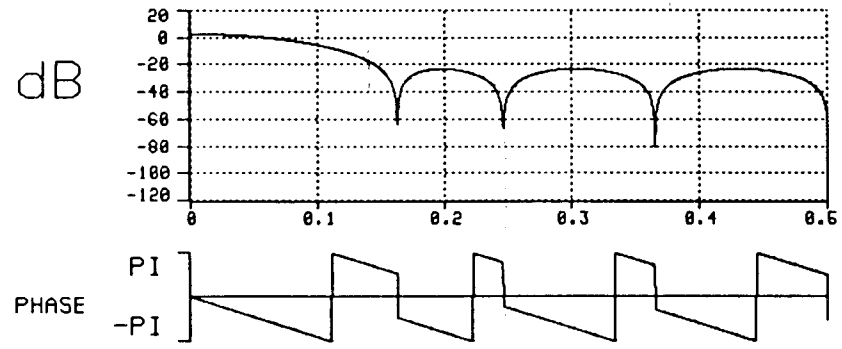
Filter =
Method = Equiripple/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
tap_length = 10 {filter length}
passband = 0.1 {passband edge frequency}
stopband = 0.15 {stopband edge frequency}
stopband_weight = 6 {stopband weight; passb

FREQUENCY RESPONSE



Point = 300 of 1000
Freq. = 0.15015
dB = -22.8557
Phase = 2.03779

POLY FILTER COEFFICIENTS

Num. of numers. = 10 Num. of denoms. = 1

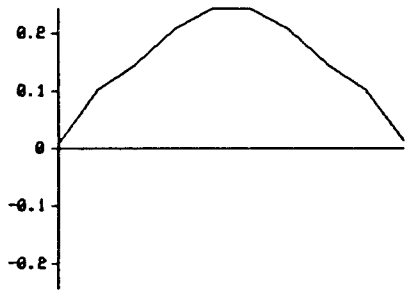
Numerators

b[0] = 0.01307235737
b[1] = 0.1048703964
b[2] = 0.1474677119
b[3] = 0.2093181831
b[4] = 0.244129798
b[5] = 0.244129798
b[6] = 0.2093181831
b[7] = 0.1474677119

Denominators

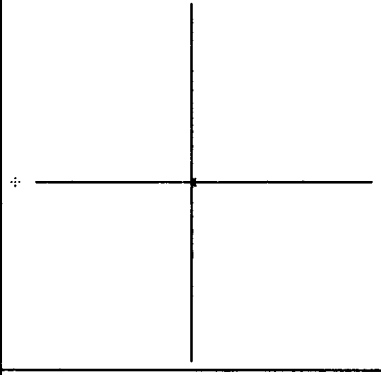
a[0] = 1

IMPULSE RESPONSE



Point = 5 of 10
Time = 5 secs
Value = 0.24413

POLE ZERO



# of Poles = 9
# of Zeros = 9
Zero
Real = -6.6294
Imag.= 0
Radius = 6.6294
Angle = 180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

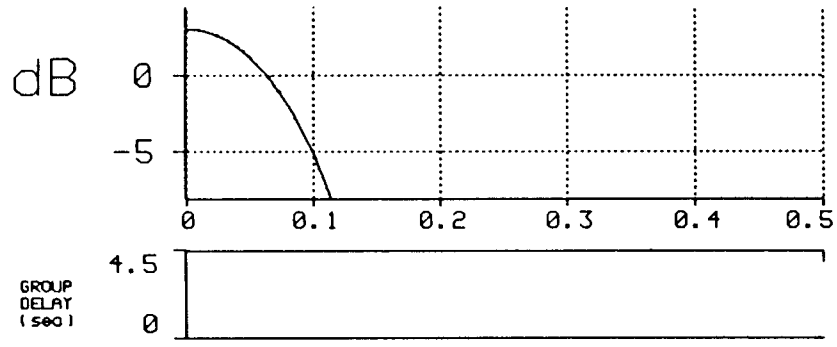
Filter =
Method = Equiripple/LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
tap_length = 10 {filter length}
passband = 0.1 {passband edge frequency}
stopband = 0.15 {stopband edge frequency}
stopband_weight = 6 {stopband weight; passb

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -41.9256
Delay = 4.5

POLY FILTER COEFFICIENTS

Num. of numers. = 10 Num. of denoms. = 10

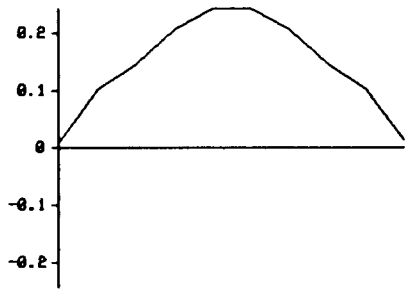
Numerators

b[0] = 0.01307235737
b[1] = 0.1048703964
b[2] = 0.1474677119
b[3] = 0.2093181831
b[4] = 0.244129798
b[5] = 0.244129798
b[6] = 0.2093181831
b[7] = 0.1474677119

Denominators

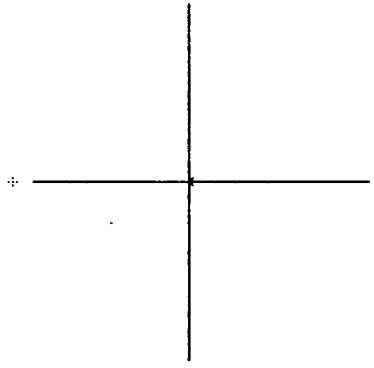
a[0] = 1

IMPULSE RESPONSE



Point = 5 of 10
Time = 5 secs
Value = 0.24413

POLE ZERO



# of Poles = 9
# of Zeros = 9
Zero
Real = -6.6294
Imag.= 0
Radius = 6.6294
Angle = 180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

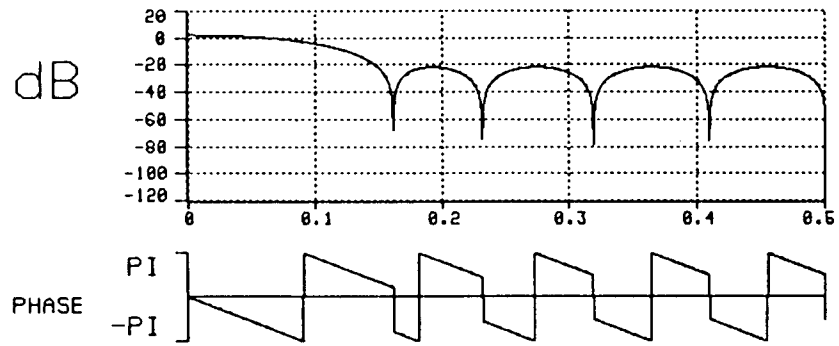
Filter =
Method = Equiripple/auto_LowPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
ripple = 3 {Passband ripple in dB}
attenuation = 20 {Stopband attenuation in d
passband = 0.1 {passband edge frequency}
stopband = 0.15 {stopband edge frequency}

FREQUENCY RESPONSE



Point = 301 of 1000
Freq. = 0.150651
dB = -21.8502
Phase = 1.07707

POLY FILTER COEFFICIENTS

Num. of numers. = 12 Num. of denoms. = 1

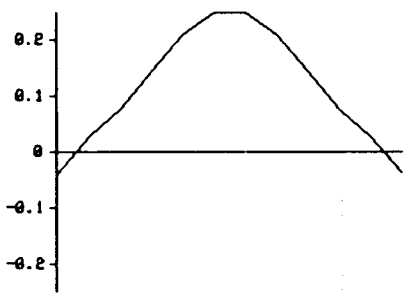
Numerators

b[0] = -0.0375612585
b[1] = 0.02931448883
b[2] = 0.07911746199
b[3] = 0.1466471354
b[4] = 0.2105767866
b[5] = 0.2495019418
b[6] = 0.2495019418
b[7] = 0.2105767866

Denominators

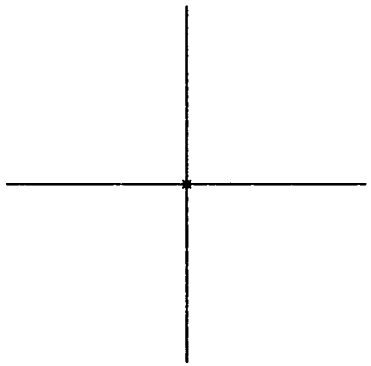
a[0] = 1

IMPULSE RESPONSE



Point = 6 of 12
Time = 6 secs
Value = 0.2496

POLE ZERO



# of Poles = 11
# of Zeros = 11
Zero
Real = -0.84083
Imag.= 0.5413
Radius = 1
Angle = 147.228

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter =
Method = Windowed_sinc/LowPass

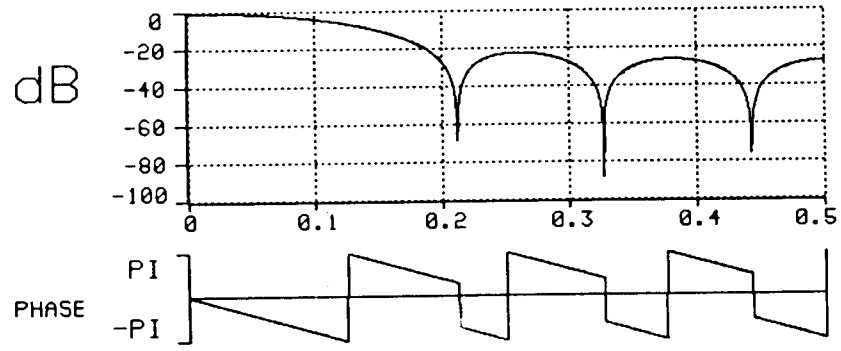
COMMENT

JANELA RECTANGULAR

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.135 {-6 dB cut off frequenc
num_coefs = 10 {number of coefficients}
window_type = 5 {1 BARTLETT, 2 BLACKMAN, 3
beta = 4.5 {applies only to KAISER window}

FREQUENCY RESPONSE



Point = 200 of 1000
Freq. = 0.1001
dB = -3.6788
Phase = -2.51579

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 1

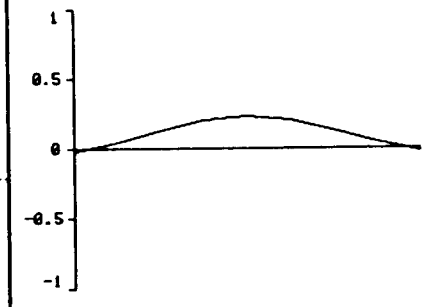
Numerators

b[0] = -0.01731368379
b[1] = 0.05217600645
b[2] = 0.138141204
b[3] = 0.2088896499
b[4] = 0.2362136467
b[5] = 0.2088896499
b[6] = 0.138141204
b[7] = 0.05217600645

Denominators

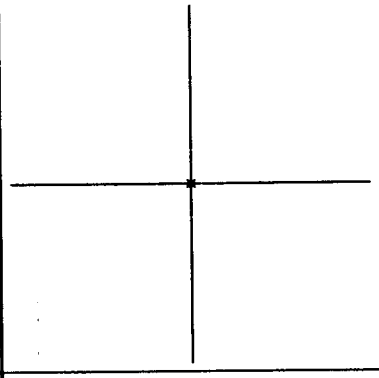
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.23621

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -0.46126
Imag.= 0.88726
Radius = 1
Angle = 117.469

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter =
Method = Windowed_sinc/LowPass

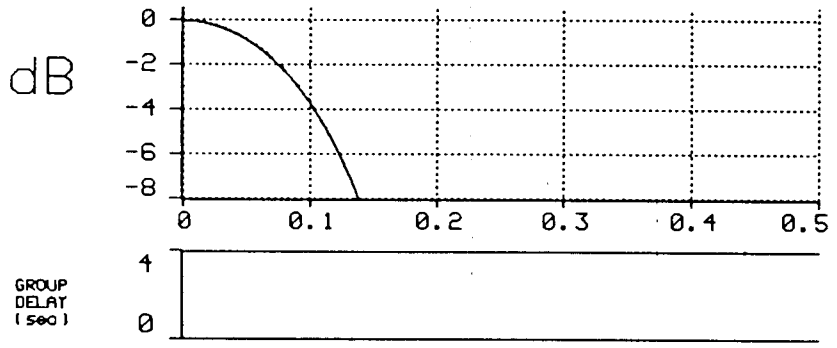
COMMENT

JANELA RECTANGULAR

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.135 {-6 dB cut off frequenc
num_coefs = 10 {number of coefficients}
window_type = 5 {1 BARTLETT, 2 BLACKMAN, 3
beta = 4.5 {applies only to KAISER window}

FREQUENCY RESPONSE



Point = 183 of 1000
Freq. = 0.0915916
dB = -3.02076
Delay = 4

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 1

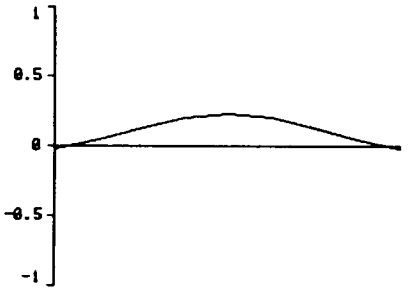
Numerators

b[0] = -0.01731368379
b[1] = 0.05217600645
b[2] = 0.138141204
b[3] = 0.2088896499
b[4] = 0.2362136467
b[5] = 0.2088896499
b[6] = 0.138141204
b[7] = 0.05217600645

Denominators

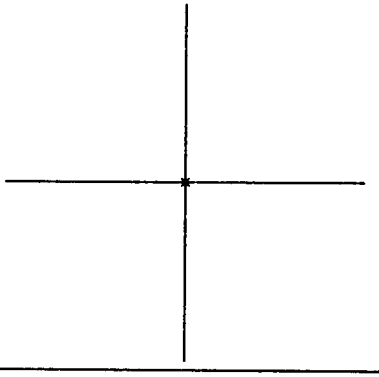
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.23621

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -0.46126
Imag.= 0.88726
Radius = 1
Angle = 117.469

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter =
Method = Windowed_sinc/LowPass

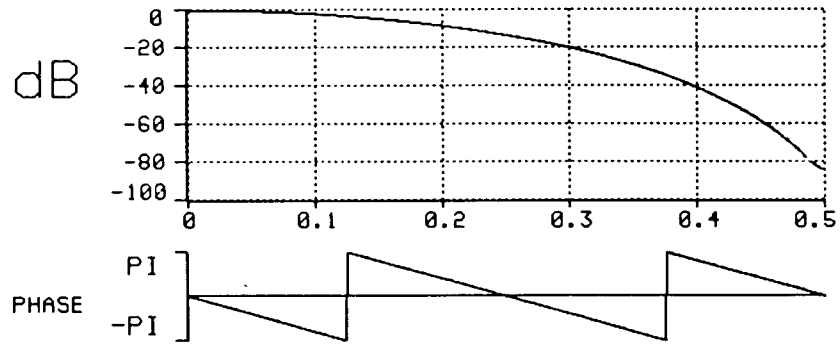
COMMENT

JANELA BLACKMAN

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.135 {-6 dB cut off frequenc
num_coefs = 10 {number of coefficients}
window_type = 2 {1 BARTLETT, 2 BLACKMAN, 3
beta = 4.5 {applies only to KAISER window}

FREQUENCY RESPONSE



Point = 600 of 1000
Freq. = 0.3003
dB = -20.0545
Phase = -1.26418

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 1

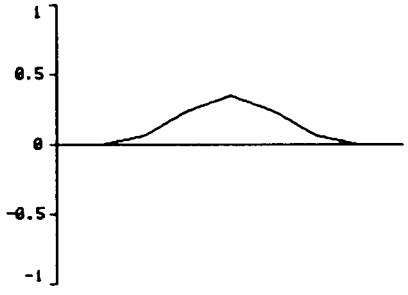
Numerators

b[0] = 3.639116748e-19
b[1] = 0.005250853777
b[2] = 0.07113583249
b[3] = 0.2447335334
b[4] = 0.3577595608
b[5] = 0.2447335334
b[6] = 0.07113583249
b[7] = 0.005250853777

Denominators

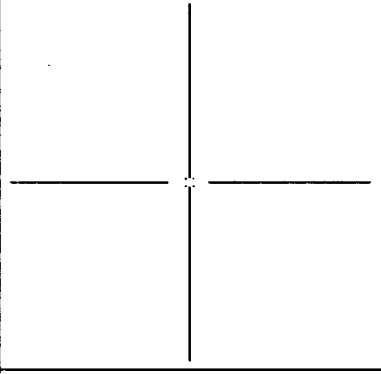
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.35776

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -1.5304
Imag.= 0
Radius = 1.5304
Angle = 180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter =
Method = Windowed_sinc/LowPass

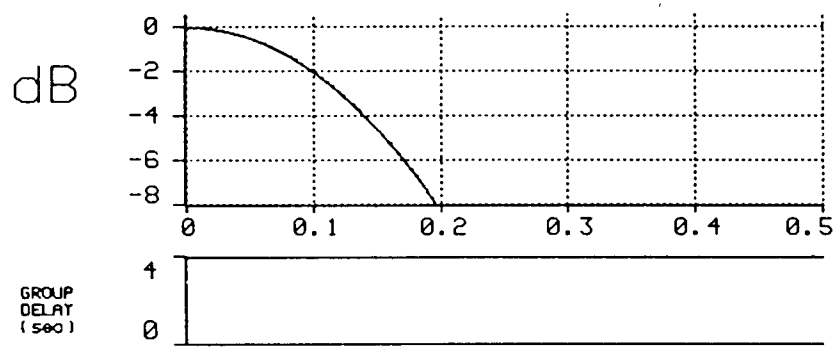
COMMENT

JANELA BLACKMAN

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.135 {-6 dB cut off frequenc
num_coefs = 10 {number of coefficients},
window_type = 2 {1 BARTLETT, 2 BLACKMAN, 3
beta = 4.5 {applies only to KAISER window}

FREQUENCY RESPONSE



Point = 244 of 1000
Freq. = 0.122122
dB = -2.9969
Delay = 4

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 1

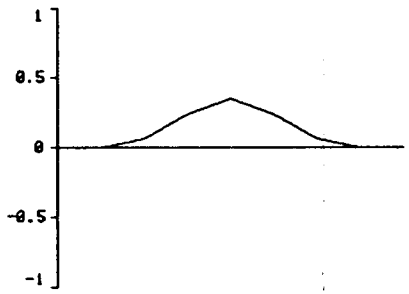
Numerators

b[0] = 3.639116748e-19
b[1] = 0.005250853777
b[2] = 0.07113583249
b[3] = 0.2447335334
b[4] = 0.3577595608
b[5] = 0.2447335334
b[6] = 0.07113583249
b[7] = 0.005250853777

Denominators

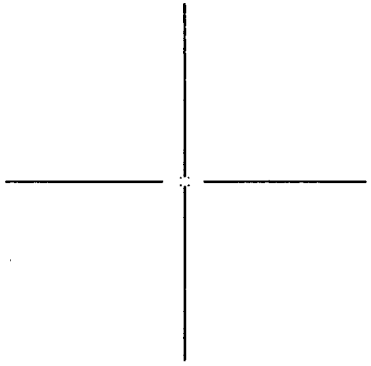
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.35776

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -1.5304
Imag.= 0
Radius = 1.5304
Angle = 180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter =
Method = Windowed_sinc/LowPass

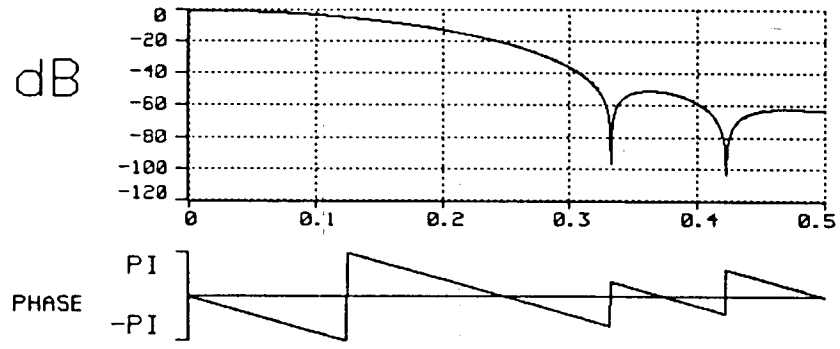
COMMENT

JANELA KAISER

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.135 {-6 dB cut off frequenc
num_coefs = 10 {number of coefficients}
window_type = 6 {1 BARTLETT, 2 BLACKMAN, 3
beta = 4.5 {applies only to KAISER window}

FREQUENCY RESPONSE



Point = 208 of 1000
Freq. = 0.104104
dB = -3.01198
Phase = -2.61642

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 1

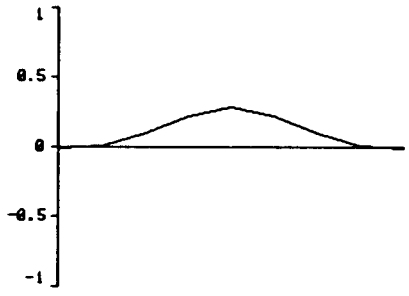
Numerators

b[0] = -0.001246124476
b[1] = 0.01798311629
b[2] = 0.1028378836
b[3] = 0.2318255808
b[4] = 0.2971990875
b[5] = 0.2318255808
b[6] = 0.1028378836
b[7] = 0.01798311629

Denominators

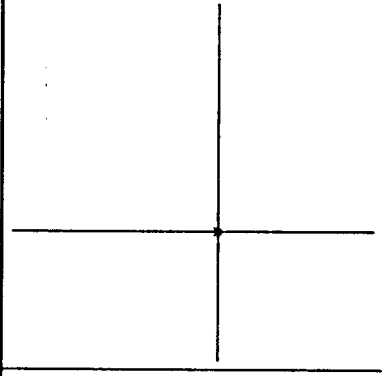
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.2972

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -0.49419
Imag.= 0.86935
Radius = 1
Angle = 119.617

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter =
Method = Windowed_sinc/LowPass

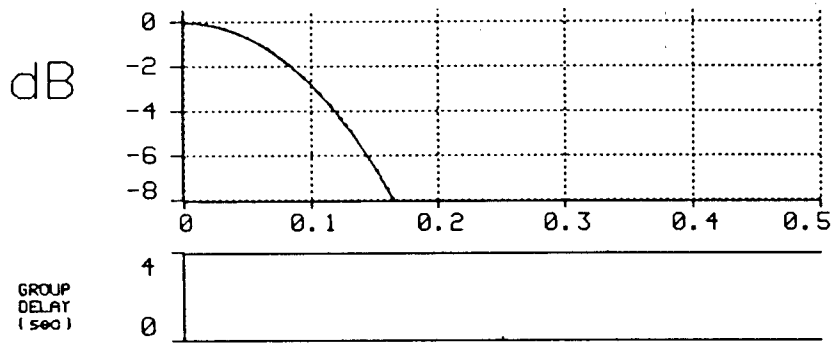
COMMENT

JANELA KAISER

PARAMETERS

sample_freq = 1 {sample frequency}
cutoff_freq = 0.135 {-6 dB cut off frequenc
num_coefs = 10 {number of coefficients}
window_type = 6 {1 BARTLETT, 2 BLACKMAN, 3
beta = 4.5 {applies only to KAISER window}

FREQUENCY RESPONSE



Point = 207 of 1000
Freq. = 0.103604
dB = -2.98215
Delay = 4

POLY FILTER COEFFICIENTS

Num. of numers. = 9 Num. of denoms. = 1

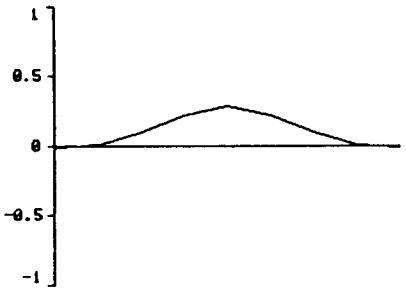
Numerators

b[0] = -0.001246124476
b[1] = 0.01798311629
b[2] = 0.1028378836
b[3] = 0.2318255808
b[4] = 0.2971990875
b[5] = 0.2318255808
b[6] = 0.1028378836
b[7] = 0.01798311629

Denominators

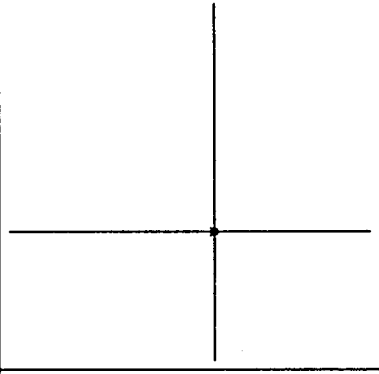
a[0] = 1

IMPULSE RESPONSE



Point = 4 of 9
Time = 4 secs
Value = 0.2972

POLE ZERO



# of Poles = 8
# of Zeros = 8
Zero
Real = -0.49419
Imag.= 0.86935
Radius = 1
Angle = 119.617

ANEXO C: "LISTAGENS DE MÉTODOS"

```
/*****************************************************************/
/*                                                               */
/*  PARAMETER FILE INFORMATION:                                  */
/*                                                               */
/*  --> This file should contain all of the names and default    */
/*      values of all of the parameters for the method.          */
/*      Each parameter may also be assigned a default comment.   */
/*                                                               */
/*      Sample Parameter list:                                   */
/*      (Actual list should be placed below this comment block)  */
/*                                                               */
/*      NOTE: First parameter must always be "sample_freq"       */
/*                                                               */
/*      sample_freq = 1 {sample frequency}                       */
/*      a_order = 2 {optional comment}                           */
/*      a_coeffs = 4.0                                           */
/*      b_order = 3.0 {The numerator order}                      */
/*      b_coeffs = 5                                             */
/*                                                               */
/*****************************************************************/

sample_freq = 1 {Frequencia de amostragem}
n_numerador = 3 {Coeficientes no numerador}
n_denominador = 3 {Coef. no denominador}
```

```
#include "FDDEFS.h"

/*****************************************************************/
/*                                                               */
/*     Method Name: manual                                       */
/*     Filter Type: POLY                                         */
/*     Date: Fri May 28 17:25:32 1993                            */
/*                                                               */
/*****************************************************************/

/*
 * FUNCTION: manual(sample_freq, n_numerador, n_denominador)
 * DESCRIPTION:
 *
 *  Sets the coefficients values based on the
 *  passed parameters.
 *
 * RETURN VALUE: SYS_OK or SYS_FATAL
 *
 */

manual(sample_freq, n_numerador, n_denominador)
double sample_freq;
double n_numerador;
double n_denominador;
{
  double *B;  /* array of numerator coefficients */
  double *A;  /* array of denominator coefficients */
  int num_numer_coefs;
  int num_denom_coefs;

  num_numer_coefs = n_numerador;
  num_denom_coefs = n_denominador;

  fd_message("Que metodo lindo!");

  /*
   * This section is used to allocate memory for the arrays of
   * coefficients.  The values of 'num_numer_coefs' and 'num_denom_coefs'
   * must be set before the following code is executed.
   */
  if (poly_allocate_memory(num_numer_coefs, num_denom_coefs, &B, &A) != SYS_OK)
    return(SYS_FATAL);

  /*
   * Use this section to assign values to the coefficient arrays.
   * For example:
   *   B[0] = 3.14;
   */

  if (poly_c_save(num_numer_coefs, num_denom_coefs, B, A) != SYS_OK)
    return(SYS_FATAL);

  return(SYS_OK);
}
```

```
/*****************************************************************/
/*                                                               */
/*  PARAMETER FILE INFORMATION:                                  */
/*                                                               */
/*  --> This file should contain all of the names and default    */
/*      values of all of the parameters for the method.          */
/*      Each parameter may also be assigned a default comment.   */
/*                                                               */
/*      Sample Parameter list:                                   */
/*      (Actual list should be placed below this comment block)  */
/*                                                               */
/*      NOTE: First parameter must always be "sample_freq"       */
/*                                                               */
/*      sample_freq = 1 {sample frequency}                       */
/*      a_order = 1 {optional comment}                           */
/*      a_coeffs = 4.0                                           */
/*      b_order = 3.0 {The numerator order}                      */
/*      b_coeffs = 5                                             */
/*                                                               */
/*****************************************************************/

/*****************************************************************/
/*                                                               */
/*              METODO CARTESOIDE                                */
/*  Pretende-se com este metodo, permitir que o projectista      */
/* possa introduzir manualmente a localiza‡.o dos polos e        */
/* zeros no dominio z.                                           */
/*                                                               */
/*****************************************************************/


sample_freq = 1 {Frequencia de Amostragem}
ganho =1 {Ganho do filtro}
num_seccoes = 1 {Limite maximo de 10}

seccao1 = 1 {Tipo 1:Segunda Ordem 2:Primeira Ordem 3:Um polo 4:Um zero}
zero1_real = 1 {Componente real do zero}
zero1_imag = 1 {Componente imag do zero}
polo1_real = 0.1 {Componente real do polo}
polo1_imag = 0.1 {Componente imag do polo}

seccao2 = 1 {Tipo para a seccao 2}
zero2_real = 1 {Componente real do zero}
zero2_imag = 1 {Componente imag do zero}
polo2_real = 0.1 {Componente real do polo}
polo2_imag = 0.1 {Componente imag do polo}

seccao3 = 1 {Tipo para a seccao 3}
zero3_real = 1 {Componente real do zero}
zero3_imag = 1 {Componente imag do zero}
polo3_real = 0.1 {Componente real do polo}
polo3_imag = 0.1 {Componente imag do polo}

seccao4 = 1 {Tipo para a seccao 4}
zero4_real = 1 {Componente real do zero}
zero4_imag = 1 {Componente imag do zero}
polo4_real = 0.1 {Componente real do polo}
polo4_imag = 0.1 {Componente imag do polo}

seccao5 = 1 {Tipo para a seccao 5}
zero5_real = 1 {Componente real do zero}
zero5_imag = 1 {Componente imag do zero}
polo5_real = 0.1 {Componente real do polo}
polo5_imag = 0.1 {Componente imag do polo}

seccao6 = 1 {Tipo para a seccao 6}
zero6_real = 1 {Componente real do zero}
zero6_imag = 1 {Componente imag do zero}
polo6_real = 0.1 {Componente real do polo}
polo6_imag = 0.1 {Componente imag do polo}

seccao7 = 1 {Tipo para a seccao 7}
zero7_real = 1 {Componente real do zero}
zero7_imag = 1 {Componente imag do zero}
polo7_real = 0.1 {Componente real do polo}
polo7_imag = 0.1 {Componente imag do polo}

seccao8 = 1 {Tipo para a seccao 8}
zero8_real = 1 {Componente real do zero}
zero8_imag = 1 {Componente imag do zero}
polo8_real = 0.1 {Componente real do polo}
polo8_imag = 0.1 {Componente imag do polo}

seccao9 = 1 {Tipo para a seccao 9}
zero9_real = 1 {Componente real do zero}
zero9_imag = 1 {Componente imag do zero}
```

```
polo9_real = 0.1 {Componente real do polo}
polo9_imag = 0.1 {Componente imag do polo}

seccao10 = 1 {Tipo para a seccao 10}
zero10_real = 1 {Componente real do zero}
zero10_imag = 1 {Componente imag do zero}
polo10_real = 0.1 {Componente real do polo}
polo10_imag = 0.1 {Componente imag do polo}
```

```
#include "FDDEFS.h"

/*****************************************************************/
/*                                                               */
/*     Method Name: cartesoide                                   */
/*     Filter Type: SOS                                          */
/*     Date: Tue Jun  1 16:28:26 1993                            */
/*                                                               */
/*****************************************************************/

/*
 * FUNCTION: cartesoide(sample_freq, ganho, num_seccoes, seccao1, zero1_real, zero1_imag, polo1_real, polo1_imag, seccao2, zero2_real,
zero2_imag, polo2_real, polo2_imag, seccao3, zero3_real, zero3_imag, polo3_real, polo3_imag, seccao4, zero4_real, zero4_imag, polo4_real,
polo4_imag, seccao5, zero5_real, zero5_imag, polo5_real, polo5_imag, seccao6, zero6_real, zero6_imag, polo6_real, polo6_imag, seccao7,
zero7_real, zero7_imag, polo7_real, polo7_imag, seccao8, zero8_real, zero8_imag, polo8_real, polo8_imag, seccao9, zero9_real, zero9_imag,
polo9_real, polo9_imag, seccao10, zero10_real, zero10_imag, polo10_real, polo10_imag)
 * DESCRIPTION:
 *
 *  Sets the coefficients values based on the
 *  passed parameters.
 *
 * RETURN VALUE: SYS_OK or SYS_FATAL
 *
 */

cartesoide(sample_freq, ganho, num_seccoes, seccao1, zero1_real, zero1_imag, polo1_real, polo1_imag, seccao2, zero2_real, zero2_imag,
polo2_real, polo2_imag, seccao3, zero3_real, zero3_imag, polo3_real, polo3_imag, seccao4, zero4_real, zero4_imag, polo4_real, polo4_imag,
seccao5, zero5_real, zero5_imag, polo5_real, polo5_imag, seccao6, zero6_real, zero6_imag, polo6_real, polo6_imag, seccao7, zero7_real,
zero7_imag, polo7_real, polo7_imag, seccao8, zero8_real, zero8_imag, polo8_real, polo8_imag, seccao9, zero9_real, zero9_imag, polo9_real,
polo9_imag, seccao10, zero10_real, zero10_imag, polo10_real, polo10_imag)
double sample_freq;
double ganho;
double num_seccoes;
double seccao1;
double zero1_real;
double zero1_imag;
double polo1_real;
double polo1_imag;
double seccao2;
double zero2_real;
double zero2_imag;
double polo2_real;
double polo2_imag;
double seccao3;
double zero3_real;
double zero3_imag;
double polo3_real;
double polo3_imag;
double seccao4;
double zero4_real;
double zero4_imag;
double polo4_real;
double polo4_imag;
double seccao5;
double zero5_real;
double zero5_imag;
double polo5_real;
double polo5_imag;
double seccao6;
double zero6_real;
double zero6_imag;
double polo6_real;
double polo6_imag;
double seccao7;
double zero7_real;
double zero7_imag;
double polo7_real;
double polo7_imag;
double seccao8;
double zero8_real;
double zero8_imag;
double polo8_real;
double polo8_imag;
double seccao9;
double zero9_real;
double zero9_imag;
double polo9_real;
double polo9_imag;
double seccao10;
double zero10_real;
double zero10_imag;
double polo10_real;
double polo10_imag;
{
  double **B;  /* two dimensional array of numerator coefficients */
  double **A;  /* two dimensional array of denominator coefficients */
```

```
int *STYLES; /* one dimensional array of section styles */
int num_sections;
double gain;
int i;   /* Contador de utilizacao geral  */
double **polos;  /* Array que conter  as coordenadas cartesianas dos polos */
double **zeros;  /* Array que conter  as coordenadas cartesianas dos zeros */

num_sections = num_seccoes;
if (num_sections > 10) {
    num_sections = 10;
    fd_message("O numero de seccoes foi reduzido para 10!");
    }

/*
 * This section is used to allocate memory for the arrays of
 * coefficients.  The value of 'num_sections' must be set before
 * the following code is executed.
 */

if (sos_allocate_memory(num_sections, &B, &A, &STYLES) != SYS_OK) {
   fd_message("Nao ha espaco para guardar os coeficientes!");
   return(SYS_FATAL);
   }

if (sos_allocate_memory(11, &polos, &zeros) != SYS_OK) {
    fd_message("Nao ha espaco para variaveis auxiliares!");
    return(SYS_FATAL);
    }

/*
 * Optionally, use this section to set the section style types.
 * The legal types are: SECOND_ORDER, FIRST_ORDER, ALL_POLE, and ALL_ZERO.
 * Note: The default type is SECOND_ORDER.
 * Example setting:
 *   STYLES[0] = FIRST_ORDER;
 */

 /*  AJUSTE DE VARIAVEIS AUXILIARES  */

    polos[0][0]=polo1_real;
    polos[0][1]=polo1_imag;
    polos[1][0]=polo2_real;
    polos[1][1]=polo2_imag;
    polos[2][0]=polo3_real;
    polos[2][1]=polo3_imag;
    polos[3][0]=polo4_real;
    polos[3][1]=polo4_imag;
    polos[4][0]=polo5_real;
    polos[4][1]=polo5_imag;
    polos[5][0]=polo6_real;
    polos[5][1]=polo6_imag;
    polos[6][0]=polo7_real;
    polos[6][1]=polo7_imag;
    polos[7][0]=polo8_real;
    polos[7][1]=polo8_imag;
    polos[8][0]=polo9_real;
    polos[8][1]=polo9_imag;
    polos[9][0]=polo10_real;
    polos[9][1]=polo10_imag;

    zeros[0][0]=zero1_real;
    zeros[0][1]=zero1_imag;
    zeros[1][0]=zero2_real;
    zeros[1][1]=zero2_imag;
    zeros[2][0]=zero3_real;
    zeros[2][1]=zero3_imag;
    zeros[3][0]=zero4_real;
    zeros[3][1]=zero4_imag;
    zeros[4][0]=zero5_real;
    zeros[4][1]=zero5_imag;
    zeros[5][0]=zero6_real;
    zeros[5][1]=zero6_imag;
    zeros[6][0]=zero7_real;
    zeros[6][1]=zero7_imag;
    zeros[7][0]=zero8_real;
    zeros[7][1]=zero8_imag;
    zeros[8][0]=zero9_real;
    zeros[8][1]=zero9_imag;
    zeros[9][0]=zero10_real;
    zeros[9][1]=zero10_imag;

    STYLES[0] = seccao1;
    STYLES[1] = seccao2;
    STYLES[2] = seccao3;
    STYLES[3] = seccao4;
    STYLES[4] = seccao5;
```

```
    STYLES[5] = seccao6;
    STYLES[6] = seccao7;
    STYLES[7] = seccao8;
    STYLES[8] = seccao9;
    STYLES[9] = seccao10;

    for (i=0;i<num_sections;i++) {
      if (STYLES[i]>4 || STYLES[i]<1) STYLES[i]=1;
      switch(STYLES[i]) {

        case 1:
            STYLES[i] = SECOND_ORDER;

            B[i][0]=1;
            B[i][1]=-2*zeros[i][0];
            B[i][2]=zeros[i][0]*zeros[i][0]+zeros[i][1]*zeros[i][1];

            A[i][0]=1;
            A[i][1]=-2*polos[i][0];
            A[i][2]=polos[i][0]*polos[i][0]+polos[i][1]*polos[i][1];
            break;

        case 2:
            STYLES[i] = FIRST_ORDER;

            B[i][0]=1;
            B[i][1]=-(zeros[i][0]+zeros[i][1]); /* Assume-se que uma das componentes , nula */

            if(zeros[i][0]!=0 && zeros[i][1]!=0)
                fd_message("Erro: os zeros de 1¦ ordem so podem ter uma componente!");

            A[i][0]=1;
            A[i][1]=-(polos[i][0]+polos[i][1]); /* Assume-se que uma das componentes , nula */

            if(polos[i][0]!=0 && polos[i][1]!=0)
                fd_message("Erro: os polos de 1¦ ordem so podem ter uma componente!");

            break;
        case 3:
            STYLES[i] = ALL_POLE;

            B[i][0]=1;
            B[i][1]=0;
            B[i][2]=0;

            A[i][0]=1;
            A[i][1]=-2*polos[i][0];
            A[i][2]=polos[i][0]*polos[i][0]+polos[i][1]*polos[i][1];
            break;
        case 4:
            STYLES[i] = ALL_ZERO;

            B[i][0]=1;
            B[i][1]=-2*zeros[i][0];
            B[i][2]=zeros[i][0]*zeros[i][0]+zeros[i][1]*zeros[i][1];

            A[i][0]=1;
            A[i][1]=0;
            A[i][2]=0;
            break;
        }
    }


  /*
   * Use this section to assign values to the coefficient arrays.
   * For example:
   *   B[0][0] = 3.14;
   */

   /* Use this section to set the gain value */
  gain = ganho;

  if (sos_c_save(num_sections, B, A, STYLES, gain) != SYS_OK) {
    fd_message("Nao consigo gravar os resultados!");
    return(SYS_FATAL);
    }

  fd_message("O metodo foi concluido com sucesso!");
  return(SYS_OK);
}
```

```
static char * FD_CODE_REV = "1.0";

/*
 * FILE: main.c
 */

#include <stdio.h>
#include "FDDEFS.h"

main(argc, argv)
int argc;
char **argv;
{
  FILE *fp_params;
  int pid;
  double param1;
  double param2;
  double param3;
  double param4;
  double param5;
  double param6;
  double param7;
  double param8;
  double param9;
  double param10;
  double param11;
  double param12;
  double param13;
  double param14;
  double param15;
  double param16;
  double param17;
  double param18;
  double param19;
  double param20;
  double param21;
  double param22;
  double param23;
  double param24;
  double param25;
  double param26;
  double param27;
  double param28;
  double param29;
  double param30;
  double param31;
  double param32;
  double param33;
  double param34;
  double param35;
  double param36;
  double param37;
  double param38;
  double param39;
  double param40;
  double param41;
  double param42;
  double param43;
  double param44;
  double param45;
  double param46;
  double param47;
  double param48;
  double param49;
  double param50;
  double param51;
  double param52;
  double param53;

  fdParamsOpen(argv[1], &fp_params);

  fdParseDouble(fp_params, &param1);
  fdParseDouble(fp_params, &param2);
  fdParseDouble(fp_params, &param3);
  fdParseDouble(fp_params, &param4);
  fdParseDouble(fp_params, &param5);
  fdParseDouble(fp_params, &param6);
  fdParseDouble(fp_params, &param7);
  fdParseDouble(fp_params, &param8);
  fdParseDouble(fp_params, &param9);
  fdParseDouble(fp_params, &param10);
  fdParseDouble(fp_params, &param11);
  fdParseDouble(fp_params, &param12);
  fdParseDouble(fp_params, &param13);
  fdParseDouble(fp_params, &param14);
  fdParseDouble(fp_params, &param15);
```

```
  fdParseDouble(fp_params, &param16);
  fdParseDouble(fp_params, &param17);
  fdParseDouble(fp_params, &param18);
  fdParseDouble(fp_params, &param19);
  fdParseDouble(fp_params, &param20);
  fdParseDouble(fp_params, &param21);
  fdParseDouble(fp_params, &param22);
  fdParseDouble(fp_params, &param23);
  fdParseDouble(fp_params, &param24);
  fdParseDouble(fp_params, &param25);
  fdParseDouble(fp_params, &param26);
  fdParseDouble(fp_params, &param27);
  fdParseDouble(fp_params, &param28);
  fdParseDouble(fp_params, &param29);
  fdParseDouble(fp_params, &param30);
  fdParseDouble(fp_params, &param31);
  fdParseDouble(fp_params, &param32);
  fdParseDouble(fp_params, &param33);
  fdParseDouble(fp_params, &param34);
  fdParseDouble(fp_params, &param35);
  fdParseDouble(fp_params, &param36);
  fdParseDouble(fp_params, &param37);
  fdParseDouble(fp_params, &param38);
  fdParseDouble(fp_params, &param39);
  fdParseDouble(fp_params, &param40);
  fdParseDouble(fp_params, &param41);
  fdParseDouble(fp_params, &param42);
  fdParseDouble(fp_params, &param43);
  fdParseDouble(fp_params, &param44);
  fdParseDouble(fp_params, &param45);
  fdParseDouble(fp_params, &param46);
  fdParseDouble(fp_params, &param47);
  fdParseDouble(fp_params, &param48);
  fdParseDouble(fp_params, &param49);
  fdParseDouble(fp_params, &param50);
  fdParseDouble(fp_params, &param51);
  fdParseDouble(fp_params, &param52);
  fdParseDouble(fp_params, &param53);

  fclose(fp_params);

  pid = atoi(argv[2]);

  fdFilesOpen(pid);

  if (cartesoide(param1, param2, param3, param4, param5, param6, param7, param8, param9, param10, param11, param12, param13,
param14, param15, param16, param17, param18, param19, param20, param21, param22, param23, param24, param25, param26, param27,
param28, param29, param30, param31, param32, param33, param34, param35, param36, param37, param38, param39, param40, param41,
param42, param43, param44, param45, param46, param47, param48, param49, param50, param51, param52, param53) !=SYS_OK)
    fd_fatal();

  fdFilesClose();

  exit(0);
}
```

ANEXO D: "EXEMPLO DE INTERACÇÃO"

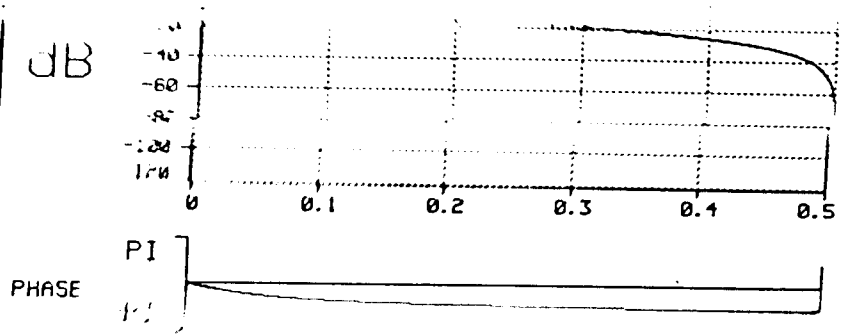
FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_1

Um polo em 0.7 [0] e um zero em 1 [180]

sample freq = 1 {Frequencia de Amostr
gain = 0.1[illegible] {Ganho do Filtro
num seccoes = 1 {Limite maximo de 10}
seccao1 = 2 {Tipo 1:Segunda Ordem 2:P
zero1_real = -1 {Componente real do z
zero1_imag = 0 {Componente imag do ze
polo1_real = 0.7 {Componente real do
polo1_imag = 0 {Componente imag do po

FREQUENCY RESPONSE



Point = 111 of 1000
Freq. = 0.[illegible]
dB = [illegible]
Phase = -0.784991

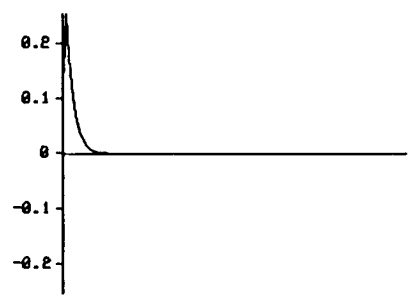
SOS FILTER COEFFICIENTS

Num. of sections = 1 Gain = 0.15

Sections

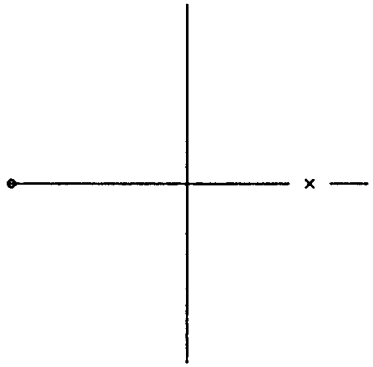
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

IMPULSE RESPONSE



Point = 1 of 128
Time = 1 secs
Value = 0.255

POLE ZERO



# of Poles = 1
# of Zeros = 1
Pole
Real = 0.7
Imag.= 0
Radius = 0.7
Angle = 0

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_1
Method = valtermeths/cartesoide

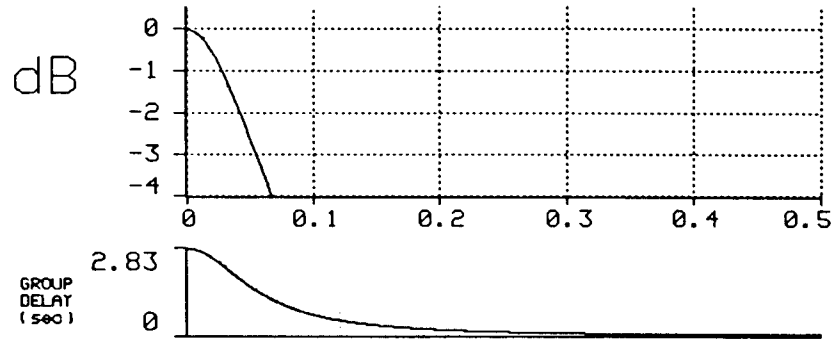
COMMENT

Exemplo 2
FASE:1
Um polo em 0.7 [0] e um zero em 1 [180]

PARAMETERS

```
sample_freq = 1            {Frequencia de Amostr
ganho = 0.1499995651           {Ganho do filtro
num_seccoes = 1            {Limite maximo de 10}
seccao1 = 2            {Tipo 1:Segunda Ordem 2:P
zero1_real = -1            {Componente real do z
zero1_imag = 0             {Componente imag do ze
polo1_real = 0.7             {Componente real do
polo1_imag = 0             {Componente imag do po
seccao2 = 1            {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 111 of 1000
Freq. = 0.0555556
dB = -3.00677
Delay = 1.4619

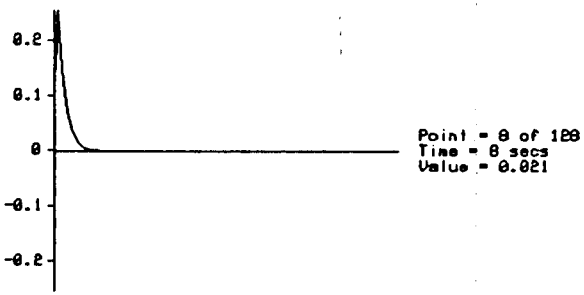
SOS FILTER COEFFICIENTS

Num. of sections = 1 Gain = 0.15

Sections

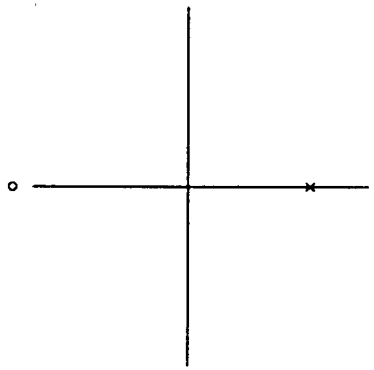
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

IMPULSE RESPONSE



Point = 0 of 100
Time = 0 secs
Value = 0.021

POLE ZERO



# of Poles = 1
# of Zeros = 1
Zero
Real = -1
Imag. = 0
Radius = 1
Angle = 180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_2
Method = valtermeths/cartesoide

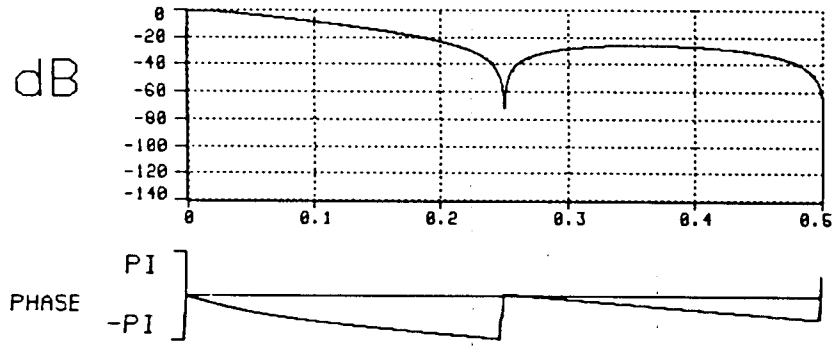
COMMENT

Exemplo: 2
FASE: 2
Mais um par de zeros em 1 [90]

PARAMETERS

sample_freq = 1	{Frequencia de Amostr
ganho = 0.074999781	{Ganho do filtro}
num_seccoes = 2	{Limite maximo de 10}
seccao1 = 2	{Tipo 1:Segunda Ordem 2:P
zero1_real = -1	{Componente real do z
zero1_imag = 0	{Componente imag do ze
polo1_real = 0.7	{Componente real do
polo1_imag = 0	{Componente imag do po
seccao2 = 4	{Tipo para a seccao 2}

FREQUENCY RESPONSE



Point = 100 of 1000
Freq. = 0.0500501
dB = -3.00677
Phase = -1.04642

SOS FILTER COEFFICIENTS

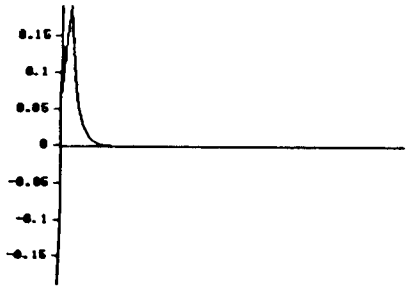
Num. of sections = 2 Gain = 0.0749998

Sections

Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

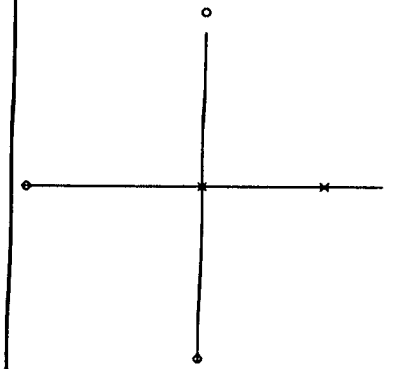
Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

IMPULSE RESPONSE



Point = 3 of 128
Time = 3 secs
Value = 0.18997

POLE ZERO



# of Poles = 3
# of Zeros = 3
Zero
Real = 0
Imag.= 1
Radius = 1
Angle = 90

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_2
Method = valtermeths/cartesoide

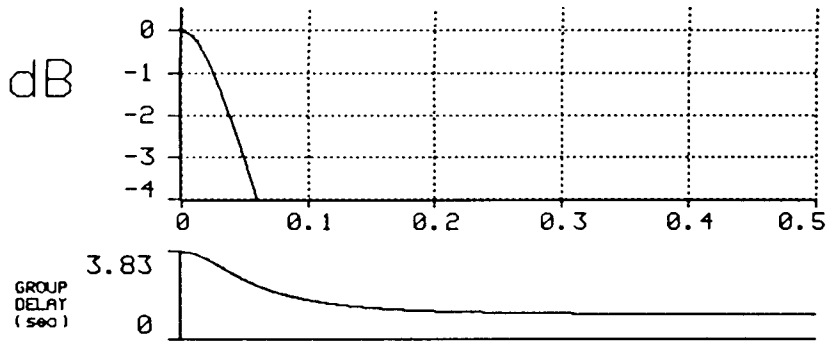
COMMENT

Exemplo: 2
FASE: 2
Mais um par de zeros em 1 [90]

PARAMETERS

```
sample_freq = 1          {Frequencia de Amostr
ganho = 0.074999781          {Ganho do filtro}
num_seccoes = 2          {Limite maximo de 10}
seccao1 = 2          {Tipo 1:Segunda Ordem 2:P
zero1_real = -1          {Componente real do z
zero1_imag = 0          {Componente imag do ze
polo1_real = 0.7          {Componente real do
polo1_imag = 0          {Componente imag do po
seccao2 = 4          {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 100 of 1000
Freq. = 0.0500501
dB = -3.00677
Delay = 2.60724

SOS FILTER COEFFICIENTS

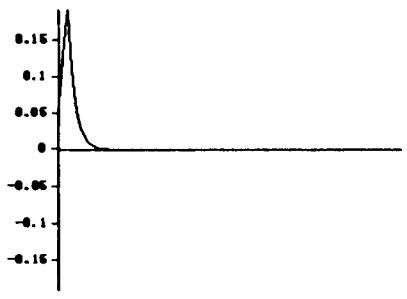
Num. of sections = 2 Gain = 0.0749998

Sections

Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

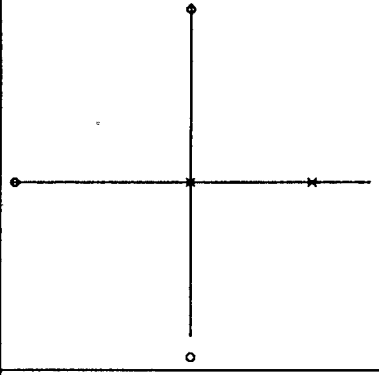
Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

IMPULSE RESPONSE



Point = 3 of 128
Time = 3 secs
Value = 0.18997

POLE ZERO



# of Poles = 3
# of Zeros = 3
Zero
Real = 0
Imag. = -1
Radius = 1
Angle = -90

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_3
Method = valtermeths/cartesoide

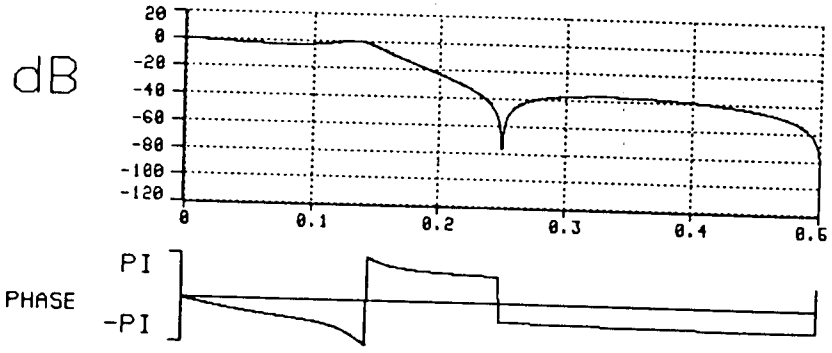
COMMENT

Exemplo: 2
FASE: 3
Mais um par de polos em 0.9 [50]

PARAMETERS

sample_freq = 1	{Frequencia de Amostr
ganho = 0.048973934	{Ganho do filtro}
num_seccoes = 3	{Limite maximo de 10}
seccao1 = 2	{Tipo 1:Segunda Ordem 2:P
zero1_real = -1	{Componente real do z
zero1_imag = 0	{Componente imag do ze
polo1_real = 0.7	{Componente real do
polo1_imag = 0	{Componente imag do po
seccao2 = 4	{Tipo para a seccao 2}

FREQUENCY RESPONSE



Point = 299 of 1000
Freq. = 0.14965
dB = -2.97376
Phase = 2.94591

SOS FILTER COEFFICIENTS

Num. of sections = 3 Gain = 0.0489739

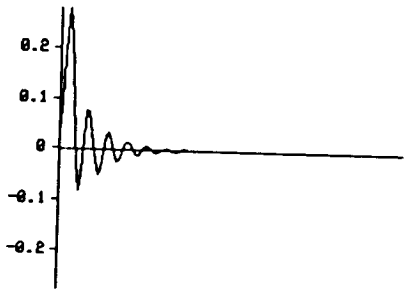
Sections

Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

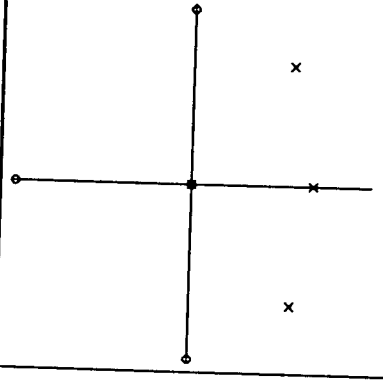
Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697

IMPULSE RESPONSE



Point = 2 of 128
Time = 2 secs
Value = 0.22947

POLE ZERO



# of Poles = 5
# of Zeros = 5
Pole
Real = 0.57851
Imag.= 0.68944
Radius = 0.9
Angle = 50

FDS 3.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_3
Method = valtermeths/cartesoide

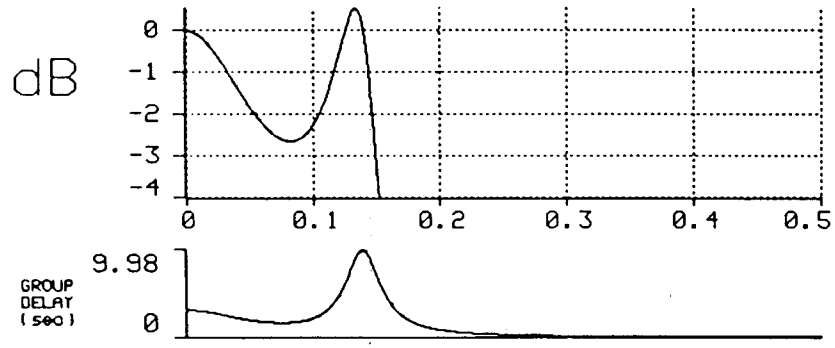
COMMENT

Exemplo: 2
FASE: 3
Mais um par de polos em 0.9 [50]

PARAMETERS

```
zero2_real = 0            {Componente real do ze
zero2_imag = 1            {Componente imag do ze
polo2_real = 0            {Componente real do po
polo2_imag = 0            {Componente imag do po
seccao3 = 3            {Tipo para a seccao 3}
zero3_real = 0            {Componente real do ze
zero3_imag = 0            {Componente imag do ze
polo3_real = 0.5785088487          {Componente
polo3_imag = 0.689439988           {Componente
```

FREQUENCY RESPONSE



Point = 268 of 1000
Freq. = 0.134134
dB = 0.542685
Delay = 9.2962

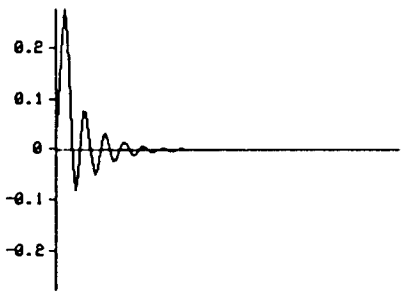
SOS FILTER COEFFICIENTS

Num. of sections = 3 Gain = 0.0489739

Sections

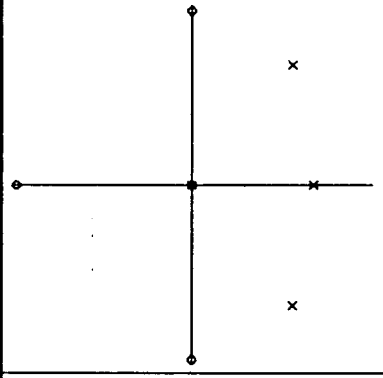
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
```

IMPULSE RESPONSE



Point = 2 of 128
Time = 2 secs
Value = 0.22947

POLE ZERO



# of Poles = 5
# of Zeros = 5
Pole
Real = 0.57851
Imag.= -0.68944
Radius = 0.9
Angle = -50

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_3
Method = valtermeths/cartesoide

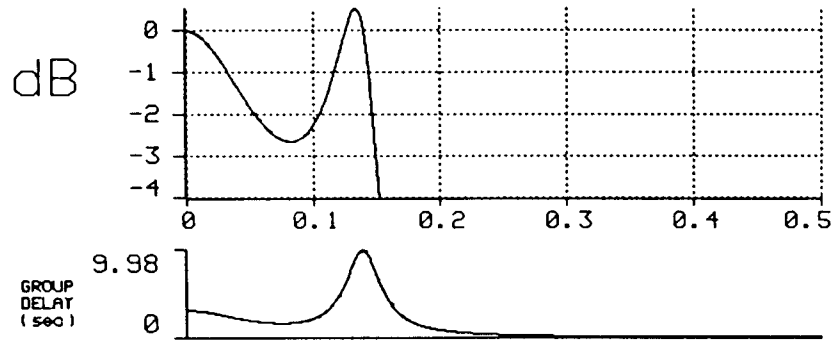
COMMENT

Exemplo: 2
FASE: 3
Mais um par de polos em 0.9 [50]

PARAMETERS

```
zero2_real = 0          {Componente real do ze
zero2_imag = 1          {Componente imag do ze
polo2_real = 0          {Componente real do po
polo2_imag = 0          {Componente imag do po
seccao3 = 3          {Tipo para a seccao 3}
zero3_real = 0          {Componente real do ze
zero3_imag = 0          {Componente imag do ze
polo3_real = 0.5785088487          {Componente
polo3_imag = 0.609439988          {Componente
```

FREQUENCY RESPONSE



Point = 268 of 1000
Freq. = 0.134134
dB = 0.542685
Delay = 9.2962

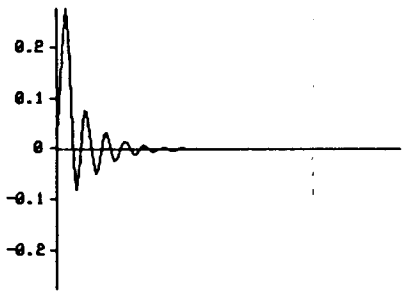
SOS FILTER COEFFICIENTS

Num. of sections = 3 Gain = 0.0489739

Sections

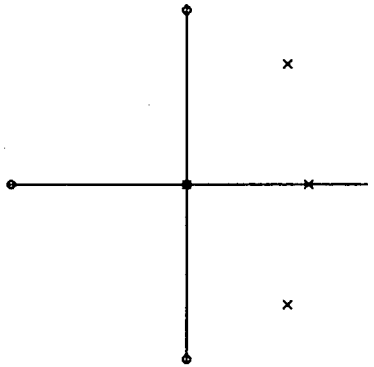
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
```

IMPULSE RESPONSE



Point = 2 of 128
Time = 2 secs
Value = 0.22947

POLE ZERO



# of Poles = 5
# of Zeros = 5
Pole
Real = 0.57851
Imag.= -0.68944
Radius = 0.9
Angle = -50

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_4
Method = valtermeths/cartesoide

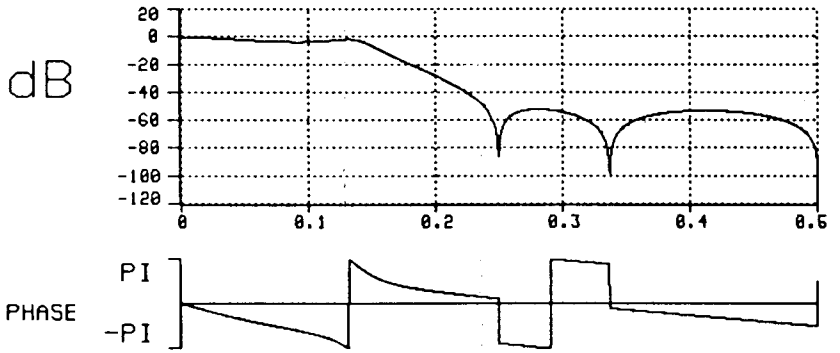
COMMENT

Exemplo:2
FASE:4
Mais um par de zeros em 1 [121]

PARAMETERS

```
sample_freq = 1          {Frequencia de Amostrag
ganho = 0.016162554          {Ganho do Filtro}
num_seccoes = 4          {Limite maximo de 10}
seccao1 = 2          {Tipo 1:Segunda Ordem 2:Pri
zero1_real = -1          {Componente real do zer
zero1_imag = 0          {Componente imag do zero
polo1_real = 0.7          {Componente real do po
polo1_imag = 0          {Componente imag do polo
seccao2 = 4          {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 285 of 1000
Freq. = 0.142643
dB = -3.08338
Phase = 2.41981

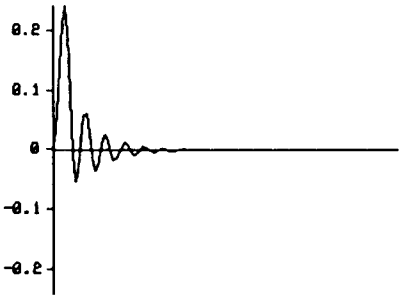
SOS FILTER COEFFICIENTS

Num. of sections = 4 Gain = 0.0161626

Sections

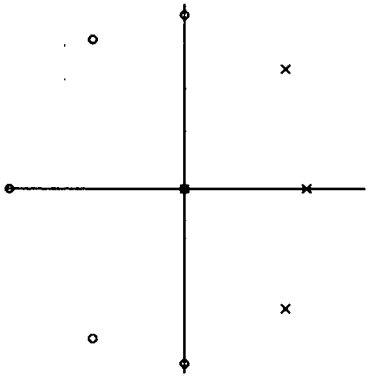
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
a[2][2] = 0.8099999851
```

IMPULSE RESPONSE



Point = 4 of 128
Time = 4 secs
Value = 0.24242

POLE ZERO



# of Poles = 7
# of Zeros = 7
Zero
Real = -0.51504
Imag.= -0.85717
Radius = 1
Angle = -121

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_4
Method = valtermeths/cartesoide

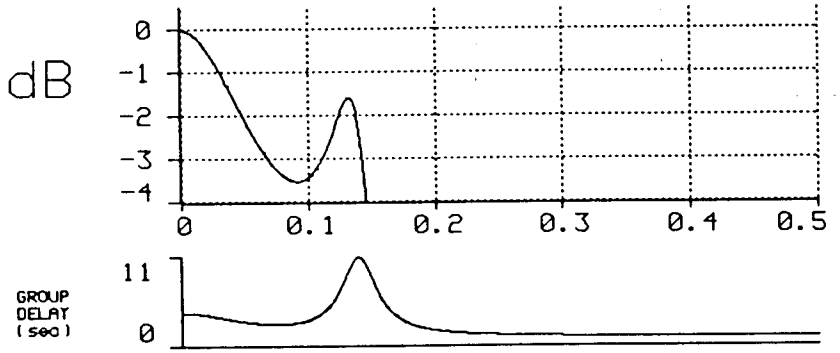
COMMENT

Exemplo:2
FASE:4
Mais um par de zeros em 1 [121]

PARAMETERS

```
sample_freq = 1          {Frequencia de Amostrag
ganho = 0.016162554          {Ganho do filtro}
num_seccoes = 4          {Limite maximo de 10}
seccao1 = 2          {Tipo 1:Segunda Ordem 2:Pri
zero1_real = -1          {Componente real do zer
zero1_imag = 0          {Componente imag do zero
polo1_real = 0.7          {Componente real do po
polo1_imag = 0          {Componente imag do polo
seccao2 = 4          {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 266 of 1000
Freq. = 0.133133
dB = -1.59042
Delay = 10.0078

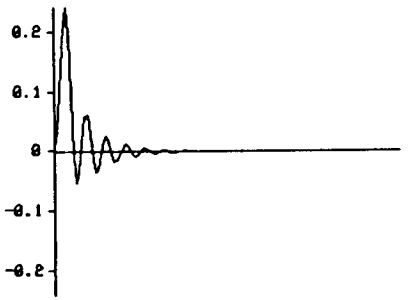
SOS FILTER COEFFICIENTS

Num. of sections = 4 Gain = 0.0161626

Sections

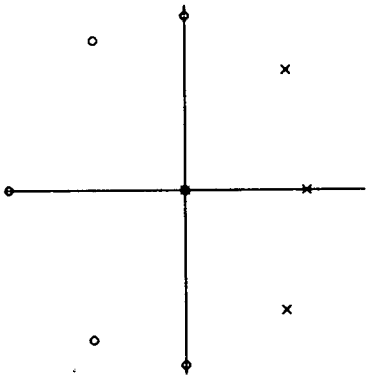
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
a[2][2] = 0.8099999851
```

IMPULSE RESPONSE



Point = 4 of 128
Time = 4 secs
Value = 0.24242

POLE ZERO



# of Poles = 7
# of Zeros = 7
Zero
Real = -0.51504
Imag.= 0.85717
Radius = 1
Angle = 121

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_5
Method = valtermeths/cartesoide

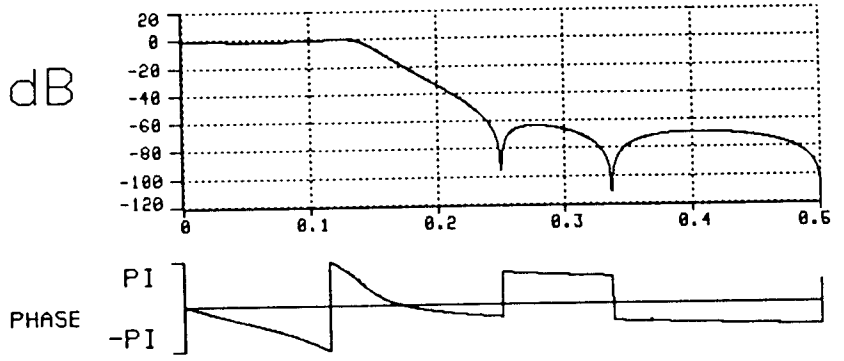
COMMENT

Exemplo:2
FASE:5
Mais um par de polos em 0.75 [40]

PARAMETERS

```
seccao4 = 4          {Tipo para a seccao 4}
zero4_real = -0.5150380749          {Componente
zero4_imag = 0.8571673007          {Componente i
polo4_real = 0.1          {Componente real do po
polo4_imag = 0.1          {Componente imag do po
seccao5 = 3          {Tipo para a seccao 5}
zero5_real = 0          {Componente real do zero
zero5_imag = 0          {Componente imag do zero
polo5_real = 0.5745333323          {Componente r
```

FREQUENCY RESPONSE



Point = 288 of 1000
Freq. = 0.144144
dB = -3.07204
Phase = 1.1648

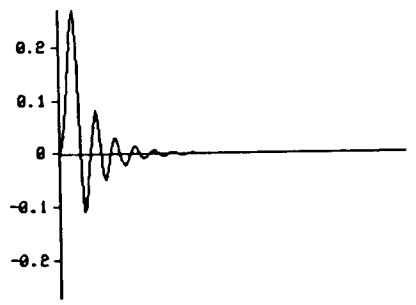
SOS FILTER COEFFICIENTS

Num. of sections = 5 Gain = 0.00668213

Sections

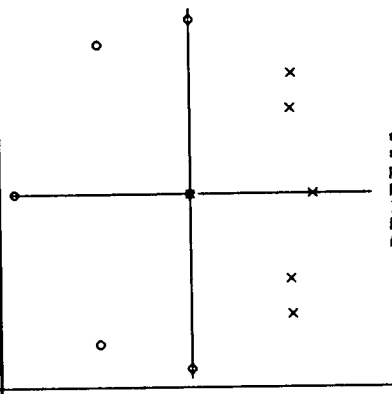
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
a[2][2] = 0.8099999851
```

IMPULSE RESPONSE



Point = 5 of 128
Time = 5 secs
Value = 0.27034

POLE ZERO



# of Poles = 9
# of Zeros = 9
Pole
Real = 0.57453
Imag.= -0.48209
Radius = 0.75
Angle = -40

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_5
Method = valtermeths/cartesoide

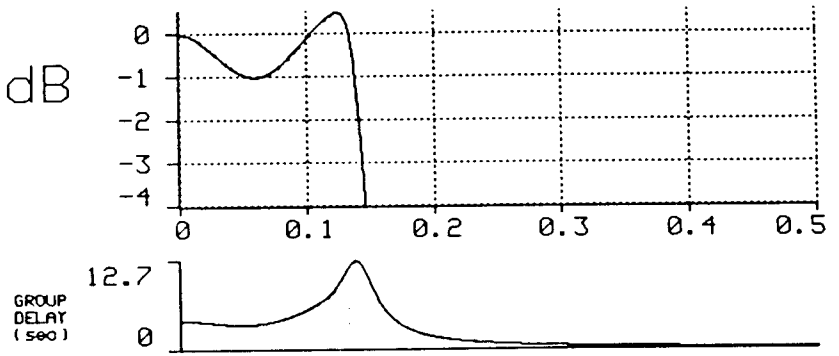
COMMENT

Exemplo:2
FASE:5
Mais um par de polos em 0.75 [40]

PARAMETERS

```
seccao4 = 4          {Tipo para a seccao 4}
zero4_real = -0.5150380749          {Componente
zero4_imag = 0.8571673007          {Componente i
polo4_real = 0.1          {Componente real do pó
polo4_imag = 0.1          {Componente imag do po
seccao5 = 3          {Tipo para a seccao 5}
zero5_real = 0          {Componente real do zero
zero5_imag = 0          {Componente imag do zero
polo5_real = 0.5745333323          {Componente r
```

FREQUENCY RESPONSE



Point = 249 of 1000
Freq. = 0.124625
dB = 0.514362
Delay = 9.45076

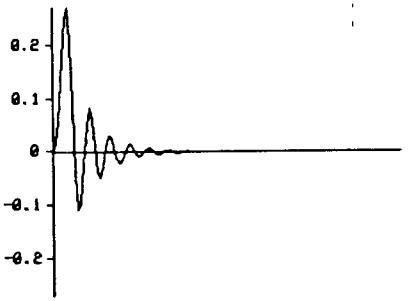
SOS FILTER COEFFICIENTS

Num. of sections = 5 Gain = 0.00668213

Sections

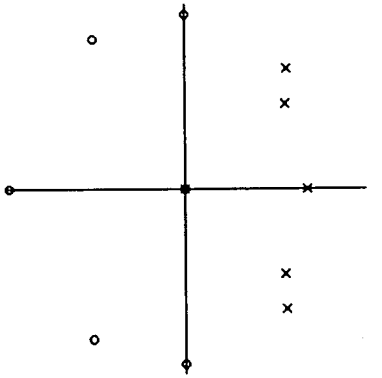
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
a[2][2] = 0.8099999851
```

IMPULSE RESPONSE



Point = 5 of 128
Time = 5 secs
Value = 0.27034

POLE ZERO



# of Poles = 9
# of Zeros = 9
Pole
Real = 0.57453
Imag.= 0.48209
Radius = 0.75
Angle = 40

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_6
Method = valtermeths/cartesoide

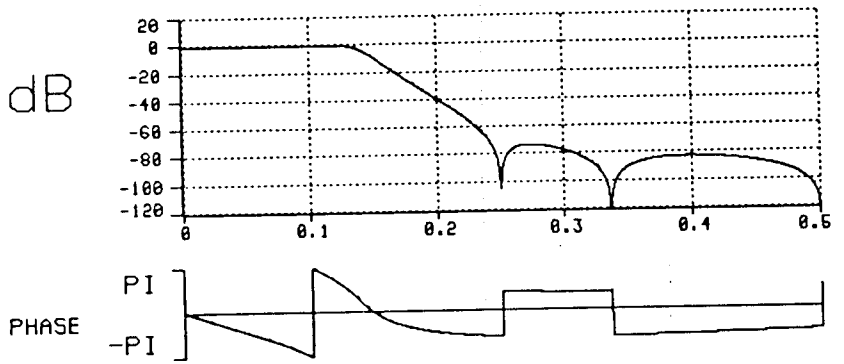
COMMENT

Exemplo:2
FASE:6
Mais um par de polos em 0.6 [38]

PARAMETERS

```
sample_freq = 1          {Frequencia de Amostrag
ganho = 0.002768981          {Ganho do Filtro}
num_seccoes = 6            {Limite maximo de 10}
seccao1 = 2          {Tipo 1:Segunda Ordem 2:Pri
zero1_real = -1          {Componente real do zer
zero1_imag = 0          {Componente imag do zero
polo1_real = 0.7          {Componente real do po
polo1_imag = 0          {Componente imag do polo
seccao2 = 4          {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 279 of 1000
Freq. = 0.13964
dB = -2.91449
Phase = 0.662119

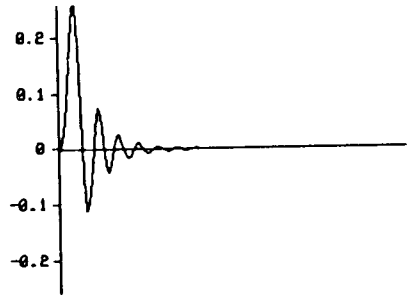
SOS FILTER COEFFICIENTS

Num. of sections = 6 Gain = 0.00276898

Sections

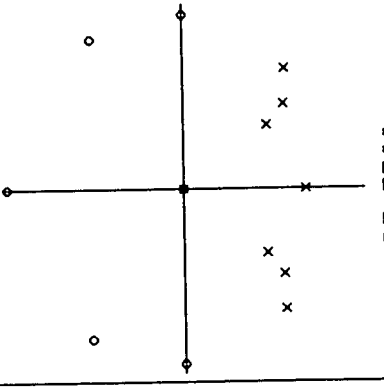
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
a[2][2] = 0.8099999851
```

IMPULSE RESPONSE



Point = 6 of 128
Time = 6 secs
Value = 0.25702

POLE ZERO



# of Poles = 11
# of Zeros = 11
Pole
Real = 0.47281
Imag.= 0.3694
Radius = 0.6
Angle = 38

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc251_6
Method = valtermeths/cartesoide

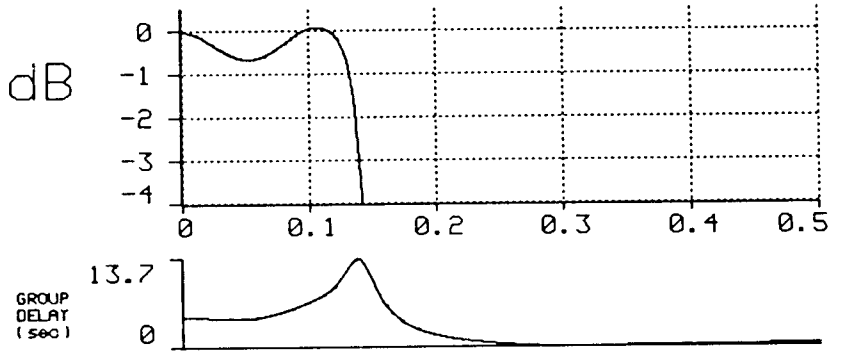
COMMENT

Exemplo:2
FASE:6
Mais um par de polos em 0.6 [38]

PARAMETERS

```
zero5_imag = 0          {Componente imag do zero
polo5_real = 0.5745333323          {Componente r
polo5_imag = 0.4820907073          {Componente i
seccao6 = 3          {Tipo para a seccao 6}
zero6_real = 0          {Componente real do zero
zero6_imag = 0          {Componente imag do zero
polo6_real = 0.4728064522          {Componente r
polo6_imag = 0.3693968852          {Componente i
seccao7 = 1          {Tipo para a seccao 7}
```

FREQUENCY RESPONSE



Point = 279 of 1000
Freq. = 0.13964
dB = -2.91449
Delay = 13.5705

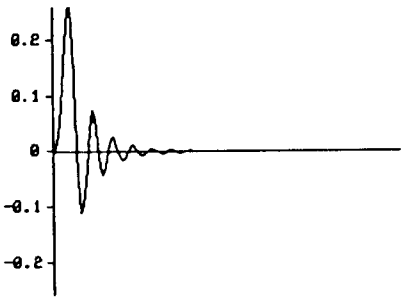
SOS FILTER COEFFICIENTS

Num. of sections = 6 Gain = 0.00276898

Sections

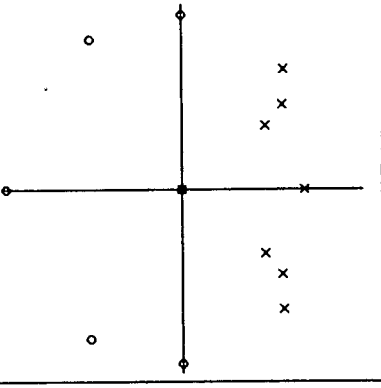
```
Section 0: first order
b[0][0] = 1
b[0][1] = 1
a[0][0] = 1
a[0][1] = -0.7

Section 1: all zero
b[1][0] = 1
b[1][1] = -0
b[1][2] = 1
a[1][0] = 1

Section 2: all pole
b[2][0] = 1
a[2][0] = 1
a[2][1] = -1.157017697
a[2][2] = 0.8099999851
```

IMPULSE RESPONSE



Point = 6 of 128
Time = 6 secs
Value = 0.25702

POLE ZERO



# of Poles = 11
# of Zeros = 11
Pole
Real = 0.47281
Imag.= -0.3694
Radius = 0.6
Angle = -38

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc255_1
Method = valtermeths/cartesoide

COMMENT

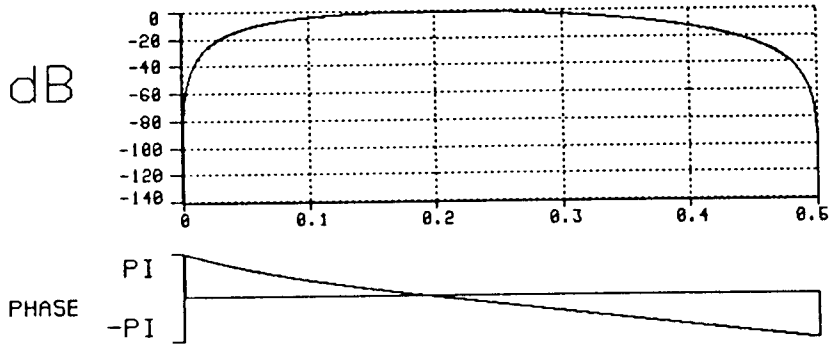
Exemplo: 3
FASE: 1

Polo em 0.7 [93] zeros em 1 [0] e 1 [180]

PARAMETERS

```
sample_freq = 1    {Frequencia de Amostragem}
ganho = 0.2803044548    {Ganho do filtro}
num_seccoes = 3    {Limite maximo de 10}
seccao1 = 2    {Tipo 1:Segunda Ordem 2:Primeir
zero1_real = 0    {Componente real do zero}
zero1_imag = 0    {Componente imag do zero}
polo1_real = -0.036635169    {Componente real
polo1_imag = 0.6990406743    {Componente imag
seccao2 = 4    {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -0.592466
Phase = -0.587712

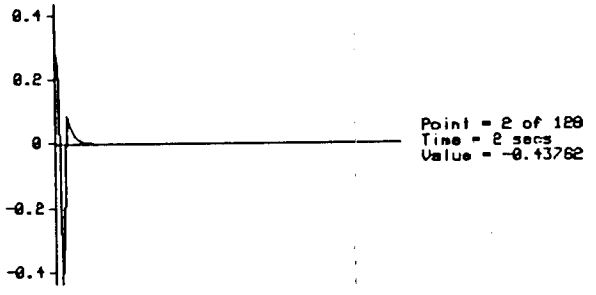
SOS FILTER COEFFICIENTS

Num. of sections = 3 Gain = 0.280304

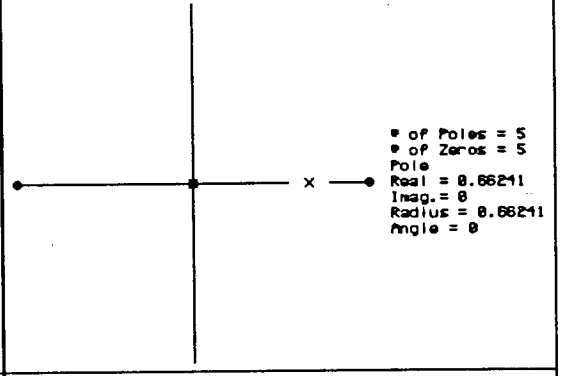
Sections

```
Section 0: first order
b[0][0] = 1
b[0][1] = -0
a[0][0] = 1
a[0][1] = -0.6624055053

Section 1: all zero
b[1][0] = 1
b[1][1] = -2
b[1][2] = 1
a[1][0] = 1

Section 2: all zero
b[2][0] = 1
b[2][1] = 2
b[2][2] = 1
```

IMPULSE RESPONSE



POLE ZERO



FDS 2.8.2 [I] Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc255_1
Method = valtermeths/cartesoide

COMMENT

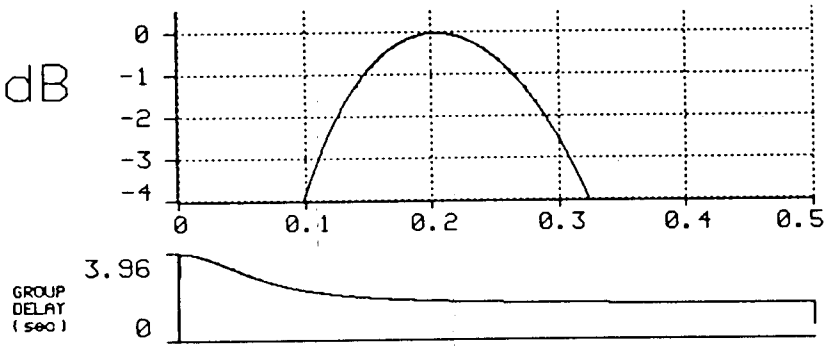
Exemplo: 3
FASE: 1

Polo em 0.7 [93] zeros em 1 [0] e 1 [180]

PARAMETERS

sample_freq = 1 {Frequencia de Amostragem}
ganho = 0.2803044548 {Ganho do filtro}
num_seccoes = 3 {Limite maximo de 10}
seccao1 = 2 {Tipo 1:Segunda Ordem 2:Primeir
zero1_real = 0 {Componente real do zero}
zero1_imag = 0 {Componente imag do zero}
polo1_real = -0.036635169 {Componente real
polo1_imag = 0.6990406743 {Componente imag
seccao2 = 4 {Tipo para a seccao 2}

FREQUENCY RESPONSE



Point = 616 of 1000
Freq. = 0.308308
dB = -3.01706
Delay = 1.64666

SOS FILTER COEFFICIENTS

Num. of sections = 3 Gain = 0.280304

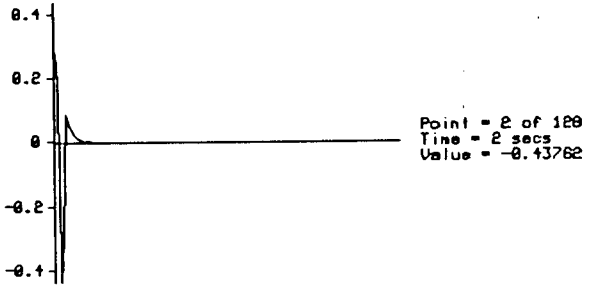
Sections

b[0][0] = 1
b[0][1] = -0
a[0][0] = 1
a[0][1] = -0.6624055053

Section 1: all zero
b[1][0] = 1
b[1][1] = -2
b[1][2] = 1
a[1][0] = 1

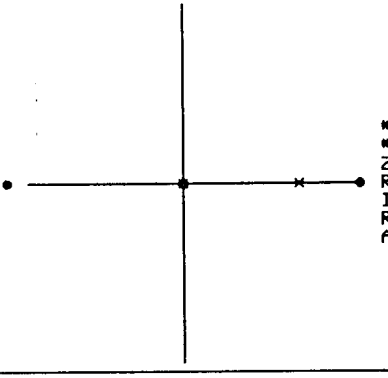
Section 2: all zero
b[2][0] = 1
b[2][1] = 2
b[2][2] = 1
a[2][0] = 1

IMPULSE RESPONSE



Point = 2 of 128
Time = 2 secs
Value = -0.43762

POLE ZERO



# of Poles = 5
# of Zeros = 5
Zero
Real = -1
Imag. = 0
Radius = 1
Angle = 180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc255_2
Method = valtermeths/cartesoide

COMMENT

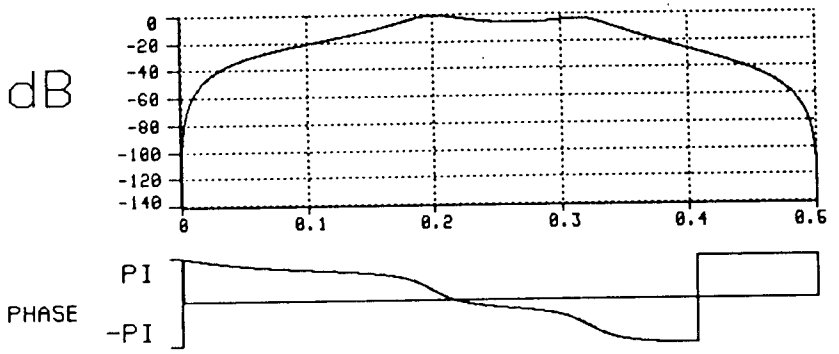
Exemplo: 3
FASE: 2

Polos em 0.88 [71] e em 0.88 [115]

PARAMETERS

```
sample_freq = 1    {Frequencia de Amostragem}
ganho = 0.079250133    {Canho do filtro}
num_seccoes = 5    {Limite maximo de 10}
seccao1 = 2    {Tipo 1:Segunda Ordem 2:Primeir
zero1_real = 0    {Componente real do zero}
zero1_imag = 0    {Componente imag do zero}
polo1_real = -0.036635169    {Componente real
polo1_imag = 0.6990406743    {Componente imag
seccao2 = 4    {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 627 of 1000
Freq. = 0.313814
dB = -3.09447
Phase = -1.61738

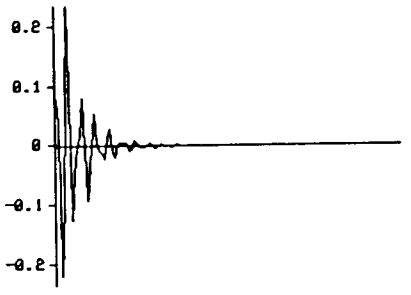
SOS FILTER COEFFICIENTS

Num. of sections = 5 Gain = 0.0792501

Sections

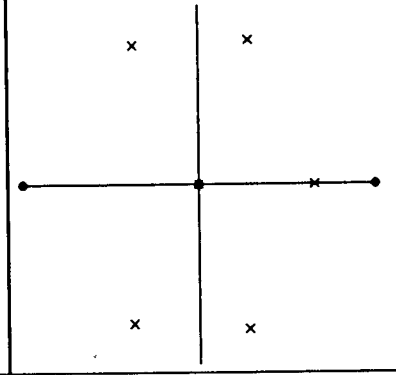
```
b[0][0] = 1
b[0][1] = -0
a[0][0] = 1
a[0][1] = -0.6624055053

Section 1: all zero
b[1][0] = 1
b[1][1] = -2
b[1][2] = 1
a[1][0] = 1

Section 2: all zero
b[2][0] = 1
b[2][1] = 2
b[2][2] = 1
a[2][0] = 1
```

IMPULSE RESPONSE



Point = 4 of 128
Time = 4 secs
Value = 0.23538

POLE ZERO



# of Poles = 9
# of Zeros = 9
Pole
Real = -0.3719
Imag. = 0.79755
Radius = 0.88
Angle = 115

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc255_2
Method = valtermeths/cartesoide

COMMENT

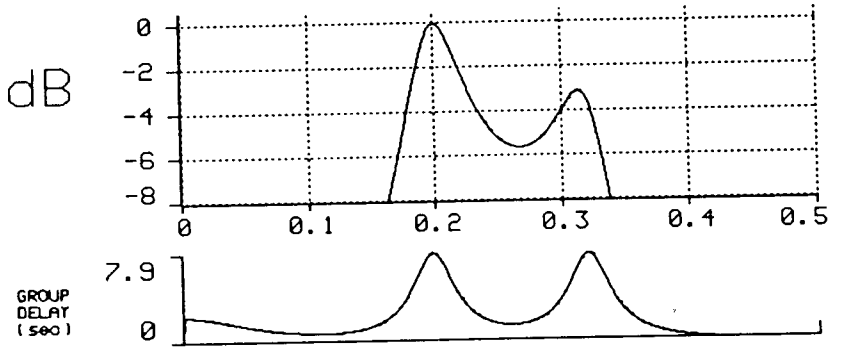
Exemplo: 3
FASE: 2

Polos em 0.88 [71] e em 0.88 [115]

PARAMETERS

zero3_imag = 0 {Componente imag do zero}
polo3_real = 0 {Componente real do polo}
polo3_imag = 0 {Componente imag do polo}
seccao4 = 3 {Tipo para a seccao 4}
zero4_real = 0 {Componente real do zero}
zero4_imag = 0 {Componente imag do zero}
polo4_real = 0.2864999759 {Componente real
polo4_imag = 0.8320563465 {Componente imag
seccao5 = 3 {Tipo para a seccao 5}

FREQUENCY RESPONSE



Point = 401 of 1000
Freq. = 0.200701
dB = -0.010048
Delay = 7.68381

SOS FILTER COEFFICIENTS

Num. of sections = 5 Gain = 0.0792501

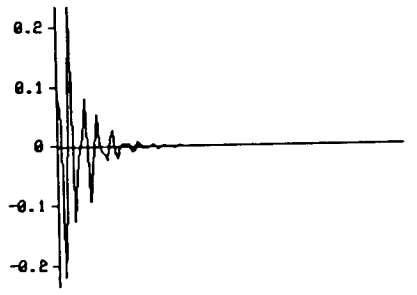
Sections

b[0][0] = 1
b[0][1] = -0
a[0][0] = 1
a[0][1] = -0.6624055053

Section 1: all zero
b[1][0] = 1
b[1][1] = -2
b[1][2] = 1
a[1][0] = 1

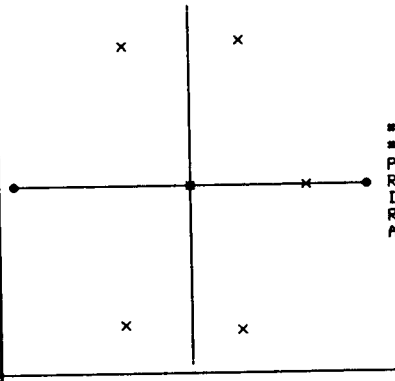
Section 2: all zero
b[2][0] = 1
b[2][1] = 2
b[2][2] = 1
a[2][0] = 1

IMPULSE RESPONSE



Point = 4 of 128
Time = 4 secs
Value = 0.23538

POLE ZERO



# of Poles = 9
# of Zeros = 9
Pole
Real = 0.2865
Imag.= -0.83206
Radius = 0.88
Angle = -71

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc255_3
Method = valtermeths/cartesoide

COMMENT

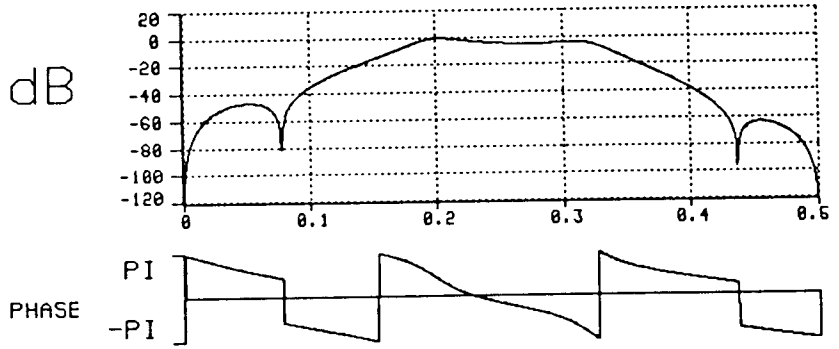
Exemplo: 3
FASE: 3

Zeros em 1 [157] e em 1 [28]

PARAMETERS

```
zero9_real = 1    {Componente real do zero}
zero9_imag = 1    {Componente imag do zero}
polo9_real = 0.1    {Componente real do polo}
polo9_imag = 0.1    {Componente imag do polo}
seccao10 = 1    {Tipo para a seccao 10}
zero10_real = 1    {Componente real do zero}
zero10_imag = 1    {Componente imag do zero}
polo10_real = 0.1    {Componente real do polo}
polo10_imag = 0.1    {Componente imag do polo}
```

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -3.96767
Phase = -0.509817

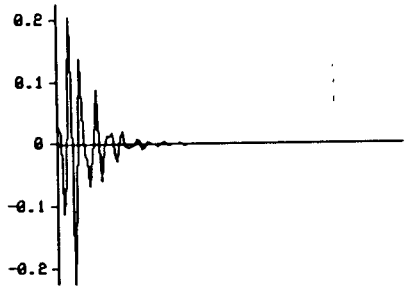
SOS FILTER COEFFICIENTS

Num. of sections = 7 Gain = 0.0280008

Sections

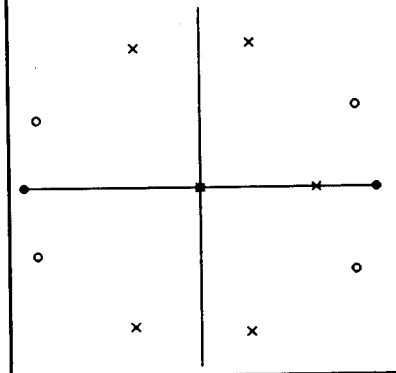
```
b[0][0] = 1
b[0][1] = -0
a[0][0] = 1
a[0][1] = -0.6624055053

Section 1: all zero
b[1][0] = 1
b[1][1] = -2
b[1][2] = 1
a[1][0] = 1

Section 2: all zero
b[2][0] = 1
b[2][1] = 2
b[2][2] = 1
a[2][0] = 1
```

IMPULSE RESPONSE



Point = 4 of 128
Time = 4 secs
Value = 0.20654

POLE ZERO



# of Poles = 13
# of Zeros = 13
Zero
Real = -0.9205
Imag.= 0.39073
Radius = 1
Angle = 157

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

Filter = valterfilts/kuc255_3
Method = valtermeths/cartesoide

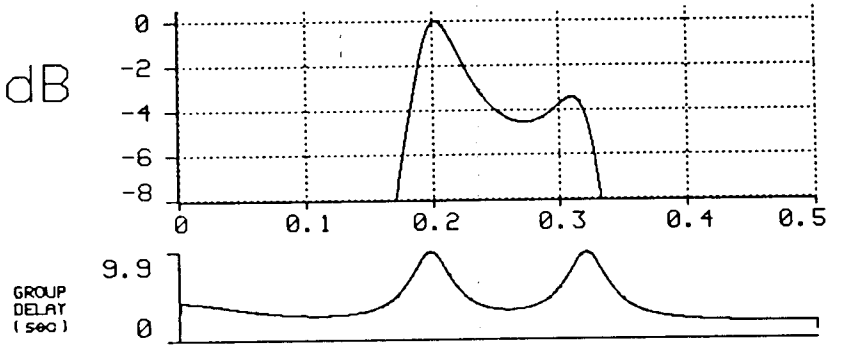
COMMENT

Exemplo: 3
FASE: 3
Zeros em 1 [157] e em 1 [28]

PARAMETERS

```
sample_freq = 1            {Frequencia de Amostr
ganho = 0.028000771            {Ganho do filtro}
num_seccoes = 7              {Limite maximo de 10}
seccao1 = 2          {Tipo 1:Segunda Ordem 2:P
zero1_real = 0          {Componente real do ze
zero1_imag = 0          {Componente imag do ze
polo1_real = -0.036635169            {Componente
polo1_imag = 0.6990406743            {Componente
seccao2 = 4          {Tipo para a seccao 2}
```

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -3.96767
Delay = 3.40876

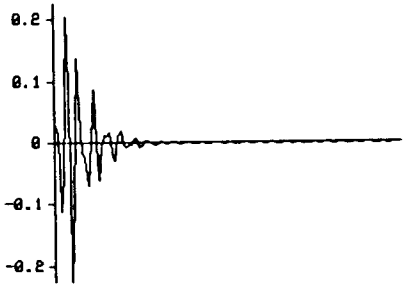
SOS FILTER COEFFICIENTS

Num. of sections = 7 Gain = 0.0280008

Sections

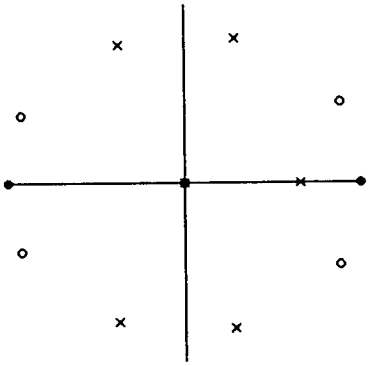
```
Section 0: first order
b[0][0] = 1
b[0][1] = -0
a[0][0] = 1
a[0][1] = -0.6624055053

Section 1: all zero
b[1][0] = 1
b[1][1] = -2
b[1][2] = 1
a[1][0] = 1

Section 2: all zero
b[2][0] = 1
b[2][1] = 2
b[2][2] = 1
```

IMPULSE RESPONSE



Point = 4 of 128
Time = 4 secs
Value = 0.20654

POLE ZERO



# of Poles = 13
# of Zeros = 13
Zero
Real = -0.9205
Imag.= -0.39073
Radius = 1
Angle = -157

ANEXO E: "DOCTOR FILTER"

EDIT MOVE
select component (DEPRESS)
select location (RELEASE)

Instance I BDE-2.8.2 Thu Jun 3 10:36:54 1993 RESET

Enter library/function name (l/f[.detail]).
Enter library/function name (l/f[.symbol]).
WARNING:
Plotting region from color display on a non-color device [05

SCHEMATIC: ezmenu/tools.menu(0)A
CONTEXT: same
VIEWPORT
VIEWPORT-SCALE: 0.721697
INSTANCE-SIZE=1
GRID=OFF MULTIPLE=0

CUSTOM BLK
- MAKEFILES
- COMPILE
- COMPILE/ADD
- COPY
- REMOVE

INSTANCE
- SHOW NUMBER
- HIGHLIGHT
- UPDATE
- UPDATE ALL

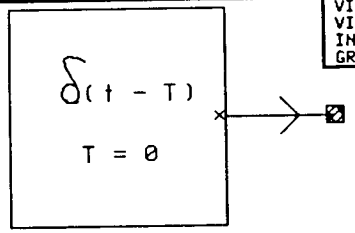
MISC
- IDENTIFY
- PLOT

SET OPS
- REGION
- ADD TO
- REMOVE
- DISBAND

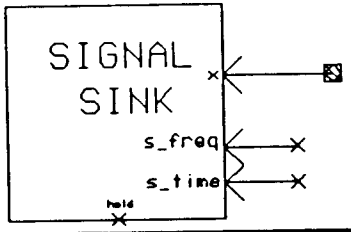
EXPERT
- ON
- OFF

MENUS
- DESIGN
- FILES
- SIMULATE
- CGS
- MPX
- DPC
- HDS
- VIEWPORT
- TOOLS
- HELP

valterlib/doctor_filter.detail(3)A



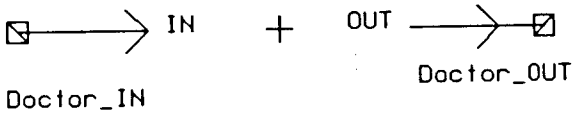
Library/File:
'valtersigs/doctor_out'



Doctor_IN

valterlib doctor_filter.symbol(3)A

DOCTOR
FILTER



ADD	BOX	TEXTPAD
ANALYSIS	CIRCLE	VIEWPORT
DISPLAY	CONNECTOR	WIRE
EDIT	DOT	
FILES	INSTANCE	
SET	LINE	
SYSTEM	PARAMETER	
	PORT	
	SIGNAL	

ADD VIEWPORT

- ADAPTIVE
- CHANNELS
- CONVERSION
- DECISION
- DEMODULATORS
- ENCODER/DECODER
- EQUALIZERS
- ESTIMATORS
- FILTERS
- INSTRUMENTS
- LOGICAL
- MATH
- MEMORY/DELAY
- MISCELLANEOUS
- MODULATORS
- NON-LINEAR PROC
- SIGNAL GEN/STORE
- TRANSFORMS
- VECTOR

OPT PRODUCTS
- CGS
- DPC
- HDS
- RADAR

- USER BLOCK
- USER PALETTE
- DEMO SYSTEMS

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

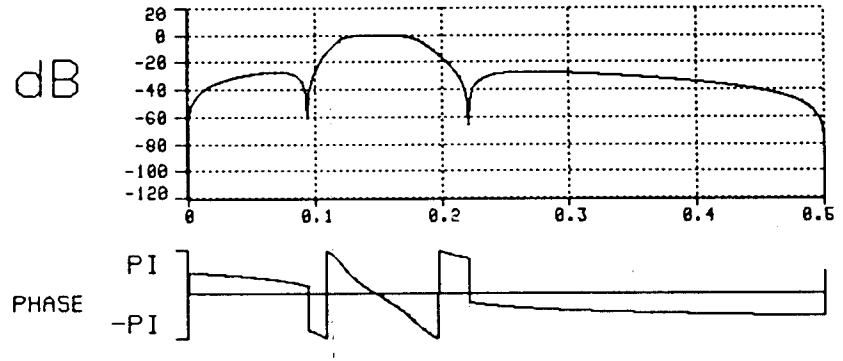
Filter = valterfilts/filtro1
Method = Chebyshev_2/BandPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
center_freq = 0.15 {center frequency}
passband_bw = 0.05 {passband bandwidth}
stopband_bw = 0.1 {stopband bandwidth}
passband_atten = 1 {passband attenuation in
filter_order = 6 {desired filter order}

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -27.2309
Phase = -0.885919

SOS FILTER COEFFICIENTS

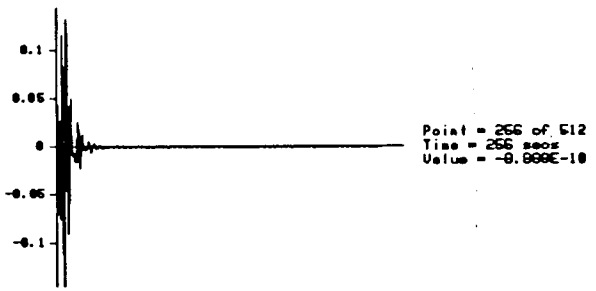
Num. of sections = 3 Gain = 0.0406313

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = -1.659375444
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.341503608
a[0][2] = 0.8620983808

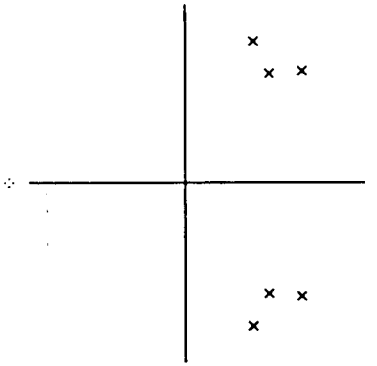
Section 1: second order
b[1][0] = 1
b[1][1] = -0.3638164399
b[1][2] = 1
a[1][0] = 1
a[1][1] = -0.7884147932
a[1][2] = 0.8243252314

IMPULSE RESPONSE



Point = 256 of 512
Time = 256 secs
Value = -0.000E-10

POLE ZERO



# of Poles = 6
# of Zeros = 6
Zero
Real = -1
Imag.= 0
Radius = 1
Angle = 180

FDS 2.8.2 I Filter Lib=valterfilts Method Lib=valtermeths Signal Lib=(null)

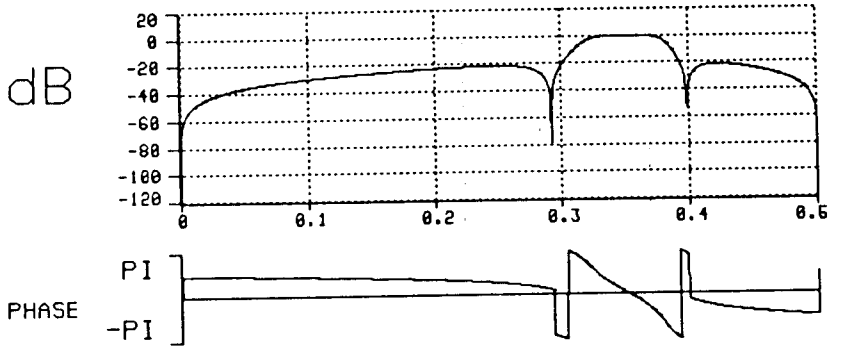
Filter = valterfilts/filtro2
Method = Chebyshev_2/BandPass

COMMENT

PARAMETERS

sample_freq = 1 {sample frequency}
center_freq = 0.35 {center frequency}
passband_bw = 0.05 {passband bandwidth}
stopband_bw = 0.1 {stopband bandwidth}
passband_atten = 1 {passband attenuatio
filter_order = 6 {desired filter order}

FREQUENCY RESPONSE



Point = 500 of 1000
Freq. = 0.25025
dB = -20.9018
Phase = 0.876969

SOS FILTER COEFFICIENTS

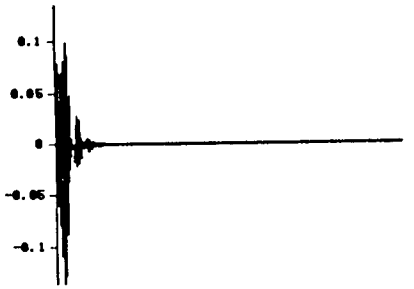
Num. of sections = 3 Gain = 0.0610683

Sections

Section 0: second order
b[0][0] = 1
b[0][1] = 0.5193451725
b[0][2] = 1
a[0][0] = 1
a[0][1] = 0.7874484518
a[0][2] = 0.8354022887

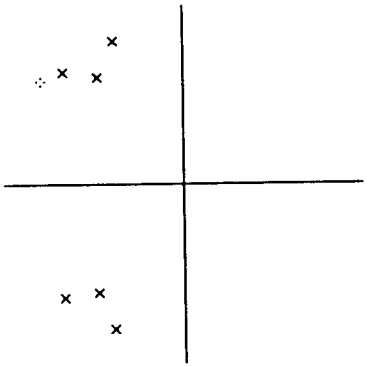
Section 1: second order
b[1][0] = 1
b[1][1] = 1.604812018
b[1][2] = 1
a[1][0] = 1
a[1][1] = 1.351728544
a[1][2] = 0.8714962319

IMPULSE RESPONSE



Point = 256 of 512
Time = 256 secs
Value = -3.3520E-09

POLE ZERO



# of Poles = 6
# of Zeros = 6
Zero
Real = -0.80241
Imag.= 0.59678
Radius = 1
Angle = 143.36

EDIT MOVE
select component (DEPRESS)
select location (RELEASE)

Instance I BDE-2.8.2 Wed Jun 2 21:44:03 1993 RESET

Welcome to SPW Ease-of-Use Menu System

WARNING:
Plotting region from color display on a non-color device [050

CUSTOM BLK
- MAKEFILES
- COMPILE
- COMPILE/ADD
- COPY
- REMOVE

INSTANCE
- SHOW NUMBER
- HIGHLIGHT
- UPDATE
- UPDATE ALL

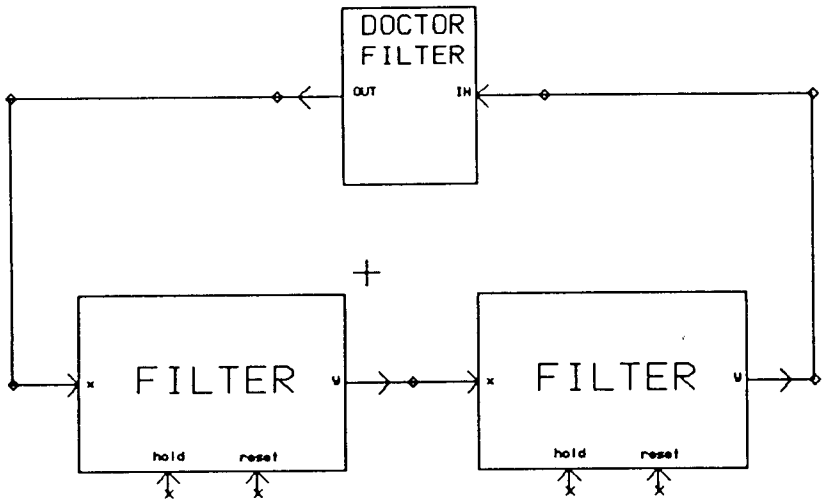
MISC
- IDENTIFY
- PLOT

SET OPS
- REGION
- ADD TO
- REMOVE
- DISBAND

EXPERT
- ON
- OFF

MENUS
- DESIGN
- FILES
- SIMULATE
- CGS
- MPX
- DPC
- HDS
- VIEWPORT
- TOOLS
- HELP

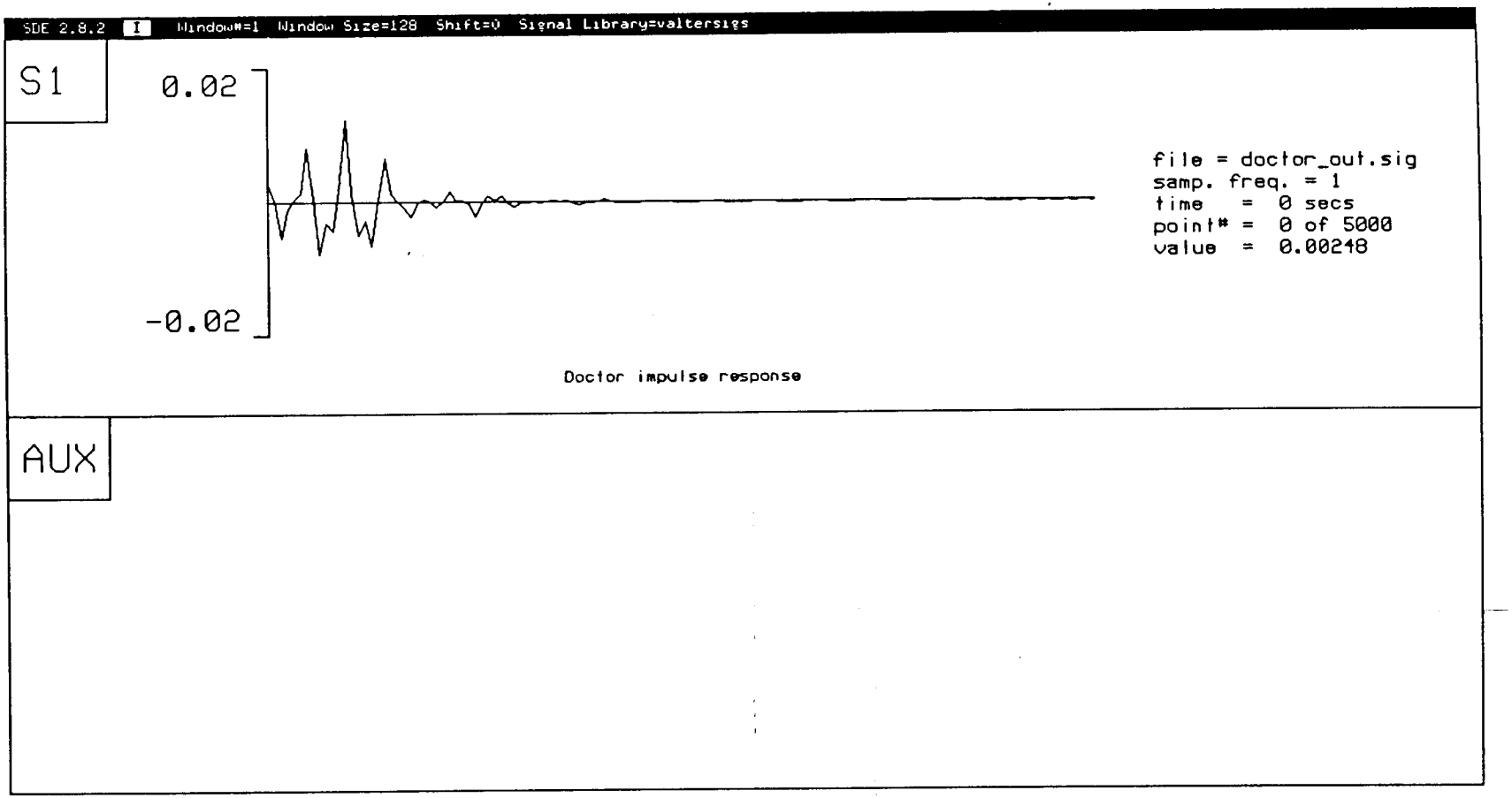
valterlib/filts.system(1)A

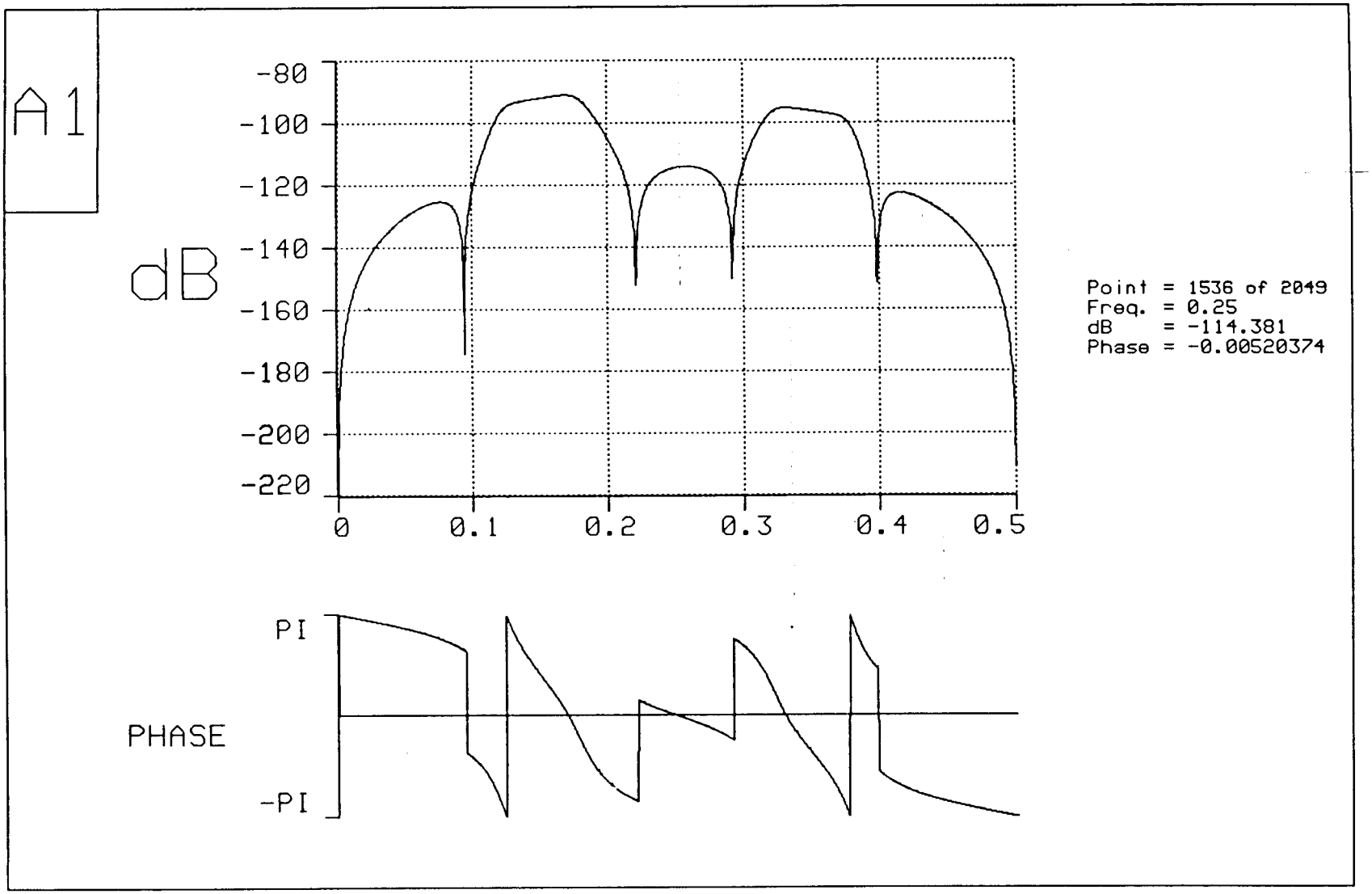


ADD ANALYSIS **DISPLAY** EDIT FILES SET SYSTEM	FRAME **OUTPUT** PAN PUSH VIEW ZOOM	ATTRIBUTES DIRECTORY HIERARCHY **PICTURE** tooldeck=	**model=** REGION	**device=**	[ALLDOTS] [zoom=] **[box=]** [frame=] **[mirror=]** [NICE] [pages=] [plotfile=] [rotate=] [size=]

DISPLAY OUTPUT PICTURE model=valterlib/filts.system device=PLOTTER box=1,0.1,0.1 mirror=none size=1W270,H180mm

- CONVOLUTION
- FILTER
- FREQ DOMAIN
- FREQ DOMAIN VECT
- FREQ FILT GAUSS
- FREQ FILT GR DLY
- FREQ FILT LAPP
- FREQ FILT RC
- HILBERT
- IIR BESSEL
- IIR BUTTER
- IIR CHEBY TYPE 1
- IIR CHEBY TYPE 2
- IIR ELLIPTIC
- LEAD LAG
- LEAD LAG CMPLX
- POLE-ZERO
- POLE-ZERO CMPLX
- RAISED COS
- SDE BANDPASS
- SDE HIGHPASS
- SDE LOWPASS
- SDE NOTCH
- TIME DOMAIN
- TIME FILT GAUSS
- TIME FILT GR DLY
- TIME FILT LAPP
- TIME FILT RC
- VFREQ FIL GAUSS
- VFREQ FIL GR DLY
- VFREQ FIL LAPP
- VFREQ FIL RC


SDE 2.8.2 I Window#=1 Window Size=128 Shift=0 Signal Library=valtersigs
S1
0.02
-0.02
file = doctor_out.sig
samp. freq. = 1
time = 0 secs
point# = 0 of 5000
value = 0.00248
Doctor impulse response
AUX


A1
dB
-80
-100
-120
-140
-160
-180
-200
-220
0
0.1
0.2
0.3
0.4
0.5
PHASE
PI
-PI
Point = 1536 of 2049
Freq. = 0.25
dB = -114.381
Phase = -0.00520374

edit delete
select component (DEPRESS)
(RELEASE)

Instance I BDE-2.8.2 Thu Jun 3 10:36:54 1993 RESET

SPB execute reports 0 errors or warnings
Evaluate Complete (job = 255).
WARNING:
Plotting region from color display on a non-color device [05

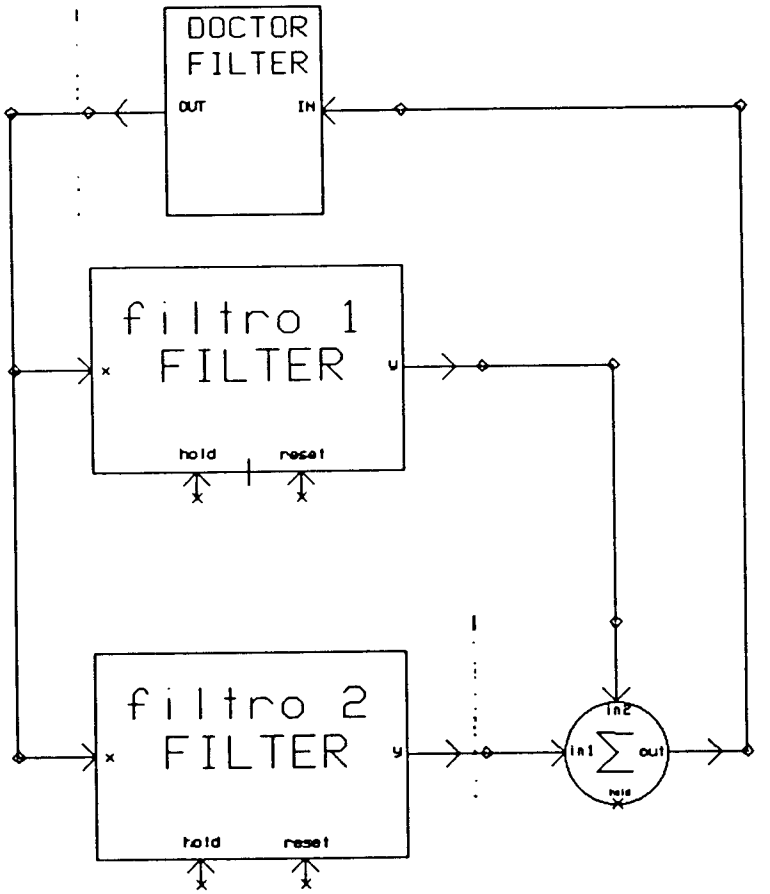
valterlib/filts.system(4)A

ADD
- BLOCK
- CONNECTOR
- PARAMETER
- PORT
- VIEWPORT
- WIRE

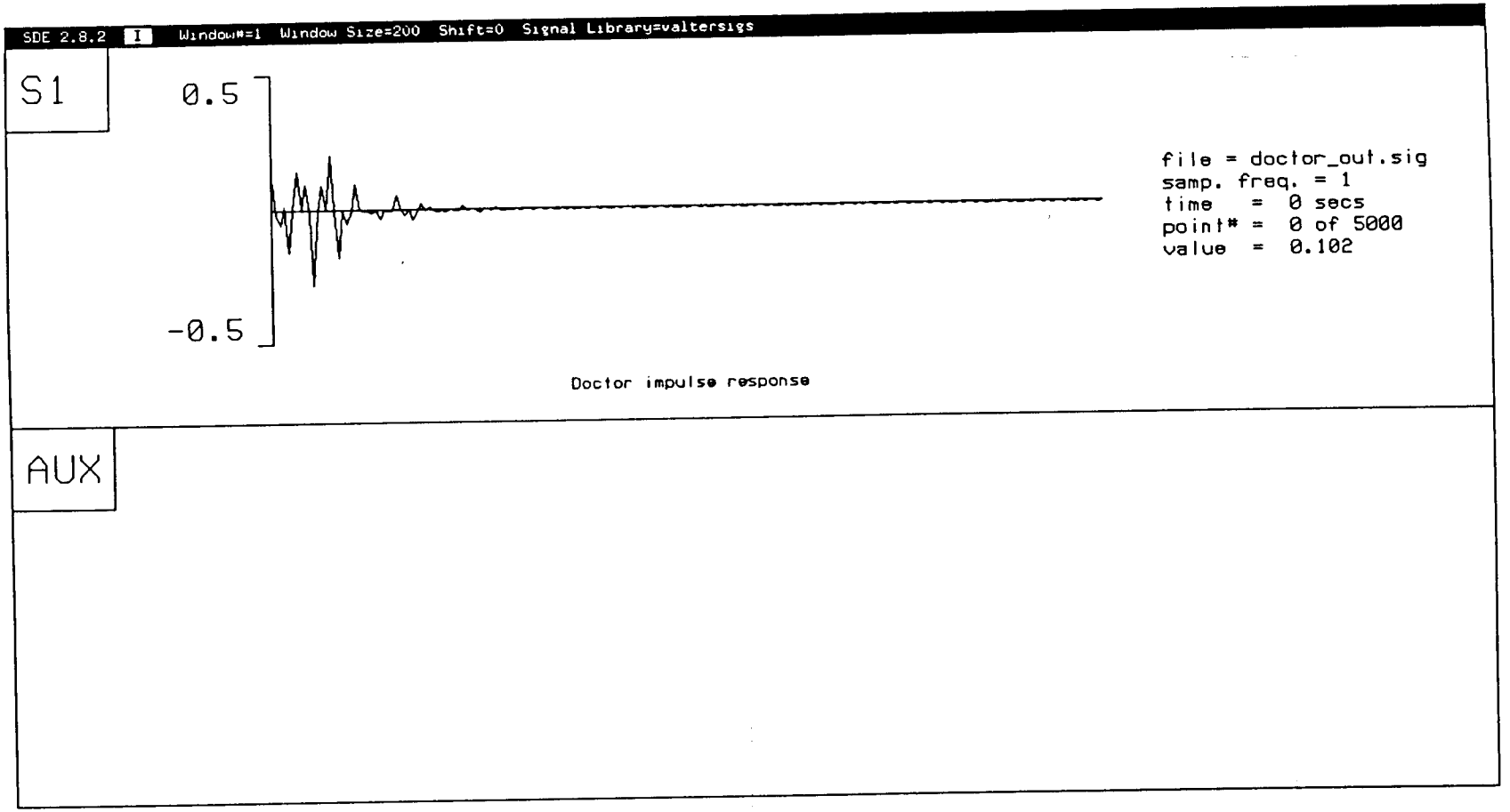
EDIT
- COPY
- DELETE
- EXPORT PARM
- MIRROR
- MOVE
- ROTATE
- SIZE DOWN
- SIZE UP
- STRETCH

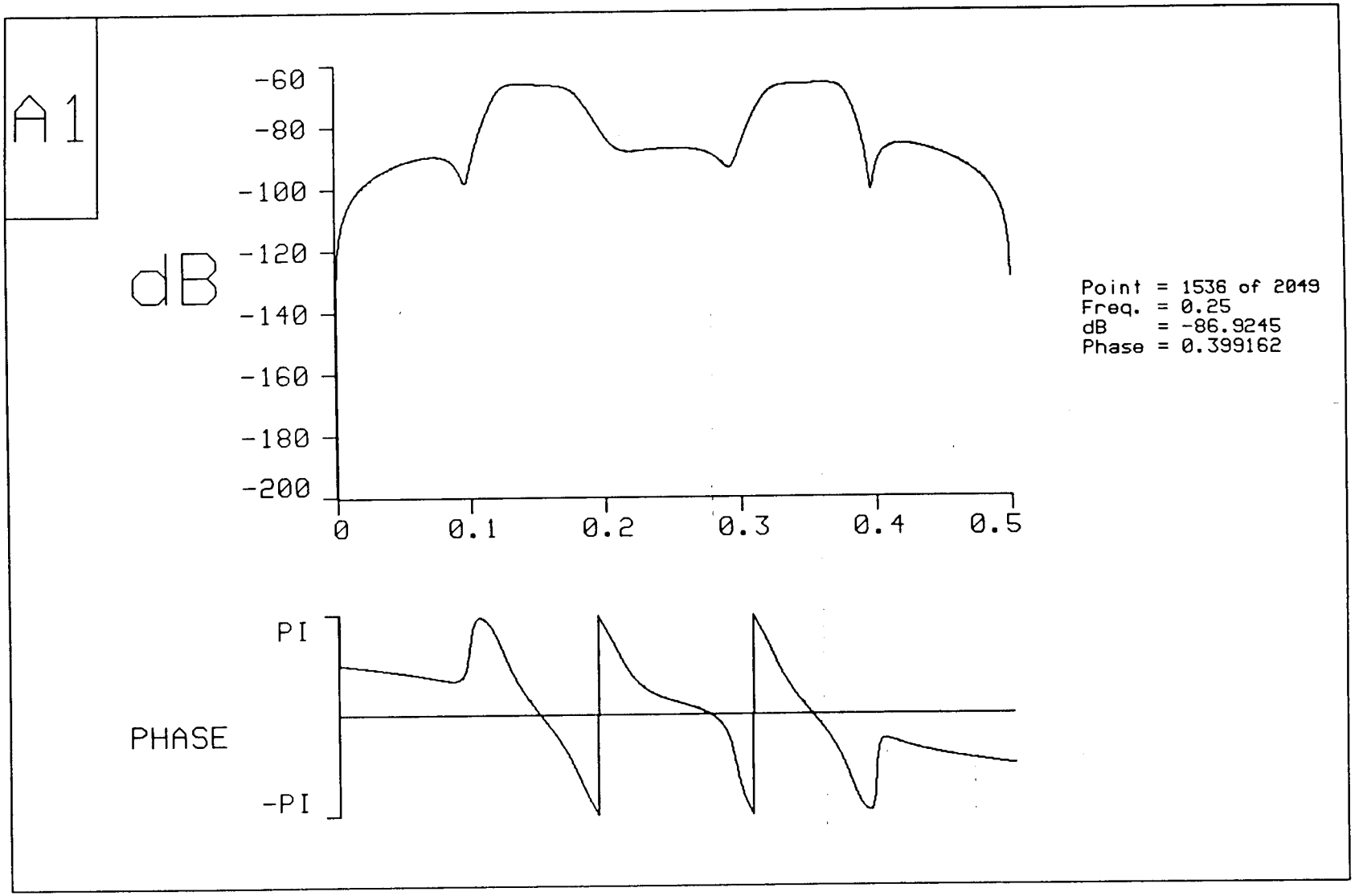
DISPLAY
- PAN
- POP
- PUSH NEXT
- PUSH BACK
- PUSH TOP
- ZOOM FIT
- ZOOM REGION
- ZOOM=1
- ZOOM IN
- ZOOM OUT

MENUS
- DESIGN
- FILES
- SIMULATE
- CGS
- MPX
- OPC
- HDS
- VIEWPORT
- TOOLS
- HELP



ADDER 2-INPUT
- COMPLEX
- SCALAR
- VECTOR
- VECTOR CMPLX


SDE 2.8.2 I Window#=1 Window Size=200 Shift=0 Signal Library=valtersigs
S1
0.5
-0.5
file = doctor_out.sig
samp. freq. = 1
time = 0 secs
point# = 0 of 5000
value = 0.102
Doctor impulse response
AUX


A1
dB
-60
-80
-100
-120
-140
-160
-180
-200
0
0.1
0.2
0.3
0.4
0.5
Point = 1536 of 2049
Freq. = 0.25
dB = -86.9245
Phase = 0.399162
PHASE
PI
-PI

File Edit View Design Tools Convert Customize Help

Filter =
Method = Hierarchical/Collapse

Parameters

Sampling Frequency	1
Expression	F0*F1
F0	valterfilts/filtro1.ascfilt
F1	valterfilts/filtro2.ascfilt
F2	mfds_site_test/bp_1.ascfilt
F3	mfds_site_test/bp_2.ascfilt
F4	"
F5	"
F6	"
F7	"
F8	"
F9	"

Apply

Sos Coefficients

Number of Sections = 6
Gain = 0.00248128

Section 0: second order

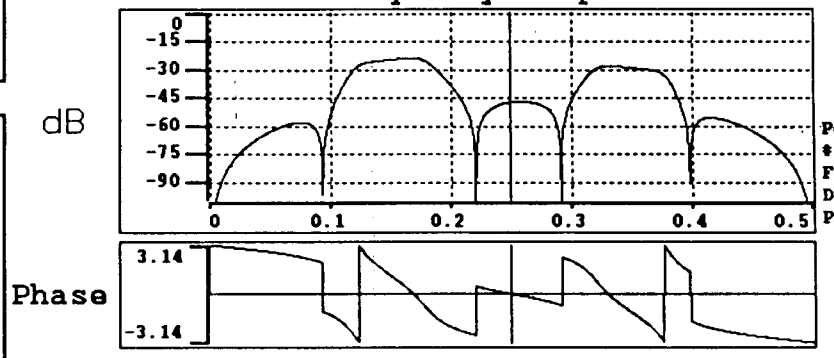
b[0][0] = 1
b[0][1] = -1.6584
b[0][2] = 1
a[0][0] = 1
a[0][1] = -1.3415
a[0][2] = 0.8621

Section 1: second order

b[1][0] = 1
b[1][1] = -0.36382
b[1][2] = 1
a[1][0] = 1
a[1][1] = -0.78841
a[1][2] = 0.82433

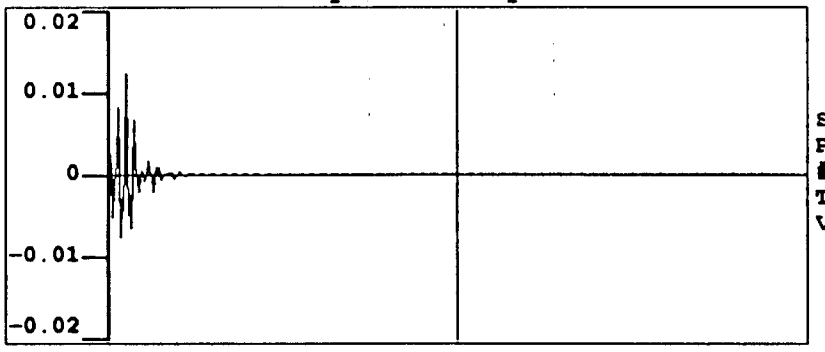
Apply

Frequency Response



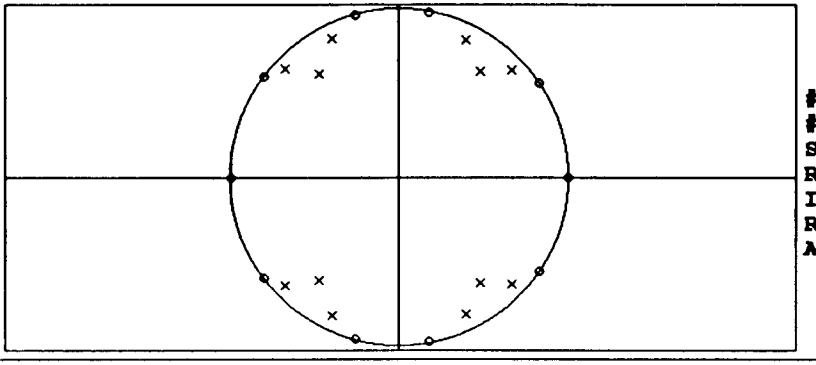
Point = 511
# Pts = 1024
Freq = 0.249512
Db = -48.176
Phase = -0.0015458

Impulse Response



Samp. Freq. = 1
Point = 256
# Pts = 512
Time = 256 secs
Value = -2.22e-11

Pole Zero



# Poles = 12
# Zeros = 12
Symbol = ZERO
Real = -1
Imag = 0
Radius = 1
Angle = 180

File Edit View Design Tools Convert Customize

Help

Filter =

Method = Hierarchical/Collapse

Parameters

Sampling Frequency	1
Expression	F0+F1
F0	valterfilts/filtro1.ascfilt
F1	valterfilts/filtro2.ascfilt
F2	mids_sks_test/bp_1.ascfilt
F3	mids_sks_test/bp_2.ascfilt
F4	--
F5	--
F6	--
F7	--
F8	--
F9	--

Apply

Sos Coefficients

Number of Sections = 6

Gain = 0.1017

Section 0: second order

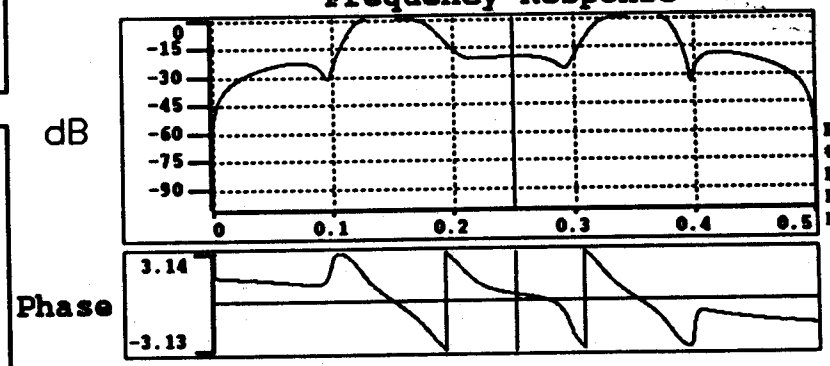
b[0][0] = 1
b[0][1] = -0.12664
b[0][2] = 0.36276
a[0][0] = 1
a[0][1] = -0.98867
a[0][2] = 0.62939

Section 1: second order

b[1][0] = 1
b[1][1] = 1.5864
b[1][2] = 0.98345
a[1][0] = 1
a[1][1] = 0.9557
a[1][2] = 0.60591

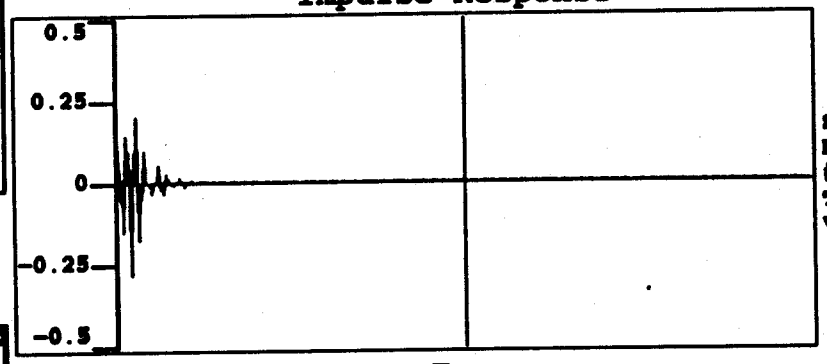
Apply

Frequency Response



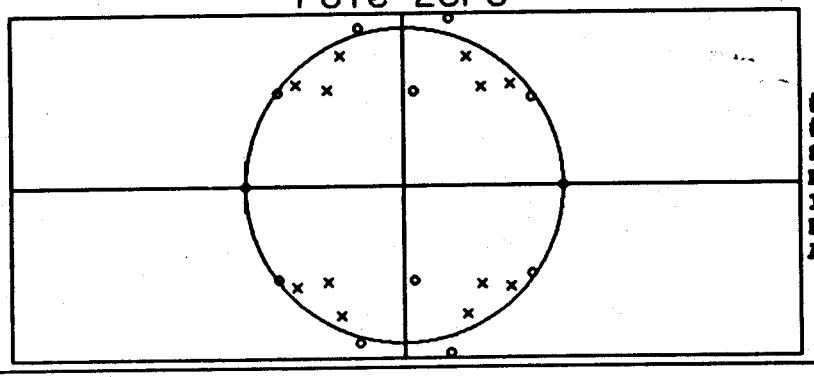
Point = 511
# Pts = 1024
Freq = 0.249512
Db = -20.700
Phase = 0.40218

Impulse Response



Samp. Freq. = 1
Point = 256
# Pts = 512
Time = 256 secs
Value = -4.24e-09

Pole Zero



# Poles = 12
# Zeros = 12
Symbol = ZERO
Real = -1
Imag = 0
Radius = 1
Angle = 180

BIBLIOTECA F.E. PORTO

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
BIBLIOTECA
0000101542