Sobre o Projeto de Filtros Digitais de Segunda Ordem Imunes a Ciclos Limite

Marcelo Oliveira Camponêz

Dissertação de Mestrado em Engenharia Elétrica (Automação)

Mestrado em Engenharia Elétrica (Automação)
Universidade Federal do Espírito Santo
Vitória, Dezembro de 1998

Sobre o Projeto de Filtros Digitais de Segunda Ordem Imunes a Ciclos Limite

Marcelo Oliveira Camponêz

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica — Automação.

Aprovada em 30/12/1998 por:

_____

Prof. Dr. Mário Sarcinelli Filho, UFES - Orientador

_____

Prof. Dr. Luiz de Calazans Calmon, UFES

_____

Prof. Dr. Francisco das Chagas Mota, UFRN

Universidade Federal do Espírito Santo
Vitória, Dezembro de 1998

Camponez, Marcelo Oliveira, 1973

Sobre o Projeto de Filtros Digitais de Segunda Ordem Imunes a Ciclos Limite. [Vitória] 1998

XII, 116p., 29.7 cm (UFES, M. Sc., Engenharia Elétrica, 1998)

Dissertação de Mestrado, Universidade Federal do Espírito Santo

Dedicatória

Dedico este trabalho
a minha esposa
Leticia

Agradecimentos

Neste momento tão festivo em que chego ao final desta pós graduação, não posso me furtar o direito de agradecer a algumas pessoas, que contribuíram de alguma maneira para a realização deste trabalho.

Em primeiro lugar eu agradeço a Deus, o autor e consumador de todas as coisas, pela experiência inarrável que é viver na dependência de um Deus vivo.

Em segundo lugar agradeço a minha família, meus pais Marisy e José Natalino, pela oportunidade e apoio que me deram durante a minha vida estudantil. Ao meu sogro Helmo e minha sogra Reneida, pela acolhida carinhosa e o conforto tão necessário que me deram quando eu precisei, especialmente aquela comida gostosa do final de semana. Aos meu tio Tony e minha tia Marta, se não fossem vocês eu não teria entrado nem na faculdade de engenharia, quem diria terminar um Mestrado, vocês são simplesmente demais. Aos meus avós, avôs, tios, tias, gente eu amo muito vocês.

Em terceiro lugar a minha esposa Leticia, você que acompanhou todos os meus passos de perto, viveu comigo as mesmas angustias, muito obrigado pela paciência e compreensão, especialmente no período final desta dissertação.

Ao professor e orientador Mário Sarcinelli Filho, por todo apoio, paciência e dedicação, que teve por este trabalho.

À nossa secretária Marlene Patrício da Silva, pela simpatia e mão amiga que sempre estendeu, sempre disposta a ajudar em tudo.

Ao pessoal da Banda Expressão, vocês me deram energia e renovaram as minhas forças para que eu pudesse trilhar esse caminho.

Ao meus amigos de trabalho da Telest, que também me motivaram a continuar estudando, me deram apoio e me fizeram acreditar na concretização deste sonho.

A todos os meus amigos de Mestrado, pelos momentos que estivemos estudando juntos.

Ao CNPq, pelo suporte financeiro concedido.

E a todos vocês que acreditaram no meu trabalho, vocês que me deram força, meu muito obrigado.

BRIGADUUUUU !!!

Marcelo Oliveira Camponêz

SUMÁRIO

Lista de Figuras

Resumo

Um conjunto de três seções de segunda ordem no espaço de estados livres de ciclos limite a entrada constante, foi proposto em 1986, para serem usadas na implementação de filtros digitais na forma paralela ou cascata. Estas seções foram chamadas respectivamente de estrutura do tipo I, estrutura do tipo II (que é inteiramente igual ao primeiro tipo) e estrutura do tipo III.

Estas estruturas são diferentes de outros blocos de segunda ordem livres de ciclo limite a entrada constante no sentido que elas são computacionalmente menos complexas. Conforme mostrado no artigo que a introduziu, a estrutura do tipo I é uma estrutura de baixo ruído, quando comparada com a rede de mínimo ruído sintetizada sob escalamento L_∞. Infelizmente, devido ao fato da rede tipo III ser ligeiramente mais complexa que a rede do tipo I (ela exige mais três somadores de duas entradas para ser computada corretamente), sua síntese e seu desempenho a nível de ruído sequer foram abordadas no artigo inicial. Porém, foi demostrado que a estrutura do tipo III apresenta variância relativa do ruído inferior à da estrutura do tipo I, para filtros passa-baixas escalados em L_2.

Então, neste trabalho, a estrutura do tipo III é revisada, e novas estratégias são propostas para sintetizá-la. Quando adequadamente projetada, mostra-se que sua variância relativa do ruído é mínima. Tanto o escalamento L_2 quanto o escalamento L_∞ são considerados. Exemplos são mostrados , os quais ilustram os resultados apresentados.

Finalmente a fim de ressaltar o bom desempenho a nível de ruído da estrutura tipo III projetada segundo as estratégias aqui discutidas, ela é comparada com o desempenho de algumas outras estruturas de filtros conhecidas da literatura. A conclusão é que a nova estrutura do tipo III aqui proposta é uma boa candidata a bloco construtivo para a implementação de filtros de banda estreita de ordem elevada.

Abstract

A set of three constant-input limit cycle-free state-space second-order digital filter sections were proposed in a paper of 1986, to be used as building blocks for cascade or parallel higher order filter design. They were called structure type I, structure type II (which is entirely identical to the first one) and structure type III, by the authors who proposed them.

Those structures are different from other constant-input limit cycle-free second-order building blocks in the sense that they are computationally less complex. As showed in the paper that introduced it, the structure type I is also a low roundoff noise one, when compared to the minimum noise structure designed under L_∞ scaling. Unfortunately, because the structure type III is slightly more complex than the structure type I (its demands three two-input adders more to be computed), its synthesis procedure and noise performance were not addressed in the initial paper. However, it was demonstrated later, that the structure type III presents relative output roundoff noise variances lower than the structure type I for the case of L_2 scaled lowpass filters.

Then, in this work, the structure type III is revised, and new strategies for designing it are proposed. Under proper design, it is showed that the relative output roundoff noise variance is minimal. Both L_2 and L_∞ scaling are here discussed. Examples are shown to support the results presented.

Finally, in order to stress the good noise performance of the structure type III designed according to the strategies here discussed, it is compared to the noise performance of some other filter structures known in the literature. The conclusion is that the new structure type III here proposed is a good candidate for being the building block to implement high order narrowband filters.

CAPÍTULO I

Introdução

Uma vez conhecida a função de transferência H(z) que o descreve, dá-se início à fase de realização de um filtro digital. Dentre as muitas propostas de realização contempladas na literatura, merece destaque a implementação de filtros digitais no espaço de estados.

A representação no espaço de estados simplifica a análise de sistemas analógicos. Ela é uma ferramenta importante na caracterização da resposta dos mais diversos sistemas, assim como permite caracterizar uma série de propriedades fundamentais dos mesmos [7]. Similarmente, este conceito foi também desenvolvido para filtros digitais. Pode-se encarar esta representação como uma forma mais genérica de representar um filtro digital qualquer, sendo que a própria realização direta é um caso particular desta representação [7].

Dentro do universo de filtros digitais, este tipo de representação ganhou notoriedade quando Mullis e Roberts [1], através de transformações de similaridade, sintetizaram o filtro de mínimo ruído. Pouco tempo depois, eles mesmos provaram uma importante propriedade desta estrutura, que é a invariância do ruído a transformações de freqüência [2]. Até hoje, a classe de estruturas digitais no espaço de estados tem merecido destaque, principalmente por seu baixo ruído, especialmente em casos de filtros de banda estreita [1], [3], [5], [6] e [22].

A implementação ótima de filtros de ordem elevada, porém, é demasiadamente complexa, do ponto de vista computacional [1]. Uma boa solução de compromisso é, então, implementar estruturas de segunda ordem ótimas, e associá-las nas formas cascata ou paralela [1]. A técnica consiste em otimizar cada bloco, no sentido de minimizar o ruído por ele gerado. A síntese ótima de blocos de segunda ordem, por sua vez, pode ser feita como proposto em [3]. Assim é possível um equilíbrio entre alta complexidade computacional e baixo ruído, que é o que se deseja obter quando se realiza um filtro digital (uma vez que não é possível evitar a geração de ruído na saída do filtro, pois o processamento numérico do sinal de entrada necessariamente gera tal ruído [9]).

Desta forma, muitos trabalhos têm proposto estruturas de segunda ordem no espaço de estados com bom desempenho, a nível do ruído na saída, além de outras características de interesse, como a imunidade a ciclos limite [3]-[6]. Sintetizando tais resultados o presente capítulo caracteriza o ruído na saída de filtros digitais e aborda as condições para garantir a ausência de ciclos limite, sempre considerando filtros digitais de segunda ordem no espaço de

estados. O objetivo é indicar o contexto em que os resultados discutidos nos próximos capítulos se inserem.

1.1 - Representação de Filtros Digitais no Espaço de Estados

A descrição de filtros digitais de segunda ordem no espaço de estados é caracterizada por

$$\mathbf{X}[n+1] = \mathbf{A}\mathbf{X}[n] + \mathbf{b}\mathrm{U}[n] \tag{1.1}$$

$$\mathrm{Y}[n] = \mathbf{c}\mathbf{X}[n] + d\mathrm{U}[n] \tag{1.2}$$

onde

$\mathbf{X}[n] = [x_1[n]\ x_2[n]]^T$ é o vetor das variáveis de estado,

U[n] é a seqüência de escalares que representa as amostras do sinal de entrada,

Y[n] é a seqüência de escalares que representa as amostras do sinal de saída,

A (matriz 2 x 2),

b (matriz 2 x 1),

c (matriz 1 x 2) e

d (escalar) são coeficientes multiplicadores, os quais caracterizam uma realização no espaço de estados. Estes parâmetros estão caracterizados na Figura 1.1, a seguir.

Aplicando a transformada *z* às equações (1.1) e (1.2) e resolvendo, pode-se mostrar que

$$\mathrm{H}(z) = \mathbf{c}^T(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + \mathrm{d} \tag{1.3}$$

e resolvendo tal equação chega-se a

$$\mathrm{H}(z) = \frac{dz^2 + [b_1c_1 + b_2c_2 - d(a_{11} + a_{22})]z + b_1(c_2a_{21} - c_1a_{22}) + b_2(c_1a_{12} - c_2a_{11}) + d(a_{11}a_{22} - a_{21}a_{12})}{z^2 - (a_{11} + a_{22})z + (a_{11}a_{22} - a_{12}a_{21})} \tag{1.4}$$

Mas a expressão genérica para a função de transferência de um filtro digital de segunda ordem é dada por

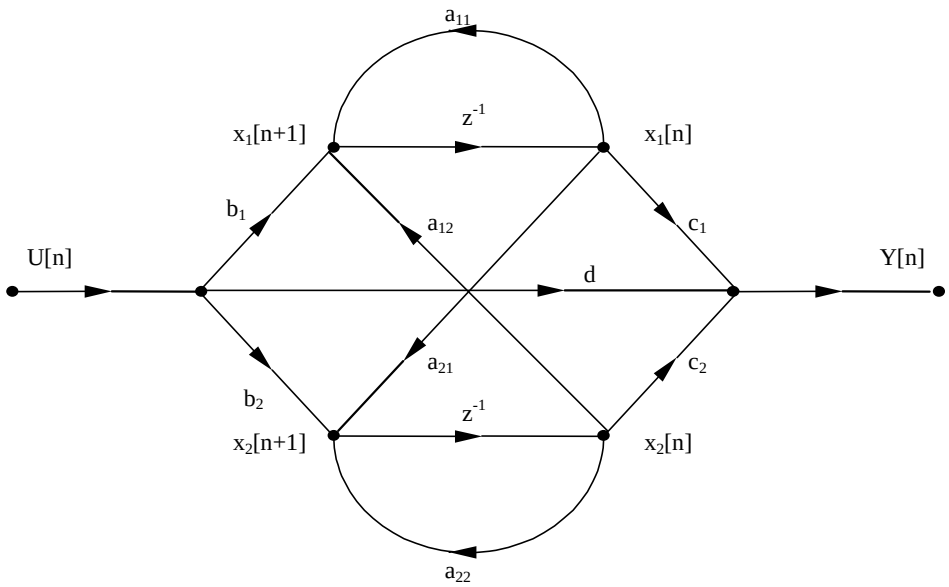

Figura 1.1 - Estrutura no espaço de estados de segunda ordem correspondendo ao filtro ideal (linear).

$$H(z) = \frac{\gamma_0 z^2 + \gamma_1 z + \gamma_2}{z^2 + \alpha_1 z + \alpha_2} \tag{1.5}$$

e comparando-se tal resultado com (1.4) obtém-se

$$\gamma_0 = d \tag{1.6}$$

$$\gamma_1 = b_1c_1 + b_2c_2 - d(a_{11} + a_{22}) \tag{1.7}$$

$$\gamma_2 = b_1(c_2a_{21} - c_1a_{22}) + b_2(c_1a_{12} - c_2a_{11}) + d(a_{11}a_{22} - a_{21}a_{12}) \tag{1.8}$$

$$\alpha_1 = -(a_{11} + a_{22}) \tag{1.9}$$

$$\alpha_2 = a_{11}a_{22} - a_{21}a_{12} \tag{1.10}$$

Alternativamente, pode-se rescrever H(z) como

$$H(z) = d + H'(z) \tag{1.11}$$

e, desta forma tem-se que

$$H'(z) = \frac{\beta_1 z + \beta_2}{z^2 + \alpha_1 z + \alpha_2} \tag{1.12}$$

e daí se pode obter as seguintes relações entre as variáveis

$$\beta_1 = \gamma_1 - \gamma_0\alpha_1 = b_1c_1 + b_2c_2 \quad (1.13)$$

$$\beta_2 = \gamma_2 - \gamma_0\alpha_2 = b_1(c_2a_{21} - c_1a_{22}) + b_2(c_1a_{12} - c_2a_{11}) \quad (1.14)$$

1.2 - Ruído em Filtros Digitais

Na implementação de filtros digitais, tanto em *software* quanto em *hardware*, é preciso acomodar números em registradores, os quais possuem um número finito de bits para armazenar cada palavra. Em conseqüência disso, os valores a serem armazenadas precisam ser quantizados, ou seja, precisam ser alterados em seu valor (por serem alterados em sua representação) para caberem nos registradores usados. A quantização gera, então, três tipos de erro, a saber

1 - erro de quantização dos coeficientes;
2 - erro de quantização dos produtos; e
3 - erro de quantização do sinal de entrada.

Tais erros de quantização é que fazem com que o ruído na saída seja intrínseco aos filtros digitais. O que se busca, então, é a minimização do ruído na saída, ou seja a minimização dos erros de quantização propagados para a saída do filtro.

O nível do ruído na saída do filtro pode ser reduzido incrementando-se o tamanho da palavra, ou, alternativamente, pode-se partir de uma estrutura genérica e, através da variação dos parâmetros ou de sua topologia, chegar a uma nova estrutura, que apresente menos ruído na saída.

Em outras palavras não é possível realizar o filtro linear representado na Figura 1.1, mas apenas um versão ligeiramente diferente desta, como mostrado na Figura 1.2.

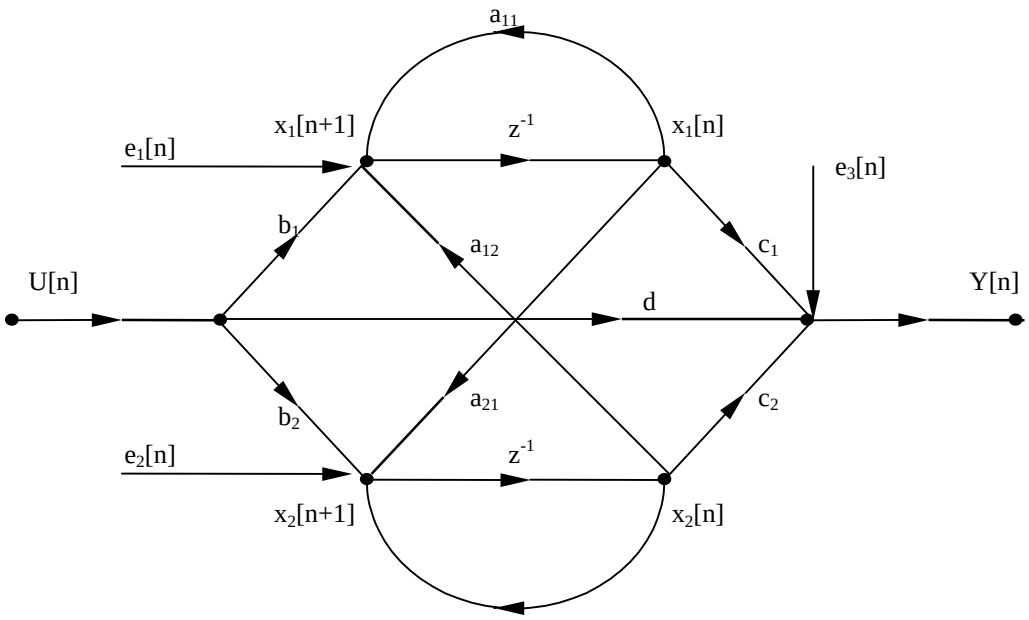

Figura 1.2 - Realização de segunda ordem no espaço de estados efetiva (filtro não linear).

Na Figura 1.2 os sinais $e_i[n]$, i = 1, 2 e 3, são fontes de ruído que são usadas para indicar, nos pontos em que elas são inseridas, a existência de quantização de produtos ou de soma de produtos [9]. O filtro realizado, então, é representado pela equação

$$\mathbf{X}[n+1] = \mathbf{A}\mathbf{X}[n] + \mathbf{b}U[n] + \mathbf{e}[n] \tag{1.15}$$

$$Y(n) = \mathbf{c}\mathbf{X}[n] + \mathbf{d}U[n] + e_3[n] \tag{1.16}$$

onde

$$\mathbf{e}^T[n] = [e_1[n]\ e_2[n]] \tag{1.17}$$

É importante frisar que o uso das fontes de erro é uma representação simplificada do erro gerado na quantização, e sua base consiste em considerar que as fontes de ruído não são correlacionadas uma com as demais, assim como amostras distintas de uma mesma fonte de ruído são não correlacionadas [3].

1.3 - Representação dos Sinais em Ponto Fixo em Complemento a Dois

Seja um número qualquer N. Considere que ele é representado por um bit de sinal (b_0) e um número binário (b_1, b_2 ,b_3, ..., b_n), este representando seu módulo. Então

$$N = b_0\, b_1\, b_2\, b_3 \ldots b_n \tag{1.18}$$

o valor de b_0 é dado por

$$b_0 = \begin{cases} 0 & \text{para } N \geq 0 \\ 1 & \text{para } N < 0 \end{cases} \tag{1.19}$$

O número de bits usados nesta representação é n + 1, onde n varia de forma a contemplar o máximo módulo de N que se quer representar. Considere-se, agora, que o valor máximo do módulo de N é 1. Assim, ao se ler o conjunto de n + 1 bits acima, é necessário interpretar que os últimos n bits correspondem a um número “fracionário”, ou seja, deve-se assumir a presença de um ponto binário (por analogia com o ponto decimal usado no sistema decimal). No caso, como $|N| \leq 1$, o ponto está posicionado entre os dois primeiros bits (b_0 e b_1)

A representação de tal número N em complemento a dois tem a forma geral dada por

$$N_2 = \begin{cases} N & \text{para } N \geq 0 \\ 2 - |N| & \text{para } N < 0 \end{cases} \tag{1.20}$$

Como N tem um ponto binário associado à sua representação, N_2 também o terá. Quando a posição de tal ponto permanece fixa na mesma posição, tem-se a representação em ponto fixo em complemento a dois.

1.4 - Representação dos Sinais em Ponto Flutuante

Em ponto flutuante um número é expresso por

$$N = M \times 2^e \tag{1.21}$$

onde ‘e’ é um inteiro e

$$\frac{1}{2} \leq M < 1 \tag{1.22}$$

M e ‘e’ são chamados mantissa e expoente, respectivamente. Nas operações de soma, basta igualar o expoente dos dois números, somar as mantissas e repetir o expoente. Em caso de multiplicação, multiplica-se as mantissas e soma-se os expoentes.

Como vantagem desse tipo de representação sobre aquela em ponto fixo, tem-se que a faixa dinâmica dos números representados é maior, levando a uma maior precisão no processo de quantização. Como desvantagens, ela aumenta o custo do *hardware* e diminui a velocidade de processamento. Na maioria das aplicações, a representação em ponto fixo é utilizada, escalando-se o filtro de forma a maximizar a faixa dinâmica. Entretanto, algumas implementações de filtros digitais em ponto flutuante têm aparecido na literatura mais recentemente [24]-[25].

Neste trabalho somente a representação em ponto fixo em complemento a dois é considerada. A seu favor, é necessário dizer que ela é utilizada em grande parte dos processadores digitais de sinais disponíveis no mercado [23]. Adicionalmente, com o número de bits relativamente pequeno, e considerando que o número representado tem módulo no máximo unitário, o erro de representação cometido não é grande [9].

1.5 - Quantização Numérica e Escalamento

Sempre que um número real x é quantizado, um erro ε é introduzido, o qual é caracterizado por

$$\varepsilon = x - Q[x] \tag{1.23}$$

onde Q[x] é o valor quantizado de x. A faixa de variação de ε vai depender do tipo de quantização e, é claro, da representação numérica usada.

A quantização por arredondamento, truncamento ou truncamento em magnitude pode ser usada para quantizar os valores dos coeficientes multiplicadores (que também têm que ser representados por uma palavra de tamanho finito) e os produtos de tais coeficientes pelos

sinais (no caso, o sinal de entrada ou as variáveis de estado). Cada método é caracterizado por um valor esperado e uma variância [8] do valor do erro.

Considerando que ε(n) é uma variável estocástica com distribuição semelhante à do ruído branco [9], tem-se que, para o arredondamento,

$$E[\varepsilon(n)] = 0 \tag{1.24}$$

$$V[\varepsilon(n)] = \sigma_e^2 = \frac{q^2}{12} = \frac{2^{-2b}}{12} \tag{1.25}$$

onde b é número de bits usados na quantização menos um. Para o truncamento, mostra-se que

$$E[\varepsilon(n)] = -\frac{q}{2} = \frac{-2^{-2b}}{12} \tag{1.26}$$

$$V[\varepsilon(n)] = \sigma_e^2 = \frac{q^2}{12} = \frac{2^{-2b}}{12} \tag{1.27}$$

enquanto que, para o truncamento em magnitude, obtém-se que

$$E[\varepsilon(n)] = 0 \tag{1.28}$$

$$V[\varepsilon(n)] = \sigma_e^2 = \frac{q^2}{12} = \frac{2^{-2b}}{3} \tag{1.29}$$

A Figura 1.3 mostra a relação entre a variável x e o valor quantizado Q[x], nos casos de arredondamento, truncamento e truncamento em magnitude.

Os erros causados pelo processo de quantização irão causar várias perturbações no sistema. No caso da quantização dos coeficientes do filtro, esses erros introduzirão perturbações nos zeros e pólos da função de transferência. Já no caso da quantização dos produtos, os erros podem ser modelados como fontes de erro do multiplicador para a saída do filtro, e correspondem a uma operação não linear que insere ruído na saída do filtro realizado, assim como pode causar oscilações na sua saída, mesmo com os pólos permanecendo no interior do círculo unitário [9]. Este efeito será discutido mais adiante. Por enquanto, considere-se apenas a questão do ruído inserido no filtro.

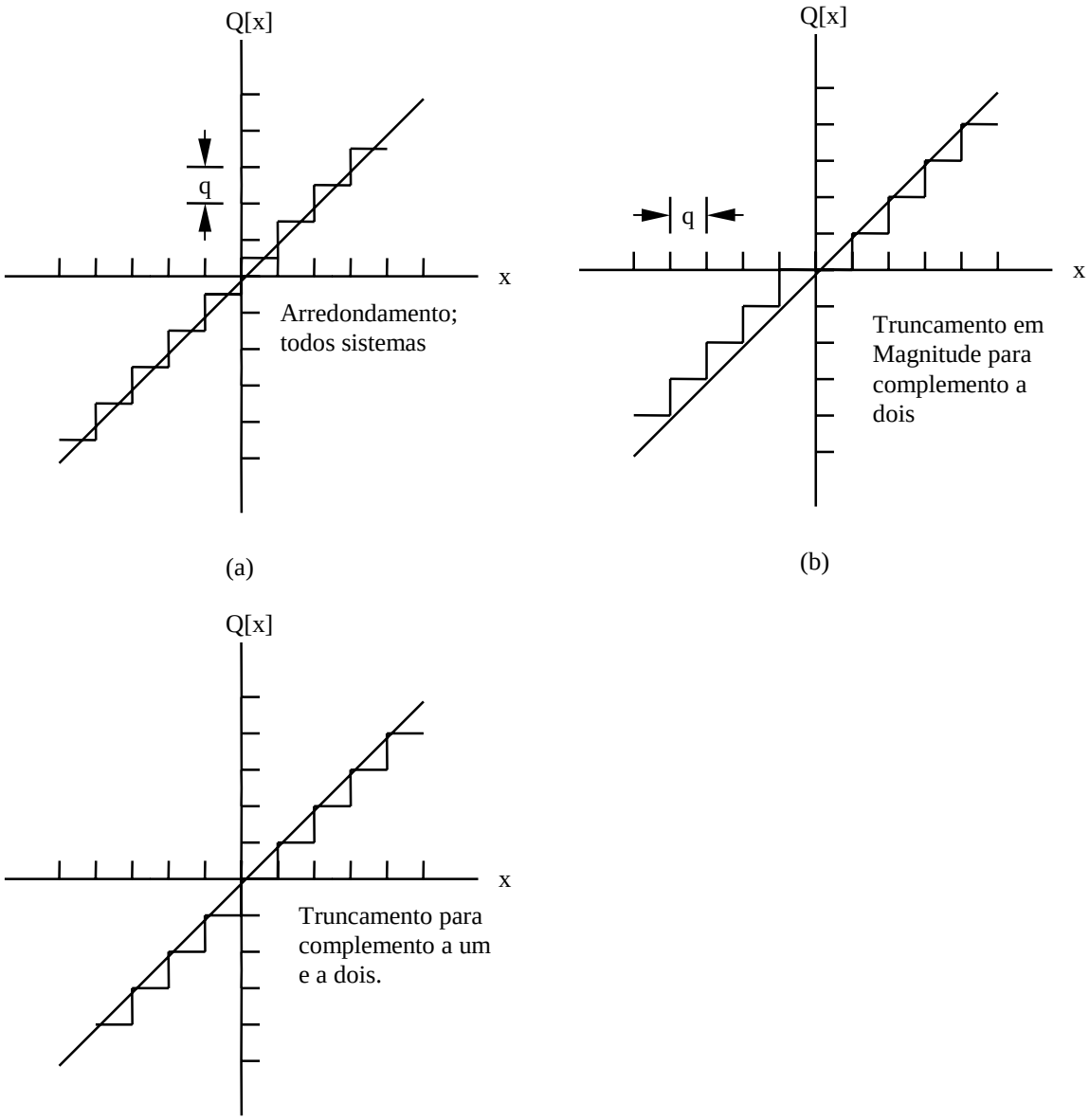

Figura 1.3 - Valor de Q[x] em função de x.

A saída do multiplicador pode ser expressa como

$$Q[c_i x[n]] = c_i x[n] + e[n] \tag{1.30}$$

onde o primeiro termo é o produto exato e o segundo o erro correspondente, respectivamente. O modelo que descreve este multiplicador é mostrado na Figura 1.4.

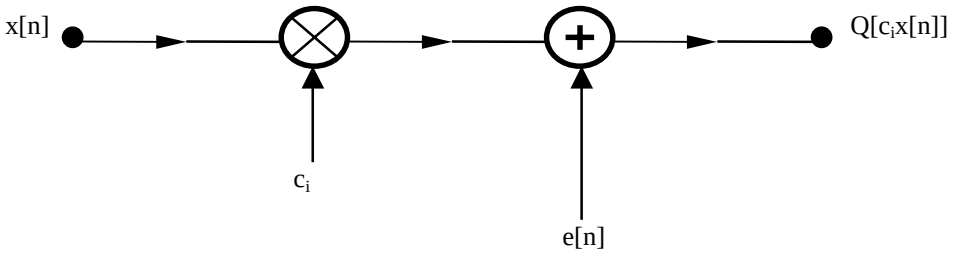

Figura 1.4 - Modelagem do ruído gerado em um multiplicador.

Além dos problemas de quantização, se a amplitude do sinal de entrada exceder a faixa do registrador um *overflow* vai ocorrer, e o sinal de saída será severamente distorcido. Mas, por outro lado, se o sinal de entrada for muito pequeno, o filtro operará ineficientemente e a relação sinal ruído será muito ruim (observe-se que o ruído inserido não depende da amplitude do sinal).

A técnica de escalamento consiste em multiplicar a entrada do filtro por uma constante λ de tal forma que não haja *overflow* (vide Figura 1.5). Mas, como visto anteriormente, tem que haver uma solução de compromisso: se λ for muito grande corre-se o risco de estouro do registrador. Se muito pequeno, a relação sinal ruído não é boa.

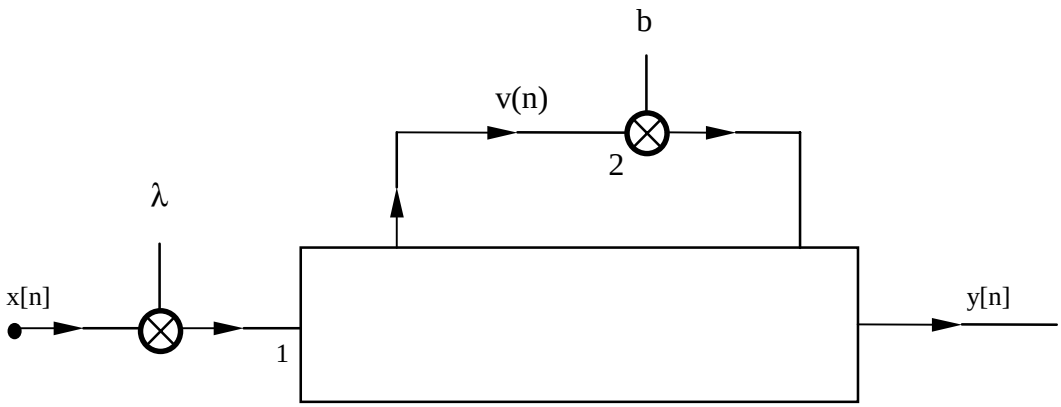

Figura 1.5 - Escalamento de um filtro digital.

Existem dois métodos para determinação da constante λ. O primeiro considera a resposta no tempo do filtro, e o valor da constante de escalamento é definido como

$$\lambda \leq \frac{1}{\sum_{k=0}^{\infty} |f(k)|} \tag{1.31}$$

onde F(z) é a função de transferência entre os nós 1 e 2 da Figura 1.5. O segundo método considera a resposta em freqüência, e define-se a constante de escalamento como

$$\lambda \leq \frac{1}{\|F(z)\|_p} \tag{1.32}$$

ou seja, em função da norma da função F(z), sendo p = 2 ou p = ∞ (norma quadrática ou infinita, respectivamente [9]). O segundo método é mais eficiente, e é o único abordado neste trabalho.

1.6 - Ciclos Limite em Filtros Digitais

Ciclos limite são oscilações parasitas que podem surgir na saída do filtro quando o sinal de entrada é muito pequeno (constante), quando o sinal de entrada é zero ou então quando ocorre um *overflow*, e que se mantêm mesmo depois de retirada a excitação do sistema.

Essas oscilações estão associadas à realimentação dos filtros recursivos e ao comportamento não linear introduzido pelos quantizadores. Elas podem ocorrer inclusive quando os pólos estão dentro do círculo de raio unitário [9].

Pode-se dividir essas oscilações em dois grupos: ciclos limite de quantização, ou granulares, e ciclos limite devidos à o*verflow*.

1.6.1 - Ciclos Limite de Quantização

Nos métodos tradicionais de análise de filtros digitais, é assumido que o nível do sinal é muito maior que o passo de quantização 'q' usado. Isto possibilita assumir independência estatística de cada amostra. Em muitas ocasiões, como em aplicações de sinais de música, o nível de sinal pode ser muito pequeno (por curtos períodos de tempo) ou constante. Sob estas

condições os erros de quantização tendem a estar fortemente correlacionados, e podem levar o filtro à instabilidade, de forma a gerar os ciclos limite de quantização. Este fenômeno é conhecido como *deadband* [9].

De acordo com a entrada, estes ciclos limite podem ser classificados como: ciclos limite à entrada zero ou ciclos limite à entrada constante, mas são sempre provenientes da quantização dos bits menos significativos do sinal [9], [10].

1.6.2 - Ciclos Limite Devidos a Overflow

São oscilações que podem ocorrer quando, por várias vezes, os sinais estouram a faixa dinâmica dos registradores internos. O *overflow* em si não é o mais preocupante. O que tem de ser garantido é que o tempo de recuperação do sistema seja menor do que o intervalo entre a ocorrência de dois *overflows* consecutivos [10], [11]. Em outras palavras, é necessário assegurar que os *overflows*, se ocorrerem, não sejam muito freqüentes.

Ao contrário de ciclos limite de quantização, este fenômeno influencia os bits mais significativos da representação numérica em ponto fixo.

1.7 - Condições Para Eliminação de Ciclos Limite

Basicamente, há duas estratégias adotadas para tratar os ciclos limite. A primeira consiste na busca de uma tamanho de palavra suficientemente grande, de tal forma a garantir que a amplitude dos ciclos limite seja pequena o bastante para não afetar o desempenho do sistema. A outra estratégia trata explicitamente da eliminação dos ciclos limite, ou seja, busca assegurar que eles não ocorram. Nesse caso, é fundamental a utilização de uma topologia adequada para garantir tal eliminação. Neste aspecto, os filtros realizados via representação no espaço de estados são bastante interessantes, visto que é possível, em tal caso, assegurar a ausência de ciclos limite [1], [3]-[6], [11]-[13].

Nos próximos tópicos apenas será abordada a segunda estratégia de forma sintética. Para uma discussão detalhada, ver [6], [12] e [13].

1.7.1 - Eliminação de Ciclos Limite com Entrada Zero

Considere-se o filtro digital representado na Figura 1.6. Observe-se que todas as não linearidades deste sistema foram destacadas, e ficam fora da subrede A. Portanto, o bloco `A` é uma subrede linear que contém apenas somadores, multiplicadores e interconexões, mas não possui atrasos e nem quantizadores. A caracterização no espaço de estados deste filtro é dada por

$$\mathbf{v}[n] = \mathbf{A}\mathbf{q}[n] + \mathbf{b}x[n] \tag{1.33}$$

$$y[n] = \mathbf{c}^T\mathbf{q}[n] + dx[n] \tag{1.34}$$

Se x[n] = 0, pode-se rescrever estas equações como

$$\mathbf{v}[n] = \mathbf{A}\mathbf{q}[n] \tag{1.35}$$

$$\mathbf{q}[n+1] = \tilde{\mathbf{v}}[n] \tag{1.36}$$

A estrutura da Figura 1.6 será imune a ciclos limite, no caso de entrada zero, se, e somente se, existir uma matriz **D**, NxN, diagonal definida positiva, tal que

$$\mathbf{A}^T\mathbf{D}\mathbf{A} \leq \mathbf{D} \tag{1.37}$$

e a quantização for realizada por truncamento em magnitude.

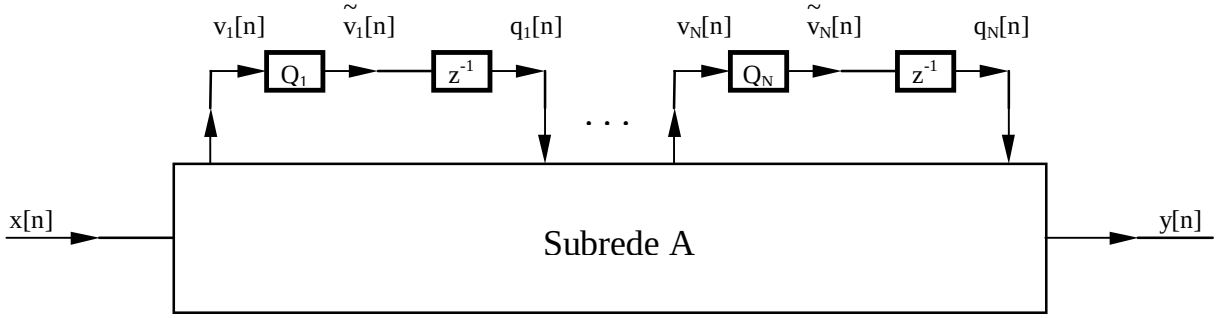

Figura 1.6 - Filtro Digital de ordem N incorporando não linearidades.

Para demonstrar tal condição, uma função de Lyapunov, que caracteriza a potência armazenada nos atrasos do sistema, é construída, e seu comportamento é analisado [9]. A quantização é realizada sobre os estados, ou seja,

$$\tilde{v}_k[n] = Q_k[v_k[n]] \qquad \text{para } k = 1,2, ..., N \qquad (1.38)$$

Seja, então, a forma quadrática

$$p[\mathbf{q}[n]] = \mathbf{q}^T[n]\mathbf{D}\mathbf{q}[n] \qquad (1.39)$$

O incremento em p[**q**[n]], em um ciclo de operação do filtro, pode ser expresso como

$$\Delta p[\mathbf{q}[n]] = p[\mathbf{q}[n + 1]] - p[\mathbf{q}[n]] \qquad (1.40)$$

e pelas equações (11.36), (11.38) e (11.40) tem-se que

$$\Delta p[\mathbf{q}[n]] = - \mathbf{q}^T[n]\mathbf{D}\mathbf{q}[n] + \tilde{\mathbf{v}}^T[n]\,\mathbf{D}\,\tilde{\mathbf{v}}[n] \qquad (1.41)$$

Usando-se as equações (1.35) e (1.41) obtém-se que

$$\Delta p[\mathbf{q}[n]] = - \mathbf{q}^T[n]\mathbf{D}\mathbf{q}[n] + \tilde{\mathbf{v}}^T[n]\,\mathbf{D}\,\tilde{\mathbf{v}}[n] + [\mathbf{A}\mathbf{q}[n]]^T\mathbf{D}[\mathbf{A}\mathbf{q}[n]] - \tilde{\mathbf{v}}^T[n]\,\mathbf{D}\,\mathbf{v}[n] \qquad (1.42)$$

ou seja,

$$\Delta p[\mathbf{q}[n]] = - \mathbf{q}^T[n](\mathbf{D}-\mathbf{A}^T\mathbf{D}\mathbf{A})\mathbf{q}[n] - \sum_{k=1}^{N} [\, v_k^2[n] - \tilde{v}_k^2[n]\,]d_{kk} \qquad (1.43)$$

onde d_{kk}, para k = 1,2, ..., N, são os elementos de **D.** Agora, se

$$\mathbf{q}^T[n](\mathbf{D}-\mathbf{A}^T\mathbf{D}\mathbf{A})\mathbf{q}[n] \geq 0 \qquad (1.44)$$

e os sinais $v_k[n]$ forem quantizados de forma que

$$|\tilde{v}_k[n]| \leq |v_k[n]| \qquad \text{para } k = 1, 2, ..., N \tag{1.45}$$

o que corresponde ao truncamento em magnitude, então a equação (1.42) leva a

$$\Delta p[\mathbf{q}[n]] \leq 0 \tag{1.46}$$

o que mostra que a energia acumulada nos atrasos é decrescente. Assim, sob tais condições, o filtro digital, com entrada zero, terá sua saída decrescendo em direção de zero.

Para os filtros digitais de segunda ordem representados no espaço de estados, a condição expressa em (1.37) será obedecida se

$$a_{12}\, a_{21} \geq 0 \tag{1.47}$$

ou

$$a_{12}\, a_{21} < 0 \quad \text{e} \quad |a_{11} - a_{22}| + \det(\mathbf{A}) \leq 1 \tag{1.48}$$

o que corresponde à caracterização da garantia de ausência de ciclos limite, sob entrada zero, em função dos parâmetros da matriz **A** [13].

1.7.2 - Eliminação de Ciclos Limite com Entrada Constante

Ciclos limite também podem ser gerados se a entrada assumir um valor constante por um determinado período de tempo. Particularmente, o caso de entrada zero é um caso especial do que se chama ciclos limite de entrada constante.

Os filtros digitais imunes a ciclos limite com entrada zero não estão garantidamente livres de ciclos limite de entrada constante. As condições suficientes para eliminação destes ciclos limite serão aqui estabelecidas.

Teorema 1 [6]: Seja a estrutura da Figura 1.7 imune a ciclos limite de entrada zero. Ela é descrita por

$$\mathbf{x}[k+1] = Q[\mathbf{A}\mathbf{x}[k] + \mathbf{b}u[k]] \tag{1.49}$$

$$y[k+1] = \mathbf{c}\mathbf{x}[k] + du[k] \tag{1.50}$$

onde Q[*] é o valor de [*] quantizado por truncamento em magnitude.

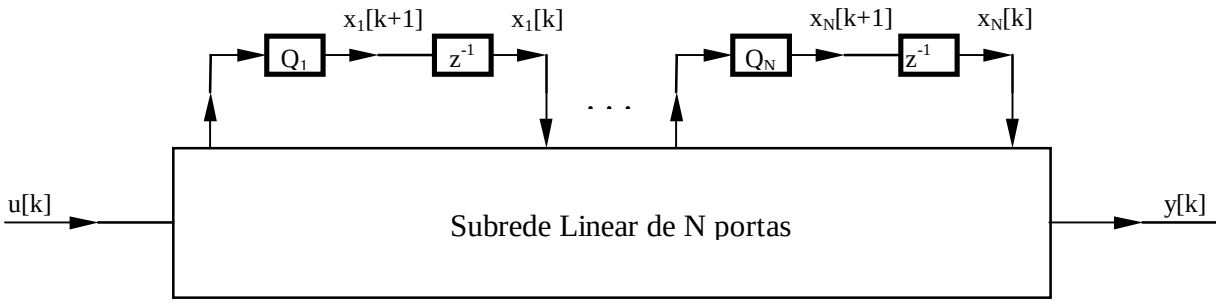

Figura 1.7 - Filtro digital de ordem N.

Então, ciclos limite de entrada constante podem ser eliminados pela modificação da estrutura da Figura 1.7, como mostrado na Figura 1.8, onde **P** é dado por

$$\mathbf{P} = [p_1\ p_2\ \ldots\ p_N]^T = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} \tag{1.51}$$

desde que $\mathbf{P}u_0$ seja representável exatamente no comprimento de palavra da máquina, sendo u_0 a entrada constante.

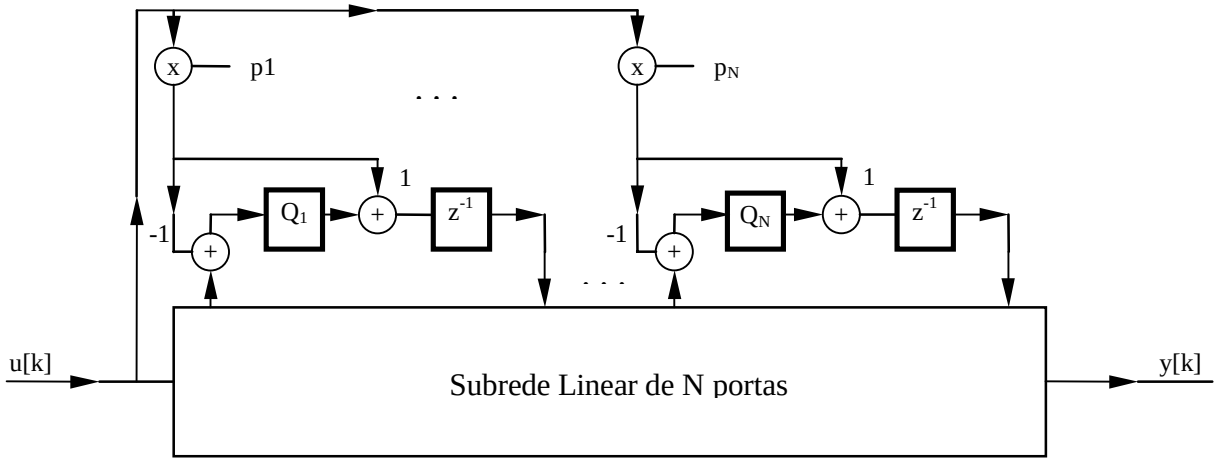

Figura 1.8 - Filtro digital de ordem N modificado.

Prova : É assumido que a estrutura da Figura 1.7 é livre de ciclos limite devidos à entrada zero. Logo,

$$\mathbf{x}[k + 1] = Q[\mathbf{A}\mathbf{x}[k]] \tag{1.52}$$

descreve um sistema estável, ou seja,

$$\lim_{k \to \infty} \mathbf{x}[k] = [0\ 0\ ...\ 0]^T \tag{1.53}$$

Se a entrada é constante, isto é, $u[k] = u_0$, a estrutura modificada é caracterizada por

$$\mathbf{x}[k +1] = Q[\mathbf{A}\mathbf{x}[k] - \mathbf{P}u_0 + \mathbf{b}u_0] + \mathbf{P}u_0 \tag{1.54}$$

Se (1.51) se mantém, obtém-se que

$$\mathbf{x}[k +1] = Q[\mathbf{A}\mathbf{x}[k] - (\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}u_0 + (\mathbf{I} - \mathbf{A})(\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}u_0] + \mathbf{P}u_0 \tag{1.55}$$

$$\mathbf{x}[k + 1] = Q[\mathbf{A}\{\mathbf{x}[k] - \mathbf{P}u_0\}] + \mathbf{P}u_0 \tag{1.56}$$

o que se reduz a

$$\mathbf{x}'[k + 1] = Q[\mathbf{A}\mathbf{x}'[k]] \tag{1.57}$$

onde

$$\mathbf{x}'[k] = \mathbf{x}[k] - \mathbf{P}u_0 \tag{1.58}$$

É evidente que (1.58) tem a mesma forma de (1.52), a não ser por uma translação das variáveis de estado, e desta forma também representa um sistema estável. Assim, a estabilidade pode ser garantida se (1.51) é satisfeita, e se $\mathbf{P}u_0$ for perfeitamente representado na máquina [6] [11].

Um caso particular de filtros digitais no espaço de estados imunes a cilcos limite no caso de entrada constante é tratado em [6]. Neste caso o vetor **P** assume valores inteiros, e são consideradas três estruturas

$$\text{Estrutura Tipo I:} \qquad \mathbf{P} = [\pm 1 \, , \, 0] \tag{1.59}$$

$$\text{Estrutura Tipo II:} \qquad \mathbf{P} = [0 \, , \, \pm 1] \tag{1.60}$$

$$\text{Estrutura Tipo III:} \qquad \mathbf{P} = [\pm 1 \, , \, \pm 1] \tag{1.61}$$

Para cada caso, o vetor **b** pode ser determinado, a partir de (1.51). Assim, tem-se, respectivamente,

$$b_1 = \pm(1 - a_{11}) \, , \, b_2 = \pm \, a_{21} \tag{1.62}$$

$$b_1 = \pm \, a_{12} \, , \, b_2 = \pm(1 - a_{22}) \tag{1.63}$$

$$b_1 = \pm(1 - a_{11}) \quad \mp a_{12}, \quad b_2 = \mp a_{21} \pm (1 - a_{22}) \tag{1.64}$$

A escolha de **P** como acima gera três estruturas imunes a ciclo limites no caso de entrada zero e entrada constante. Tais estruturas serão abordadas mais detalhadamente no próximo capítulo, especialmente a estrutura Tipo III. Uma outra propriedade interessante de tais estruturas é que os coeficientes do vetor **b**, como caracterizado acima, são gerados por combinações dos coeficientes da matriz **A**, o que faz menos complexas, do ponto de vista computacional [6].

1.7.3 - Eliminação de Ciclos Limite Devidos a Overflow

O o*verflow* pode ser evitado pelo uso das regras de escalamento já vistas na seção 1.5. Como visto, pode-se usar um método que garante a não existência de *overflow*, mas que, como desvantagem, apresenta uma relação pobre entre sinal e ruído. Alternativamente, pode-se utilizar um método que não garante a inexistência de *overflow*, embora a sua probabilidade

de ocorrência seja garantidamente pequena, com a vantagem de permitir uma melhor relação sinal ruído.

A melhor solução é permitir alguns *overflows* ocasionais, mas prevenindo a ocorrência de ciclos limite. Uma solução para isto é incorporar um mecanismo especial aos somadores, chamado de saturação aritmética [14]. A função de transferência que caracteriza este elemento está descrita na Figura 1.9. Ela pode ser caracterizada por

$$Q(x) = \begin{cases} x & se\,|\,x\,| < M \\ M & se\,|\,x\,| \geq M \end{cases} \tag{1.65}$$

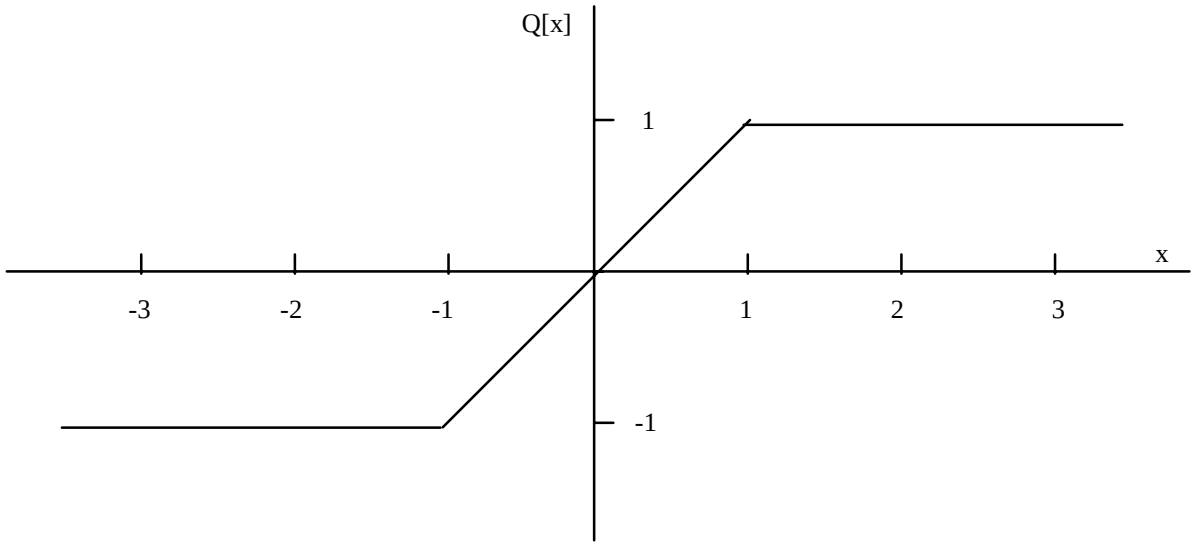

Figura 1.9 - Função de transferência correspondente à saturação aritmética.

Através da saturação aritmética, quando um *overflow* é detectado, o valor máximo substitui o sinal x, com o sinal apropriado. A utilização da saturação aritmética, conforme já está bem caracterizado na literatura [21-22], é suficiente para eliminar ciclos limite em caso de *overflow*, uma vez que o filtro seja imune a ciclos limite devidos à entrada zero.

1.8 - A Contribuição deste Trabalho

Partindo de uma estrutura de segunda ordem no espaço de estados imune a ciclos limite já conhecida da literatura [6], este trabalho discute uma forma de torná-la ótima em

relação ao ruído na saída do filtro. Através da variação de um parâmetro, foi possível fazer um estudo paramétrico de como o ruído na saída deste filtro varia em função do referido parâmetro, de forma a encontrar a realização ótima. Uma nova maneira de sintetizar tal estrutura é, então, proposta, isto é descrito no Capítulo 3. Adicionalmente, é feita uma análise comparativa desta estrutura com algumas estruturas já consagradas na literatura, no Capítulo 4. Previamente no Capítulo 2, são discutidas em detalhes as estruturas no espaço de estados usadas em tal comparação.

CAPÍTULO II

Realização de Filtros Digitais de Segunda Ordem Imunes a Ciclos Limite

Para realização de filtros digitais de ordem elevada, é usual a implementação na forma cascata ou paralela de blocos de segunda ordem diretos. Tal implementação traz como vantagens a redução da sensibilidade da rede, quando os coeficientes multiplicadores são quantizados, e a redução do ruído na saída do filtro [15].

Quando estes filtros têm banda passante estreita, porém, a implementação usando blocos de segunda ordem diretos é muito ineficiente. Com a redução da banda passante o ruído na saída do filtro assim implementado tende a crescer muito [16], e a relação sinal ruído fica muito deteriorada.

Em tal situação, as estruturas no espaço de estado têm se mostrado muito eficientes. Especificamente, inclusive, a rede no espaço de estados de mínimo ruído, têm uma característica importante, que é a invariância do ruído na saída em relação à largura da banda passante do filtro [1] [2]. Por isto, redes no espaço de estados, são adequadas para implementar filtros de banda estreita. Porém, há um inconveniente: a complexidade computacional destas estruturas é grande, dado o número de multiplicações necessários para sua computação [1]. Para solucionar este problema, filtros digitais de ordem elevada são implementados na forma cascata ou paralela de blocos de segunda ordem, onde cada bloco é um estrutura no espaço de estados de mínimo ruído [3]. Desta forma há um bom compromisso entre o baixo ruído na saída do filtro e a elevada complexidade computacional.

Os filtros digitais implementados em processadores programáveis têm merecido grande destaque nos últimos tempos. Conhecidos como DSP's (*digital signal processor*) estes *chips* têm uma arquitetura especialmente projetada para implementação de filtros digitais. A principal diferença destes processadores em relação aos processadores tradicionais é que a operação de multiplicação é computada em um único ciclo de *clock* de barramento [23].

Seguindo tal linha de implementação de filtros digitais, alguns trabalhos de pesquisa têm se dedicado ao desenvolvimento de estruturas que primem por um bom desempenho a nível de ruído na saída do filtro e uma baixa complexidade computacional, além de outra característica importante que é a imunidade a ciclos limite [3]-[6], [17]-[18].

Dentre estas, este capítulo faz uma revisão de algumas estruturas, com destaque para sua síntese, análise do ruído e caracterização da imunidade a ciclos limite.

2.1 - Estrutura de Mínimo Ruído [1][3]

Seja a estrutura de segunda ordem dado na Figura 2.1, a qual é descrita pelas equações

$$\mathbf{X}[n+1] = \mathbf{A}\mathbf{X}[n] + \mathbf{b}U[n] + \mathbf{e}[n] \tag{2.1}$$

$$Y[n] = \mathbf{c}^T\mathbf{X}[n] + e_3[n] \tag{2.2}$$

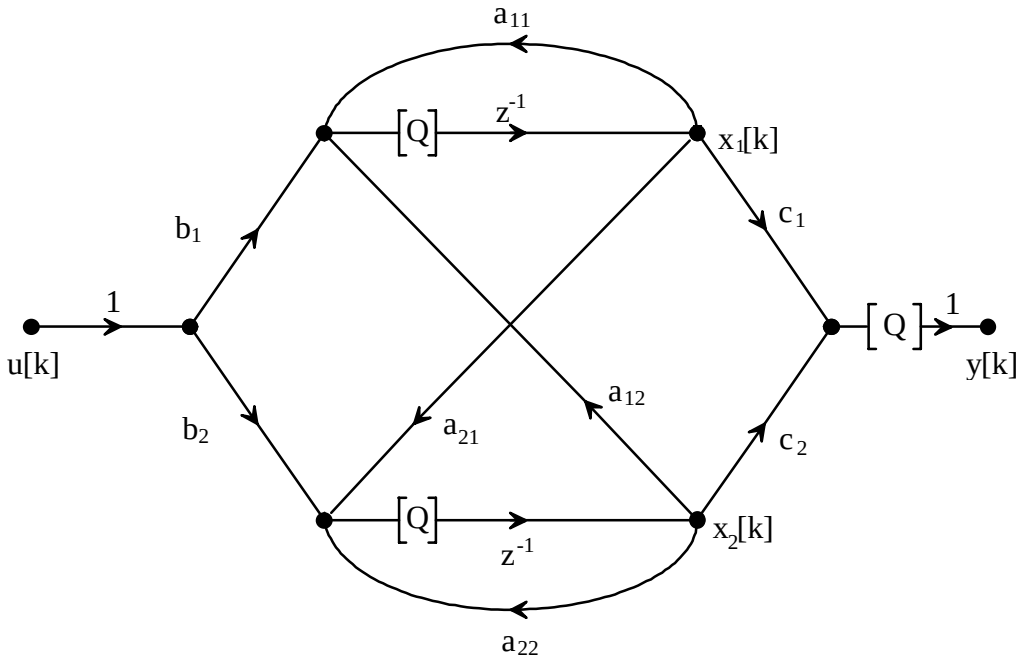

Figura 2.1 - Estrutura de mínimo ruído.

onde $e_i[n]$, i = 1,2 e 3, são fontes de ruído devido à quantização dos produtos e

$$\mathbf{x}[n] = [x_1[n] \;\; x_2[n]]^T$$
$$\mathbf{e}^T[n] = [e_1[n] \;\; e_2[n]] \tag{2.3}$$

Para tal estrutura $F_1(z)$, $F_2(z)$, $G_1(z)$ e $G_2(z)$ são as funções de transferência da entrada para os nós $x_1[n]$ e $x_2[n]$ e dos nós $x_1[n + 1]$ e $x_2[n + 1]$ para a saída, respectivamente. Com esta notação, pode-se definir os vetores **f**(z) e **g**(z) como

$$\mathbf{f}^T(z) = [F_1(z) \;\; F_2(z)] \qquad \text{e} \qquad \mathbf{g}^T(z) = [G_1(z) \;\; G_2(z)] \tag{2.4}$$

e pode-se mostrar que [1]

$$\mathbf{f}(z) = (z\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} \quad \text{e} \quad \mathbf{g}(z) = (z\mathbf{I} - \mathbf{A}^{T})^{-1}\mathbf{c} \tag{2.5}$$

Se a realização da Figura 2.1 é representada pelo conjunto de parâmetros $[\mathbf{A},\mathbf{b},\mathbf{c}]$, pode-se obter uma nova realização através de uma transformação de similaridade. Desta forma, se $\tilde{\mathbf{x}}(n) = \mathbf{T}\mathbf{x}(n)$, onde $\mathbf{T}$ é a matriz de transformação, uma nova realização $[\tilde{\mathbf{A}},\tilde{\mathbf{b}},\tilde{\mathbf{c}}]$ é obtida, onde

$$\tilde{\mathbf{A}} = \mathbf{T}\mathbf{A}\mathbf{T}^{-1}, \qquad \tilde{\mathbf{b}} = \mathbf{T}\mathbf{b}, \qquad \tilde{\mathbf{c}} = \mathbf{c}^{T}\mathbf{T}^{-1} \tag{2.6}$$

Através da equação (2.2) pode-se mostrar que

$$\tilde{\mathbf{f}}(z) = \mathbf{T}\mathbf{f}(z) \qquad \text{e} \qquad \tilde{\mathbf{g}}(z) = \mathbf{T}^{-1}\mathbf{g}(z) \tag{2.7}$$

A realização $[\tilde{\mathbf{A}},\tilde{\mathbf{b}},\tilde{\mathbf{c}}]$ tem mínimo ruído na saída, no contexto de escalamento L_2, se e, somente se,

$$\tilde{\mathbf{W}} = \mathbf{D}\tilde{\mathbf{K}}\mathbf{D} \tag{2.8}$$

$$\tilde{K}_{ii}\tilde{W}_{ii} = \tilde{K}_{jj}\tilde{W}_{jj} \qquad \text{para todo } i, j \tag{2.9}$$

onde $\mathbf{D}$ é uma matriz diagonal e $\tilde{\mathbf{K}} = \{\tilde{K}_{ij}\}$ e $\tilde{\mathbf{W}} = \{\tilde{W}_{ij}\}$ são matrizes dadas por

$$\tilde{\mathbf{K}} = \frac{1}{2\pi j}\oint_{\Gamma}\tilde{\mathbf{f}}(z)\tilde{\mathbf{f}}^{T}(z^{-1})z^{-1}dz \tag{2.10}$$

e

$$\tilde{\mathbf{W}} = \frac{1}{2\pi j}\oint_{\Gamma}\tilde{\mathbf{g}}(z)\tilde{\mathbf{g}}^{T}(z^{-1})z^{-1}dz \tag{2.11}$$

e são equivalentes aos gramianos de controlabilidade e de observabilidade, respectivamente.

Das equações (2.10) e (2.11) tem-se que

$$K_{ii} = \frac{1}{2\pi j}\oint_{\Gamma} \tilde{F}_i(z)\,\tilde{F}_i^{\,T}(z^{-1})z^{-1}dz \qquad (2.12)$$

$$= \frac{1}{w_s}\int_0^{w_s} |\tilde{F}_i(e^{jwt})|^2\, dw$$

ou seja, o escalamento L_2 é feito de forma que

$$K_{ii} = \|\tilde{F}_i(z)\|_2^2 = 1 \qquad \text{para todo i,} \qquad (2.13)$$

e então a segunda condição para mínimo ruído na equação (2.9) assume a forma

$$\tilde{W}_{ii} = \tilde{W}_{jj} \qquad (2.14)$$

e da equação (2.11) tem-se que

$$\|\tilde{G}_i(z)\|_2^2 = \|\tilde{G}_j(z)\|_2^2 \qquad \text{para todo i,j} \qquad (2.15)$$

O método acima é valido para um filtro no espaço de estados de ordem N. Este mesmo método pode ser particularizado para uma seção de ordem 2. Para tanto, basta notar que a equação (2.8) é satisfeita se e somente se $\mathbf{D} = \rho\mathbf{I}$. Observando as equações (2.13) e (2.14), a equação (2.8) pode ser expressa como:

$$\tilde{\mathbf{W}} = \rho^2\,\tilde{\mathbf{K}} \qquad (2.16)$$

e assim, dado que $\tilde{\mathbf{W}}$ e $\tilde{\mathbf{K}}$ são matrizes de elementos diagonais iguais [1] [3], a equação (2.16) assume a forma

$$\mathbf{D} = \rho^2\,\mathbf{J}\tilde{\mathbf{K}}\mathbf{J} \qquad (2.17)$$

onde

$$\mathbf{J} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Por sua vez, a equação (2.17) é satisfeita por uma rede na qual

$$\tilde{\mathbf{A}}^{T} = \mathbf{J}\tilde{\mathbf{A}}\mathbf{J} \tag{2.18}$$

$$\tilde{\mathbf{c}} = \rho\mathbf{J}\tilde{\mathbf{b}} \tag{2.19}$$

Em termos dos elementos de $[\tilde{\mathbf{A}},\tilde{\mathbf{b}},\tilde{\mathbf{c}}]$ as equações (2.18) e (2.19) tomam a forma

$$\tilde{a}_{11} = \tilde{a}_{22} \tag{2.20}$$

$$\frac{\tilde{b}_1}{\tilde{b}_2} = \frac{\tilde{c}_2}{\tilde{c}_1} \tag{2.21}$$

Em relação à síntese da estrutura, tendo em conta as equações (2.20) e (2.21), pode-se obter, a partir da função de transferência desejada

$$a_{11} = a_{22} = -\frac{\alpha_1}{2} \tag{2.22}$$

$$a_{12} = -\sqrt{\alpha_2 - \frac{\alpha_1^{\,2}}{4}} = -a_{21} \tag{2.23}$$

$$b_1 = \sqrt{\frac{(\sqrt{\beta_2^{\,2} - \beta_1\beta_2\alpha_1 + \beta_1^{\,2}\alpha_2}) + \beta_2 + a_{11}\beta_1}{2a_{21}}} \tag{2.24}$$

$$b_2 = \frac{\beta_2}{2b_1} \tag{2.25}$$

$$c_1 = b_2 \tag{2.26}$$

$$c_2 = b_1 \tag{2.27}$$

Depois de encontrada a solução acima, caracterizada por $[\mathbf{A},\mathbf{b},\mathbf{c}]$ o vetor $\mathbf{f}^{T}(z) = [F_1(z) \;\; F_2(z)]$ pode ser obtido, ou seja, são obtidas as funções de transferência

$$F_1(z) = \frac{b_1 z + (b_2 a_{12} - b_1 a_{22})}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.28}$$

e

$$F_2(z) = \frac{b_2 z + (b_1 a_{21} - b_2 a_{11})}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.29}$$

De posse de tais funções é possível escalar a rede, através de uma transformação de similaridade **T** específica que leve à condição expressa em (2.13). Ela é dada por

$$\mathbf{T} = \begin{bmatrix} \|F_1(z)\|_2 & 0 \\ 0 & \|F_2(z)\|_2 \end{bmatrix} \tag{2.30}$$

De forma semelhante pode-se encontrar o vetor $\tilde{\mathbf{g}}^{T}(z) = [\tilde{G}_1(z)\ \tilde{G}_2(z)]$, que representa as funções de transferência dos nós $x_1[n + 1]$ e $x_2[n + 1]$ para a saída, ou seja pode-se obter

$$\tilde{G}_1(z) = \frac{c_1 z + (c_2 a_{21} - c_1 a_{22})}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.31}$$

e

$$\tilde{G}_2(z) = \frac{c_2 z + (c_1 a_{12} - c_2 a_{11})}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.32}$$

A partir de tais funções, pode-se calcular a densidade espectral relativa do ruído na saída do filtro já escalado, dada por

$$RPSD(w) = \sum_{i=1}^{2} |\tilde{G}_i(e^{jw})|^2 + 1 \tag{2.33}$$

Assim como calcular a variância relativa do ruído na saída do filtro já escalado, dada por

$$\sigma_0^2 = 1 + \sum_{i=1}^{2} \|\tilde{G}_i(e^{jw})\|_2^2 \tag{2.34}$$

Em termos de filtros digitais de ordem elevada, uma boa implementação é aquela que usa a forma paralela de blocos de segunda ordem ótimos. No caso de *n* seções a densidade espectral relativa do ruído na saída do filtro escalado é dada por

$$RPSD(w) = \sum_{j=1}^{n} \sum_{i=1}^{2} |\tilde{G}_i(e^{jw})|^2 + 1 \tag{2.35}$$

Da mesma forma a variância relativa total torna-se

$$\frac{\sigma_0^2}{\sigma_i^2} = 1 + \sum_{j=1}^{n} \sum_{i=1}^{2} ||\tilde{G}_i(e^{jw})||_2^2 \tag{2.36}$$

Se aplicados os resultados em [12] e [13] pode-se verificar que a estrutura de mínimo ruído é imune a ciclos limite devidos à entrada zero, desde que a quantização seja realizada por truncamento em magnitude, que é a estratégia comumente adotada. Para o tratamento de *overflow*, de acordo com os resultados em [21] e [22], se usada a saturação aritmética, a estrutura é imune também a ciclos limite devidos a *overflow*.

Finamente, vale à pena destacar que a síntese acima discutida foi originalmente desenvolvida para o escalamento do filtro usando norma quadrática. Porém, se for usado $||F_i(z)||_\infty$ na equação (2.30), os resultados ainda são válidos, embora o ruído na saída não seja mais mínimo [3]. Para que o ruído seja muito próximo do mínimo, usando escalamento em norma infinita, os resultados em [5] mostram que o filtro de mínimo ruído deve ser projetado usando-se o escalamento em norma quadrática, e em seguida deve ser novamente escalado, agora em norma infinita.

2.2 - Estruturas Imunes a Ciclos Limite no Caso de Entrada Constante

Conforme citado acima, a estrutura ótima tem características importantes: apresenta o mínimo ruído na saída do filtro, apresenta imunidade a ciclos limites de entrada zero, apresenta imunidade a ciclos limites devidos a *overflow*, e apresenta invariância do ruído em relação à largura de banda passante [1]-[3]. Embora essa estrutura tenha todas essas vantagens, ela não é garantidamente imune a ciclos limite devidos à entrada constante.

Usando-se as condições estabelecidas na seção 1.7.2, é possível a eliminação de tais ciclos limite. Partindo-se, então, de uma estrutura imune a ciclos limite no caso de entrada zero, algumas estruturas já bem conhecidas da literatura podem ser derivadas, de modo a eliminar ciclos limite de entrada constante. Algumas destas estruturas são revisadas nesta seção.

2.2.1 - Estrutura do Tipo I [6]

Parte-se de uma matriz **A** que representa um filtro no espaço de estados de segunda ordem, já escalado, ou seja

$$\mathbf{A} = \begin{bmatrix} a & \dfrac{-\delta}{\sigma} \\ \delta\sigma & a \end{bmatrix} \tag{2.37}$$

onde α e ξ são constantes relacionadas com os pólos da função de transferência do filtro, tal que

$$p_i = a \pm j\delta \qquad \text{para i = 1, 2} \tag{2.38}$$

A constante σ depende do escalamento usado, não interfere no posicionamento dos pólos e irá influenciar diretamente no ruído do filtro.

Como foi visto no seção 1.7.2, a técnica de eliminação de ciclos limite devidos à entrada constante consiste em fazer uma mudança na estrutura, escolhendo-se um vetor **P** apropriado. Para este caso $\mathbf{P} = [1 \;\; 0]^T$. De acordo com (1.51) tem-se que

$$b_1 = (1 - a_{11}), \quad b_2 = -a_{21} \tag{2.39}$$

A Figura 2.2 ilustra tal estrutura.

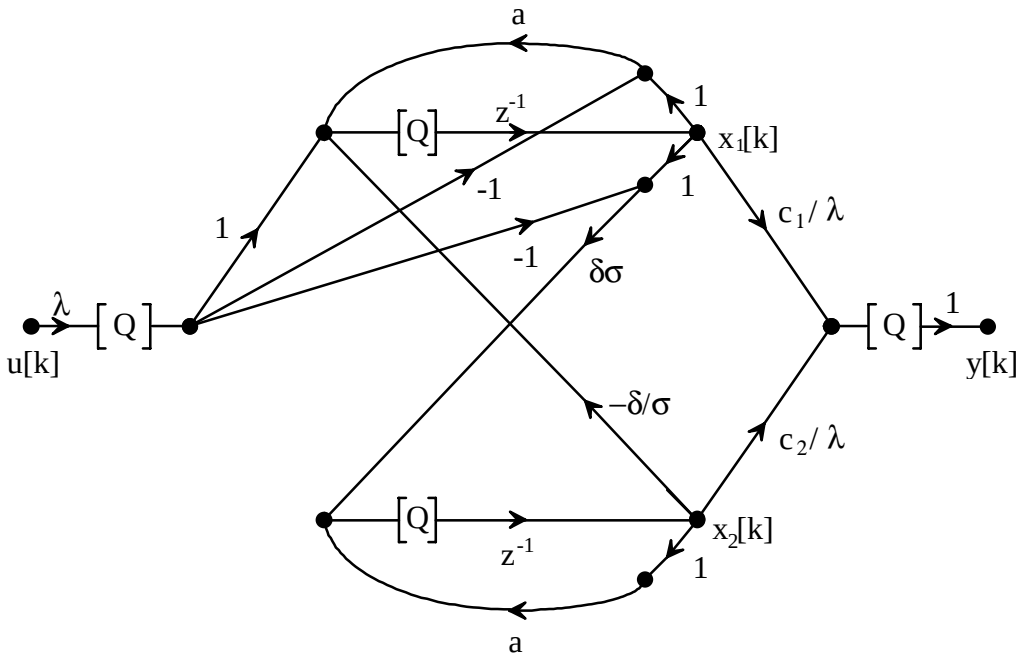

Figura 2.2 - Estrutura do tipo I.

As constantes multiplicadoras podem ser deduzidas a partir das equações (1.9), (1.10) (2.20) e (2.37), e são

$$a_{11} = a_{22} = a \tag{2.40}$$

$$a_{12} = -\frac{\delta}{\sigma} \tag{2.41}$$

$$a_{21} = \sigma\delta \tag{2.42}$$

onde

$$a = -\frac{\alpha_1}{2} \tag{2.43}$$

$$\delta = \sqrt{\alpha_2 - \frac{\alpha_1^2}{4}} \tag{2.44}$$

Conhecidas as matrizes **A** e **b**, o vetor **c** pode ser deduzido a partir das equações (1.13) e (1.14), obtendo-se

$$c_1 = \frac{\beta_1 + \beta_2}{1 + \alpha_1 + \alpha_2} \tag{2.45}$$

$$c_2 = \frac{-(\alpha_1 + \alpha_2)\beta_1 + (2 + \alpha_1)\beta_2}{2\sigma\delta(1 + \alpha_1 + \alpha_2)} \tag{2.46}$$

Como pode ser visto, os coeficientes b_1 e b_2 são formados sem a necessidade de multiplicadores adicionais, e, por conseqüência, esta estrutura requer menos multiplicadores do que a seção ótima. Ou seja, do ponto de vista computacional esta estrutura é menos complexa do que a estrutura ótima.

A relação sinal ruído de um filtro digital implementado em ponto fixo pode ser melhorada aumentando-se a faixa dinâmica do filtro. Para tanto é preciso equalizar os valores máximos nas entradas dos multiplicadores. Isto significa nada mais nada menos do que a equalização das variáveis $x_i[k]$, i = 1 ,2. O máximo nível de sinal na entrada dos quantizadores pode ser equalizado fazendo-se

$$||F_1(z)||_p = ||F_2(z)||_p \tag{2.47}$$

onde

$$F_1(z) = \frac{(1-a)z + (\delta^2 - a + a^2)}{z^2 - 2az + a^2 + \delta^2} = F_1^{'}(z) \tag{2.48}$$

$$F_2(z) = \frac{\sigma(-\delta\ z + \delta)}{z^2 - 2az + a^2 + \delta^2} = \sigma F_2^{'}(z)$$

sendo que $F_i^{'}(z)$ se refere ao filtro com $\sigma = 1$ e p = 2 significa escalamento em norma quadrática, enquanto p = ∞ significa escalamento em norma infinita. A partir daí, pode-se perceber que

$$\sigma = \frac{\| F_1^{'}(z) \|_p}{\| F_2^{'}(z) \|_p} \tag{2.49}$$

O *overflow* pode ser eliminado multiplicando-se o sinal de entrada por um escalar λ dado por

$$\lambda = \frac{1}{||F_a(z)||_p} \tag{2.50}$$

onde $F_a(z)$ é a função de transferência da entrada da rede para o nó $x_1[n]$-$u[n]$.

Pode-se mostrar que essa estrutura não é ótima em relação ao ruído. Para que isto ocorra, ela precisaria respeitar as equações (2.20) e (2.21). Pela sua definição, porém, a matriz **A** obedece à equação (2.20), enquanto das equações (2.39), (2.45) e (2.46) pode-se escrever que

$$\frac{b_1}{b_2} = -\frac{(2+\alpha_1)}{2\sigma\delta} \tag{2.51}$$

$$\frac{c_1}{c_2} = \frac{-(\alpha_1 + 2\alpha_2)\beta_1 + (2+\alpha_1)\beta_2}{2\sigma\delta(\beta_1 + \beta_2)} \tag{2.52}$$

donde se vê que para que (2.21) seja respeitada é necessário que

$$\frac{\beta_1}{\beta_2} = \frac{\alpha_1 + 2}{\alpha_2 + 1} \tag{2.53}$$

condição esta que pode ser respeitada em casos particulares mas não de forma geral.

Uma outra informação importante é a função de transferência das variáveis de estado $x_i[n+1]$ para saída da rede. Elas são dadas, por

$$G_1(z) = \frac{c_1(z-a) + c_2\delta\,\sigma}{z^2 - 2az + a^2 + \delta^2} \tag{2.54}$$

$$G_2(z) = \frac{c_2(z-a) + c_1\dfrac{\delta}{\sigma}}{z^2 - 2az + a^2 + \delta^2} \tag{2.55}$$

Também pode-se usar a estrutura tipo I para o desenvolvimento de filtros digitais de alta ordem na forma paralela. Neste caso o seguinte procedimento deve ser usado

1) Escreva a função de transferência do filtro na forma

$$T(z) = d + \sum_{i=1}^{m} H_i^{'}(z) \tag{2.56}$$

onde cada $H_i^{'}(z)$ é dado por (1.11);

2) Compute *a* e δ para cada $H_i^{'}(z)$ usando (2.43) e (2.44);

3) Calcule σ para cada seção, de acordo com (2.49). No caso do uso de norma L_2 para escalamento, isto levará a

$$\sigma = \sqrt{\frac{(2+\alpha_1)^2[(1+\alpha_2)(1+\mu^2)-2\alpha_1\mu]}{8\delta^2(1+\alpha_1+\alpha_2)}} \qquad (2.57)$$

$$\mu = \frac{\alpha_1 + 2\alpha_2}{\alpha_2 + 2}$$

e no caso de escalamento em norma L_∞, isto levará a

$$\sigma = \frac{2+\alpha_1}{2\delta}\sqrt{\frac{1+f^2+2f\cos\omega_0}{2(1-\cos\omega_0)}} \qquad (2.58)$$

onde ω_0 é a freqüência do pólo e

$$f = \frac{\alpha_1}{2} + \frac{2\delta^2}{2+\alpha_1}\ ; \qquad (2.59)$$

4) Calcule **A** e **c** para cada seção, de acordo com (2.40), (2.45) e (2.46);

5) Calcule a constante de escalamento λ para cada seção, como em (2.50);

6) Para restabelecer o sinal de saída de cada seção faça,

$$c_i^{'} = \frac{c_i}{\lambda} \qquad (2.60)$$

No caso abordado de uma estrutura paralela de m seções do tipo I, o cálculo da densidade espectral relativa de potência do ruído (RPSD (ω)), na saída do filtro, é feito de acordo com

$$\mathrm{RPSD}(\omega) = 1 + \sum_{j=1}^{m}\left\{\sum_{i=1}^{2}|\,G_{ij}(e^{j\omega T})\,|^2 + \frac{|\,H_j(e^{j\omega T})\,|^2}{2}\right\} \qquad (2.61)$$

e a variância relativa do ruído na saída do filtro pode ser escrita como

$$\frac{\sigma_0^{\,2}}{\sigma_i^2} = 1 + \sum_{j=1}^{m} \left\{ \sum_{i=1}^{2} \| G_{ij}(z) \|_2^2 + \frac{\| H_j(z) \|_2^2}{2} \right\} \tag{2.62}$$

onde $G_{1i}(z)$ e $G_{2i}(z)$ são dadas por (2.54) e (2.55), respectivamente, para cada seção e $H_j'(z)$ é a função de transferência da entrada da rede para a saída da j-ésima seção.

2.2.2 - Estrutura do Tipo II [6]

A estrutura do tipo II é similar à do tipo I. Para ela, o vetor **P** é dado por

$$\mathbf{P} = [0 \; 1]^T$$

e, de acordo com (1.51), tem-se que

$$b_1 = -a_{12}, \quad b_2 = 1 - a_{22} \tag{2.63}$$

Como mencionado em [6], a estrutura do tipo II é muito similar à aquela do tipo I. Toda a análise feita para a estrutura tipo I pode ser repetida para a estrutura tipo II. A simetria destas duas estruturas mostra que o ruído na saída do filtro é equivalente para ambas estruturas, como pode ser verificado rapidamente.

2.2.3 - Estrutura do Tipo III [6] [26]

Para a estrutura do tipo III, o vetor **P** é dado por

$$\mathbf{P} = [1 \quad 1]^T$$

A matriz **A** é a mesma das estruturas I e II, resultando que as equações (2.40), (2.41), (2.42), (2.43) e (2.44) se mantêm. O vetor **b** pode ser obtido pela equação (1.51), ou seja

$$b_1 = (1 - a_{11}) - a_{12}, \qquad b_2 = -a_{21} + (1 - a_{22}) \tag{2.64}$$

A Figura 2.3 ilustra tal estrutura.

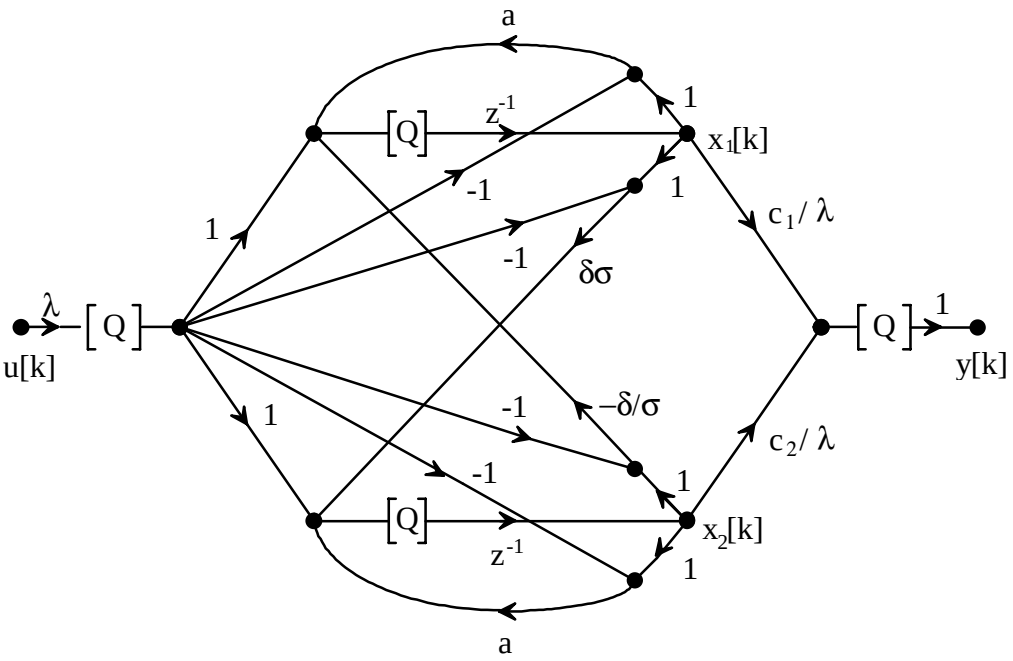

Figura 2.3 - Estrutura do tipo III

De posse da matriz **A** e do vetor **b**, é possível se obter os coeficientes do vetor **c**, de maneira idêntica àquela desenvolvida para a estrutura do tipo I, o que resulta em

$$c_1 = \frac{\beta_1(2\sigma\delta + 2\alpha_2 + \alpha_1) - \beta_2(2 + \alpha_1 - 2\sigma\delta)}{2(1 + \alpha_1 + \alpha_2)(\sigma\delta + \frac{\delta}{\sigma})} \tag{2.65}$$

$$c_2 = \frac{\beta_1(2\delta - 2\sigma\alpha_2 - \alpha_1\sigma) + \beta_2(2\sigma + \alpha_1\sigma + 2\delta)}{2\sigma(1 + \alpha_1 + \alpha_2)(\sigma\delta + \frac{\delta}{\sigma})} \tag{2.66}$$

As funções de transferência da entrada da rede para os nós $x_1[n]$-$u[n]$ e $x_2[n]$-$u[n]$ e dos nós $x_1[n + 1]$ e $x_2[n + 1]$ para a saída, respectivamente, são dadas por $F_a(z)$, $F_b(z)$, $G_1(z)$ e $G_2(z)$, e são definidas como

$$F_a(z) = \frac{z^2 + (1 - a_{12} + a_{22})z + a_{12} - a_{22}}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.67}$$

$$F_b(z) = \frac{z^2 + (1 - a_{21} + a_{11})z + a_{21} - a_{11}}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.68}$$

$$G_1(z) = \frac{c_1 z + c_2 a_{21} - c_1 a_{22}}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.69}$$

$$G_2(z) = \frac{c_2 z + c_1 a_{12} - c_2 a_{11}}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.70}$$

A constante σ é dada, de forma similar à estrutura do tipo I, pela condição

$$\| F_a(z) \|_p = \| F_b(z) \|_p \tag{2.71}$$

que resulta, no caso de p = 2, em [26]

$$\sigma = \frac{(1 - \alpha_2) \pm \sqrt{(\alpha_2 + 1)^2 - \alpha_1^{\,2}}}{2\delta}$$

A constante de escalamento λ é dada por

$$\lambda = \frac{1}{max\ \{\|F_a(z)\|_p, \|F_b(z)\|_p\}} \tag{2.72}$$

De forma idêntica à estrutura do tipo I, a estrutura do tipo III pode ser usada para o desenvolvimento de filtros digitais de alta ordem na forma paralela. Neste caso o seguinte procedimento deverá ser usado:

1) Escreva a função de transferência do filtro na forma

$$T(z) = d + \sum_{i=1}^{m} H_i'(z) \tag{2.73}$$

onde cada $H_i'(z)$ é dado por (1.12);

2) Compute *a* e δ para cada $H_i^{'}(z)$ usando (2.43) e (2.44);

3) Calcule σ para cada seção de acordo com (2.71);

4) Calcule **A** e **c** para cada seção de acordo com (2.40), (2.65) e (2.66);

5) Calcule a constante de escalamento λ_i para cada seção como em (2.72);

6) Para restabelecer o sinal de saída de cada seção faça

$$c_i^{'} = \frac{c_i}{\lambda} \tag{2.74}$$

No caso de uma estrutura paralela de *m* blocos do tipo III, a densidade espectral relativa de potência do ruído (RPSD (ω)) na saída do filtro e a variância relativa do ruído na saída do filtro podem novamente ser calculadas como em (2.61) e (2.62).

Vale ser ressaltado que o objetivo estabelecido em [6] era a busca de uma estrutura de baixa complexidade computacional. Como a estrutura tipo III requer três adições a mais que a do tipo I, ela foi ignorada. Em estudos posteriores, porém, comprovou-se que tal estrutura possui um melhor desempenho, no que tange ao ruído na saída do filtro [26].

2.3 - A Estrutura Quase Ótima [18]

A estrutura quase ótima é derivada da seção de segunda ordem ótima. Esta estrutura, assim como as do tipo I, II e III, é imune a ciclos limite com entrada zero, entrada constante e em caso de o*verflow*.

A mesma técnica de eliminação de ciclos limite no caso de entrada constante já empregada nas estruturas I, II e III é novamente usada, sendo que o vetor **P** agora é dado por

$$\mathbf{P} = [p_1 \;\; p_2]^{\mathrm{T}} \tag{2.75}$$

Como nas outras estruturas mostradas, para assegurar a não existência de ciclos limite devidos à entrada zero e a *overflow*, os quantizadores implementam truncamento em magnitude e saturação aritmética nas variáveis de estado. Juntamente com a matriz **A** que implementa a rede ótima, tal condição é suficiente.

Tal estrutura é representada na Figura 2.4. Em tal figura tem-se que

$$\mathbf{X}[n+1] = Q[\mathbf{A}\mathbf{X}[n] + (\mathbf{I} - \mathbf{A})Q[\mathbf{P}\mathrm{U}[n]]] \tag{2.76}$$

$$\mathrm{Y}[n] = \mathrm{Q}[\mathbf{c}\mathbf{X}[n] + \mathrm{dU}[n]] \tag{2.77}$$

onde os quantizadores aplicados a $p_1U[n]$ e $p_2U[n]$ são necessários para assegurar que **P**U[n] seja exatamente representado no comprimento de palavra usado [6].

A estrutura recursiva da Figura 2.4 é a mesma da Figura 2.1, a menos do escalamento.

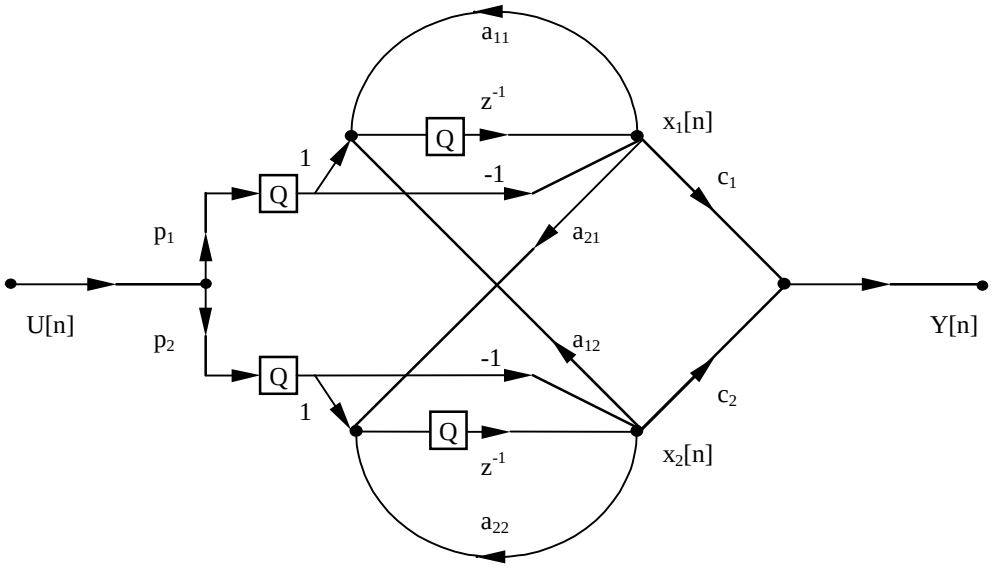

Figura 2.4 - Estrutura quase ótima genérica.

Como a estrutura de mínimo ruído é imune a ciclos limite devidos à entrada zero [12], garantidamente esta nova estrutura também será. O mesmo raciocínio é válido para ciclos limite devidos a *overflow* [20]-[21]. Da mesma forma, esta estrutura é semelhante à do tipo III, que é imune a ciclos limite de entrada constante, e logo ela também o é. Isto pode ser provado ao se verificar que a equação (2.76) corresponde a um filtro com entrada zero, onde os estados são agora redefinidos como **x**[k] - **P**U_0, onde U_0 é a entrada constante.

Uma versão mais detalhada da estrutura quase ótima [18] está na Figura 2.5. Para a sua síntese, alguns passos devem ser observados, a saber

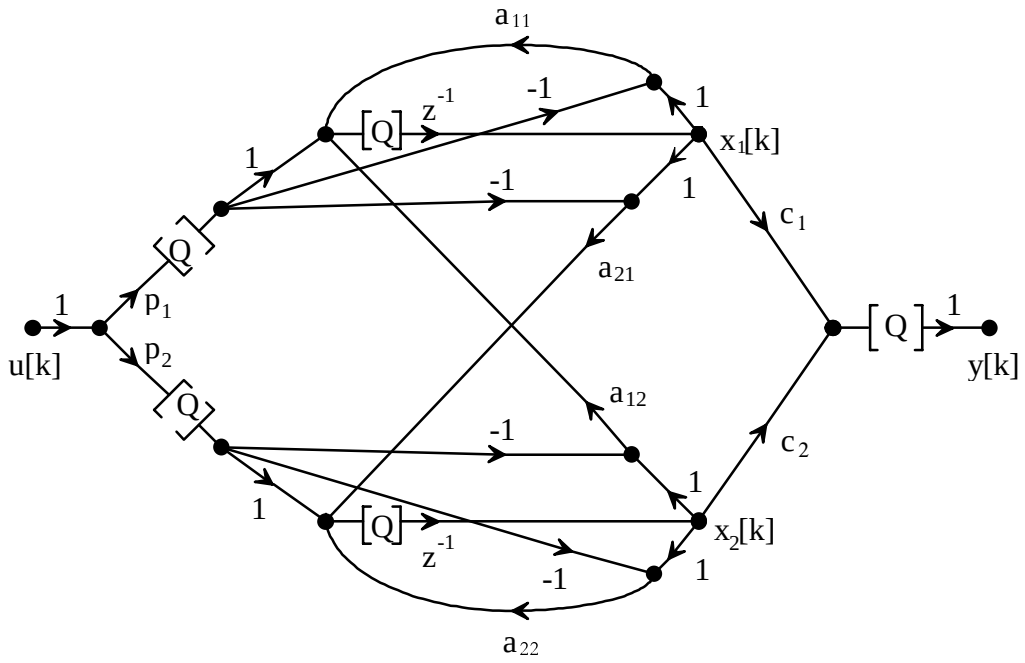

Figura 2.5 - Estrutura quase ótima.

1) sintetizar a estrutura ótima devidamente escalada;

2) calcular o vetor **P**, dado por **P** = (**I** - **A**)$^{-1}$**b**, e introduzi-lo na estrutura, como na Figura 2.5;

3) re-escalar a estrutura através da matriz **T** dada por

$$\mathbf{T} = \begin{bmatrix} \|F_a(z)\|_p & 0 \\ 0 & \|F_b(z)\|_p \end{bmatrix} \tag{2.78}$$

onde $F_a(z)$ e $F_b(z)$ são definidas, respectivamente, como as funções de transferência da entrada da rede para os nós $x_1[n] - p_1 U[n]$ e $x_2[n] - p_2 U[n]$ e são dadas por

$$F_a(z) = \frac{-p_1 z^2 + [p_1(1+a_{22}) - p_2 a_{12}]z + p_2 a_{12} - p_1 a_{22}}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.79}$$

$$F_b(z) = \frac{-p_2 z^2 + [p_2(1+a_{11}) - p_1 a_{21}]z + p_1 a_{21} - p_2 a_{11}}{z^2 - (a_{11} + a_{22})z + a_{11}a_{22} - a_{12}a_{21}} \tag{2.80}$$

No caso de $p = 2$, ou seja, quando se trabalha com escalamento em norma quadrática, a norma pode ser calculada usando-se o resultado em [19]. Quando se trabalha com escalamento em norma infinita, deve-se usar o resultado em [27] e [28].

A densidade espectral relativa do ruído na saída de um filtro de m seções quase ótimas conectadas em paralelo, e a sua variância relativa, são dadas, respectivamente, por

$$\mathrm{RPSD(w)} = 1 + \sum_{i=1}^{m} \left\{ \sum_{j=1}^{2} | G_{ji}(e^{jwT}) |^2 + | L_{ji}(e^{jwT}) |^2 \right\} \tag{2.81}$$

$$\frac{\sigma_0^{\,2}}{\sigma_i^2} = 1 + \sum_{i=1}^{m} \left\{ \sum_{j=1}^{2} || G_{ji}(z) ||_2^2 + || L_{ji}(z) ||_2^2 \right\} \tag{2.82}$$

onde $G_1(z)$ e $G_2(z)$ são as funções de transferência dos nós $x_1[n + 1]$ e $x_2[n + 1]$ para saída da rede, para uma seção, as quais são dadas por

$$G_1(z) = \frac{c_1 z + c_2 a_{21} - c_1 a_{22}}{z^2 - (a_{11} + a_{22}) z + a_{11} a_{22} - a_{12} a_{21}} \tag{2.83}$$

$$G_2(z) = \frac{c_2 z + c_1 a_{12} - c_2 a_{11}}{z^2 - (a_{11} + a_{22}) z + a_{11} a_{22} - a_{12} a_{21}} \tag{2.84}$$

enquanto $L_1(z)$ e $L_2(z)$ são as funções de transferência dos nós de saída dos multiplicadores p_1 e p_2 para a saída da rede, respectivamente, para uma seção, e são dadas por

$$L_1(z) = \frac{z(c_2 a_{21} - c_1 a_{11} + c_1(a_{11} a_{22} - a_{12} a_{21})) + c_1(1 - a_{11}) - c_2 a_{21}}{z^2 - (a_{11} + a_{22}) z + a_{11} a_{22} - a_{12} a_{21}} \tag{2.85}$$

$$L_2(z) = \frac{z(c_1 a_{12} - c_2 a_{22} + c_2(a_{11} a_{22} - a_{12} a_{21})) + c_2(1 - a_{22}) - c_1 a_{12}}{z^2 - (a_{11} + a_{22}) z + a_{11} a_{22} - a_{12} a_{21}} \tag{2.86}$$

2.4 - Análise das Estruturas Abordadas [26]

Nesta seção é feita uma análise comparativa das estruturas tipo I, tipo III e quase ótima, descritas nesta seção, abordando sua complexidade computacional e o ruído gerado na saída do filtro.

Quanto à complexidade computacional, a preocupação maior é com o número de estruturas básicas necessárias para implementar cada uma das realizações. Estas estruturas básicas são os somadores de duas entradas, os multiplicadores e os quantizadores. A filosofia aqui adotada é que quanto menor for o número de estruturas básicas, menor é o esforço computacional necessário na implementação. Será tomado por base de comparação uma implementação na forma paralela. Se for usada a estrutura da Figura 2.2, os filtros gerados são menos complexos, necessitando de 6 somadores, 7 multiplicadores e 3 quantizadores por seção, mais um quantizador na saída e mais um multiplicador d. Usando a estrutura da Figura 2.3, o filtro gerado é mais complexo, necessitando de 9 somadores por seção e o mesmo número de multiplicadores e de quantizadores do caso anterior. Por sua vez, um filtro implementado com a estrutura da Figura 2.5 é mais complexo ainda, necessitando de 9 somadores, 9 multiplicadores e 4 quantizadores por seção, além do multiplicador d. Assim, a estrutura quase ótima é a mais complexa de todas, enquanto as estruturas do tipo I e tipo II são as menos complexas.

Com relação à análise comparativa das quatro estruturas descritas neste capítulo no que tange ao ruído gerado na sua saída, a Figura 2.6 mostra um exemplo ilustrativo. Ele corresponde à implementação de um filtro na forma paralela de blocos de segunda ordem, usando como blocos construtivos cada uma das estruturas das Figuras 2.1, 2.2, 2.3 e 2.5. Trata-se de um filtro passa-baixas de ordem oito, com espectro elíptico, freqüência de amostragem de 10kHz, banda passante com largura variando de 100 a 1500 Hz, e banda de transição com largura de vinte por cento da banda passante.

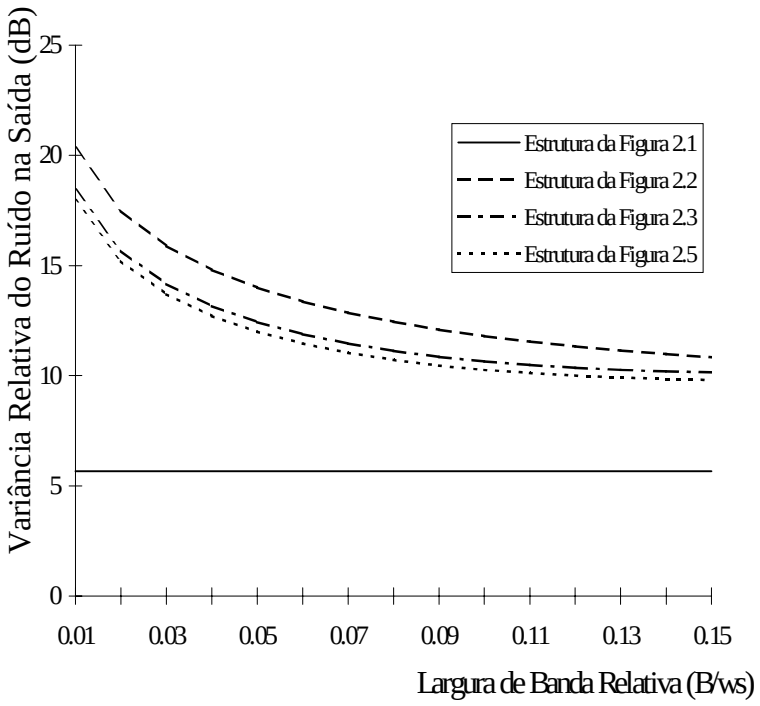

Figura 2.6 - Desempenho das estruturas quanto ao ruído na saída do filtro.

Como pode ser visto a partir da Figura 2.6, a estrutura ótima é a que apresenta o menor ruído, é imune ciclos limite de entrada zero e em casos de *overflow,* mas não é imune a ciclos limite devidos a entrada constante. Das estruturas imunes a todos os tipos de ciclos limite a que apresenta menor ruído é a quase ótima. Na seqüência, vem a estrutura do tipo III, e, depois, as demais. Por sua vez a estrutura do tipo III, embora ligeiramente mais ruidosa que a estrutura quase ótima, representa uma solução de compromisso interessante, por ter ruído e complexidade computacional relativamente reduzidos. Por esta razão, tal estrutura será abordada de forma mais profunda nos próximos capítulos, com o objetivo de discutir uma forma mais eficiente de usar o parâmetro σ que aparece na equação (2.37), com vistas a obter uma versão menos ruidosa da mesma, que substitua aquela originalmente proposta em [6].

CAPÍTULO III

Síntese Ótima da Estrutura Tipo III

Como já demostrado no Capítulo 2, a estrutura do tipo III original apresenta menor ruído na saída que as estruturas do tipo I ou II [26]. Uma das razões para isto é justamente a simetria dos dois caminhos da entrada da rede para os nós correspondentes às variáveis de estado, que é uma importante característica para a redução de ruído na saída do filtro [1] [3]. Este melhor desempenho a nível de ruído é uma característica fundamental, fazendo com que a estrutura tipo III se torne, então, atrativa.

Assim é que, neste capítulo, a estrutura tipo III proposta em [6] é rediscutida. Seu desempenho a nível do ruído na saída do filtro é avaliado minuciosamente, através de um estudo detalhado da variância relativa do ruído na saída da rede em função do parâmetro livre σ, e é mostrado que existe um valor de σ para o qual é mínima a variância relativa do ruído na saída (para esta classe de estruturas). Estratégias de síntese são também propostas, objetivando o mínimo ruído para esta estrutura do tipo III. No final deste capítulo, também, é feita uma análise de desempenho da nova rede sintetizada (com σ modificado).

Para desenhar os vários gráficos apresentados neste capítulo foram usados algoritmos implementados em MATLAB. Os programas podem ser encontrados no final desta dissertação em um anexo.

3.1 - Análise do Desempenho da Estrutura do Tipo III

A estrutura do tipo III é caracterizada pela matriz

$$\mathbf{A} = \begin{bmatrix} a & -\dfrac{\delta}{\sigma} \\ \delta\sigma & a \end{bmatrix} \tag{3.1}$$

onde o parâmetro σ é usado para reduzir o ruído na saída do filtro, uma vez que os pólos estarão corretamente alocados, independentemente de seu valor. Quando foram sintetizadas as estruturas do tipo I e do tipo II, σ foi usado de forma que as normas das funções de transferência auxiliares $F_1(z)$ e $F_2(z)$ (da entrada da rede para as variáveis de estado) fossem equalizadas. Esta é uma condição para mínimo ruído na saída [1] [3]. Aqui, é importante notar que para a estrutura do tipo I, as normas de $F_1(z)$ e $F_2(z)$ serão muito diferentes, se σ não for

devidamente calculado, devido à assimetria dos caminhos da entrada da rede até as variáveis de estado. No caso da estrutura do tipo III, porém, isso não acontece, uma vez que tais caminhos não são tão assimétricos. Pode-se usar σ, então, de maneira a minimizar o ruído na saída da rede através da outra abordagem.

Para analisar o desempenho a nível de ruído da estrutura do tipo III, um primeiro gráfico é apresentado na Figura 3.1, que mostra a variância relativa do ruído na saída do filtro em função do valor de σ. O gráfico corresponde a um filtro passa-baixas Chebyschev, com banda passante de 100 Hz, banda de rejeição começando em 120 Hz, máximo *ripple* na banda passante de 0,1 dB e freqüência de amostragem de 10 kHz. No mesmo gráfico, a variância relativa do ruído na saída correspondente à estrutura do tipo I é apontada (símbolo +), assim como a variância relativa do ruído na saída correspondente à estrutura do tipo III, quando a estratégia de equalizar as normas de $F_1(z)$ e $F_2(z)$ é usada (símbolo x). Em todos os casos o escalamento usado é L_2.

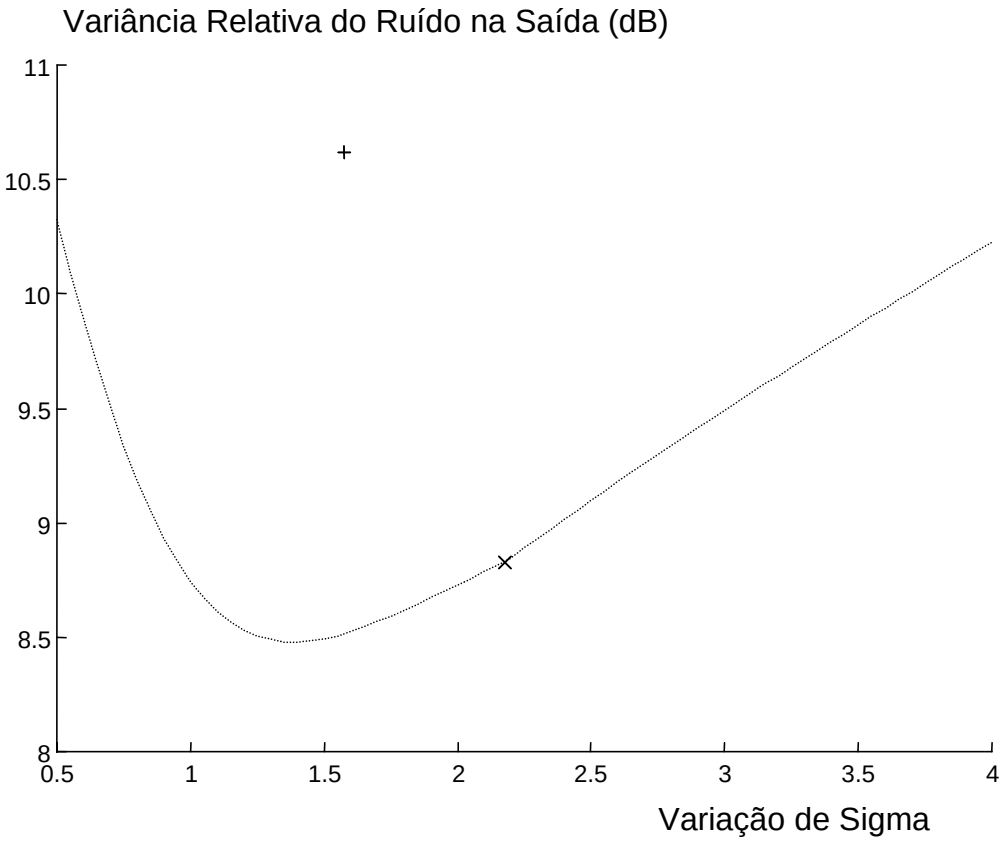

Figura 3.1 - Variância relativa do ruído em função de σ para a estrutura do tipo III, escalada em norma quadrática.

Tal figura mostra dois resultados importantes: o primeiro é que, de fato, a variância relativa do ruído na saída do filtro é menor para a estrutura do tipo III do que para a estrutura do tipo I, o que por si só já justificaria o uso desta ao invés da estrutura do tipo I. O segundo resultado é que o valor da variância que corresponde à estrutura do tipo III com $\|F_1(z)\|_2 = \|F_2(z)\|_2$ não é um mínimo. Este resultado é muito interessante, uma vez que mostra que é possível mudar o procedimento de síntese da estrutura do tipo III, de forma a obter um melhor valor de σ, agora correspondente à mínima variância relativa do ruído na saída.

3.2 - Novas Estratégias de Síntese

A fim de completar a síntese da estrutura do tipo III, ao mesmo tempo que se busca reduzir o ruído na saída do filtro, o cálculo de σ é aqui discutido, tanto usando o escalamento L_2 como o escalamento L_∞.

3.2.1 - Escalamento L_2

Os gráficos apresentados nas Figuras 3.2, 3.3 e 3.4, mostram, para três exemplos (Butterworth, Chebyschev e Elíptico, respectivamente) com as mesmas especificações do filtro da Figura 3.1, como se comportam a variância relativa do ruído, as normas quadráticas de $F_a(z)$ e $F_b(z)$, e as normas quadráticas de $G_1(z)$ e $G_2(z)$, quando se varia σ. Nestes gráficos, o ponto onde se verifica

$$\|F_a(z)\|_2 = \|F_b(z)\|_2 \tag{3.2}$$

correspondente à estratégia usada em [6] para sintetizar a estrutura do tipo III, é o cruzamento curvas que representam as normas de $F_a(z)$ e $F_b(z)$. Para este valor de σ, pode-se checar nas Figuras 3.2, 3.3 e 3.4 que a variância relativa do ruído na saída do filtro não é mínima.

Os gráficos das Figuras 3.2, 3.3 e 3.4, por sua vez, também mostram a variação das normas quadráticas das funções intermediárias $G_1(z)$ e $G_2(z)$. O valor de σ correspondente à condição

$$\|G_1(z)\|_2 = \|G_2(z)\|_2 \tag{3.3}$$

que é o cruzamento das curvas que representam as normas de $G_1(z)$ e $G_2(z)$, corresponde a uma menor variância relativa do ruído na saída do filtro. Note-se que (3.3) é também uma condição para mínimo ruído para um filtro digital de segunda ordem no espaço de estados, escalado com L_2 [1] [3]. Então, a condição em (3.3) é uma estratégia mais adequada para a síntese da estrutura do tipo III. Entretanto, é importante mencionar, como mostram os exemplos, que para filtros elípticos os resultados são equivalentes, quando se usa (3.2) ou (3.3).

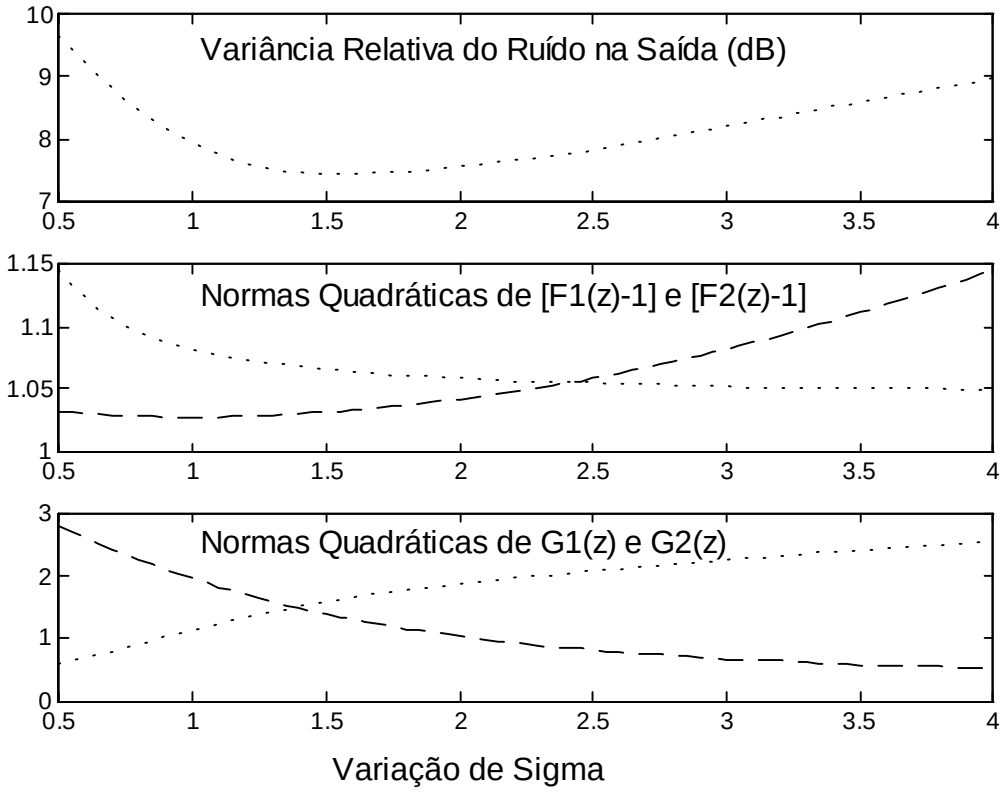

Figura 3.2 - Comportamento de um filtro Butterworth escalado em norma quadrática com a variação de sigma.

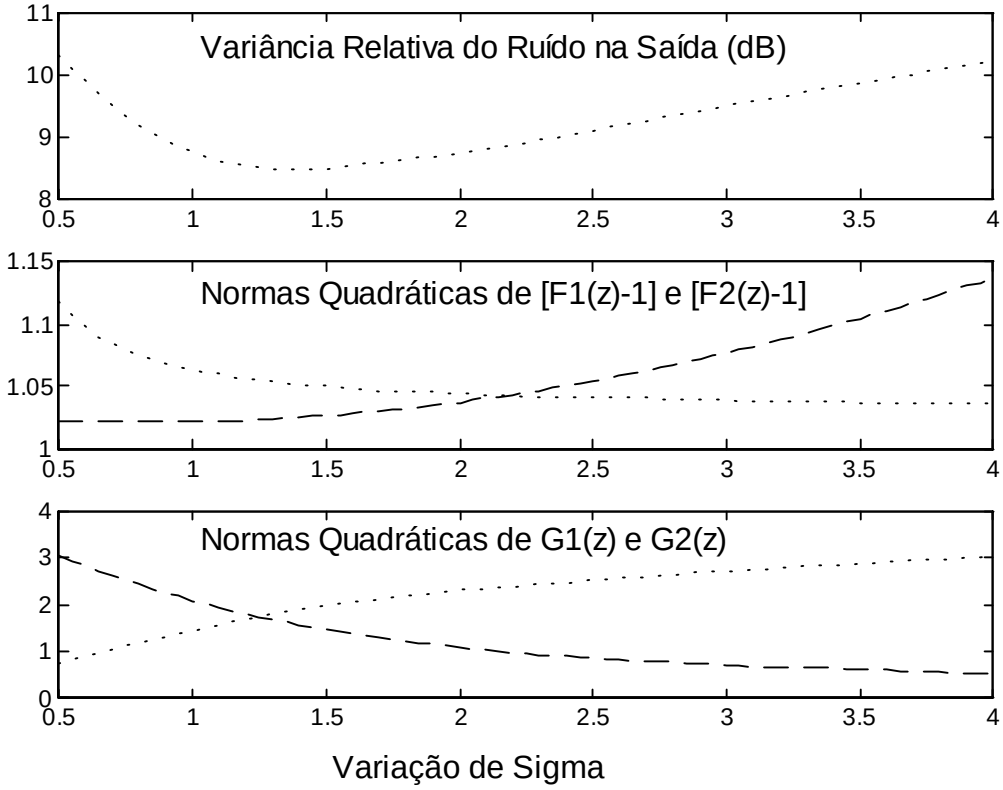

Figura 3.3 - Comportamento de um filtro Chebychev escalado em norma quadrática com a variação de sigma.

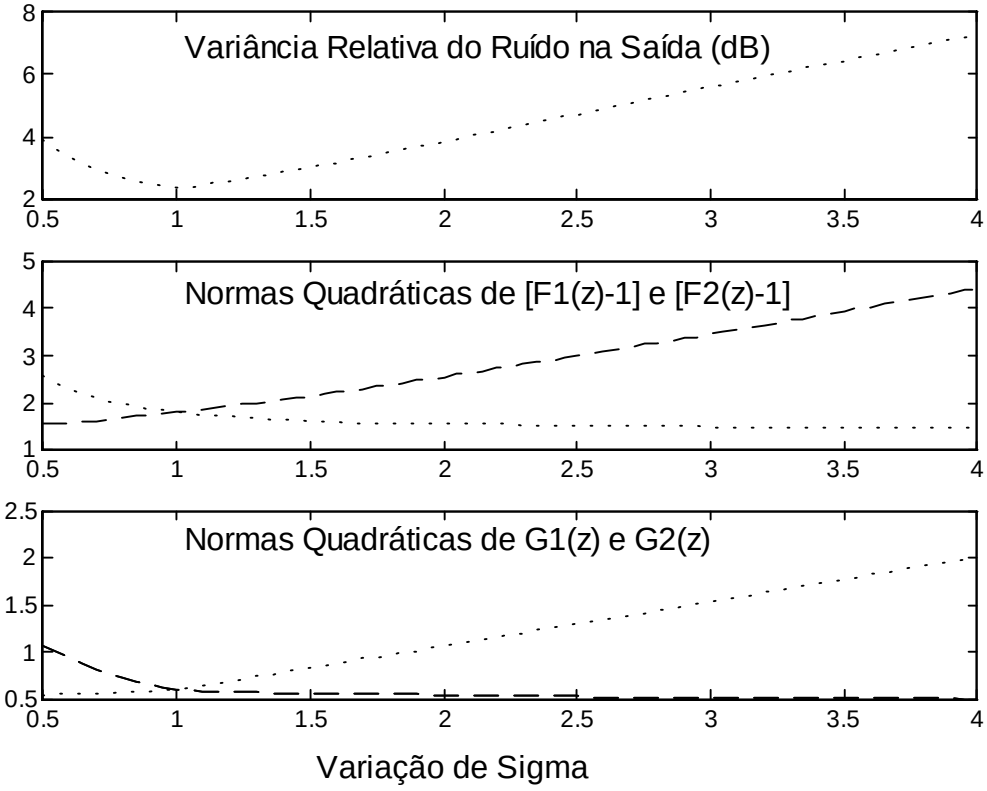

Figura 3.4 - Comportamento de um filtro Elíptico escalado em norma quadrática com a variação de sigma.

3.2.2 - Escalamento L_∞

Uma análise similar é feita, usando as Figuras 3.5, 3.6 e 3.7, para os mesmos exemplos das Figuras 3.2, 3.3 e 3.4, considerando agora o escalamento L_∞. Neste caso pode-se notar, analisando-se os gráficos mostrados, que a melhor estratégia de síntese é impor a condição

$$\|F_a(z)\|_\infty = \|F_b(z)\|_\infty \tag{3.4}$$

ao invés de (3.3). Este resultado não é surpreendente, pois no caso de escalamento L_∞ a condição (3.3) não é mais uma condição para se obter mínimo ruído, como no caso de escalamento L_2 [1] [3] [5].

Então, quando se desejar sintetizar estruturas do tipo III escaladas em L_2, a melhor estratégia para calcular σ é dada por (3.3), enquanto que quando se desejar sintetizar estruturas do tipo III escaladas em L_∞ a estratégia em (3.4) é a mais indicada.

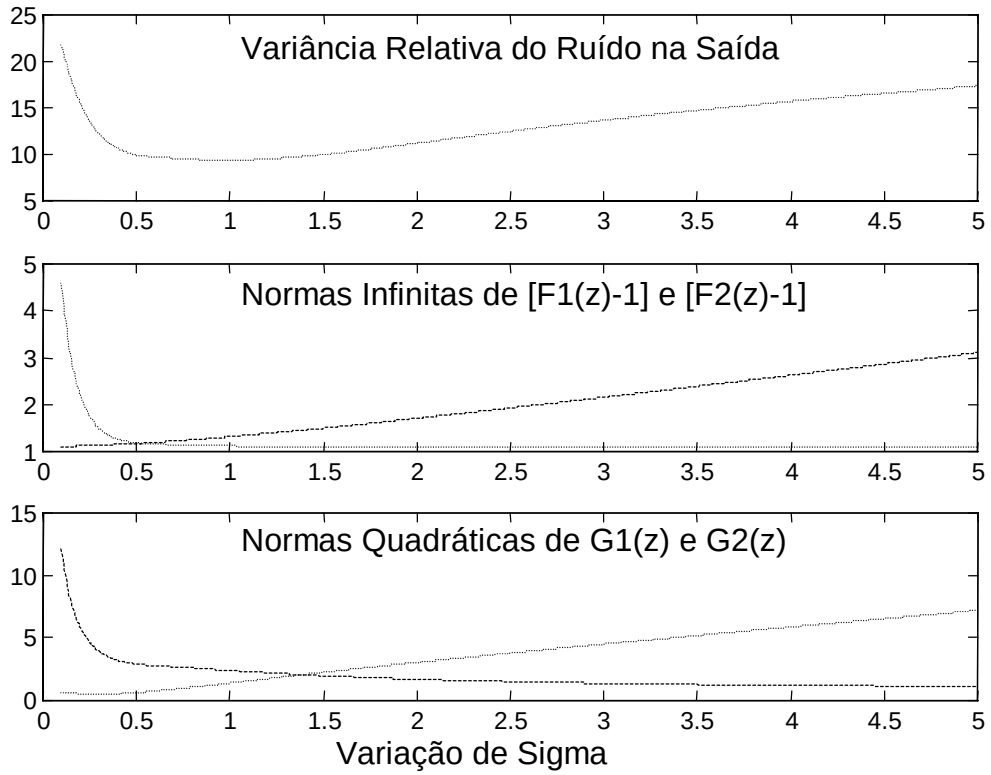

Figura 3.5 - Comportamento de um filtro Butterworth escalado em norma infinita com a variação de sigma.

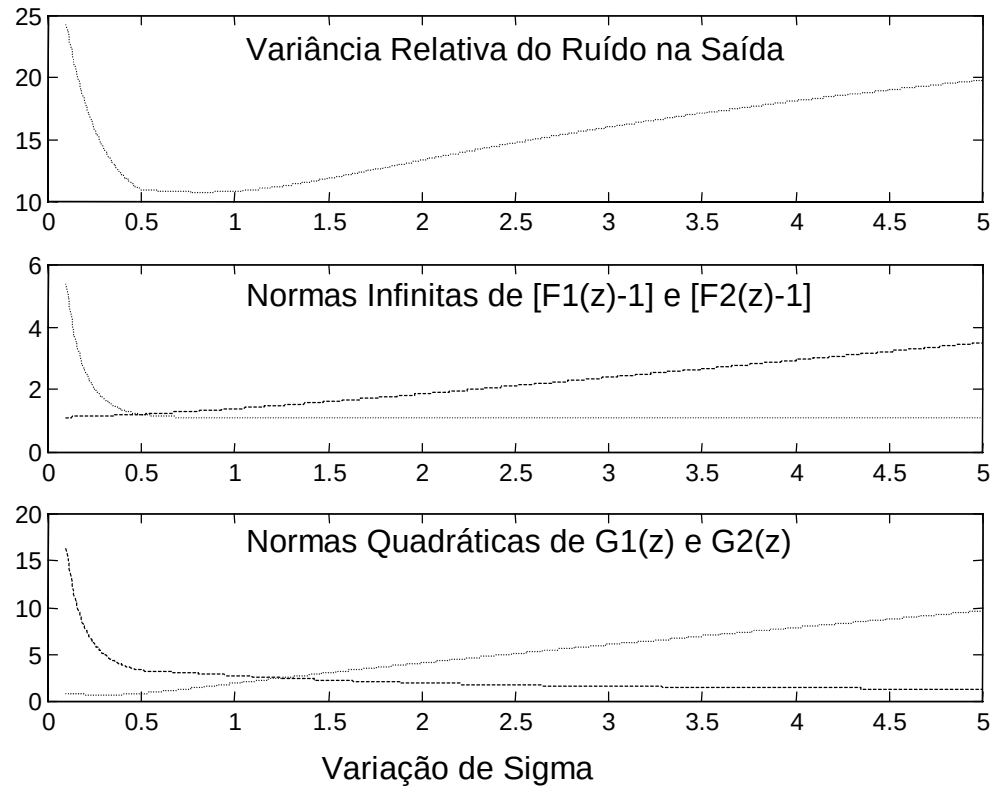

Figura 3.6 - Comportamento de um filtro Chebyschev escalado em norma infinita com a variação de sigma.

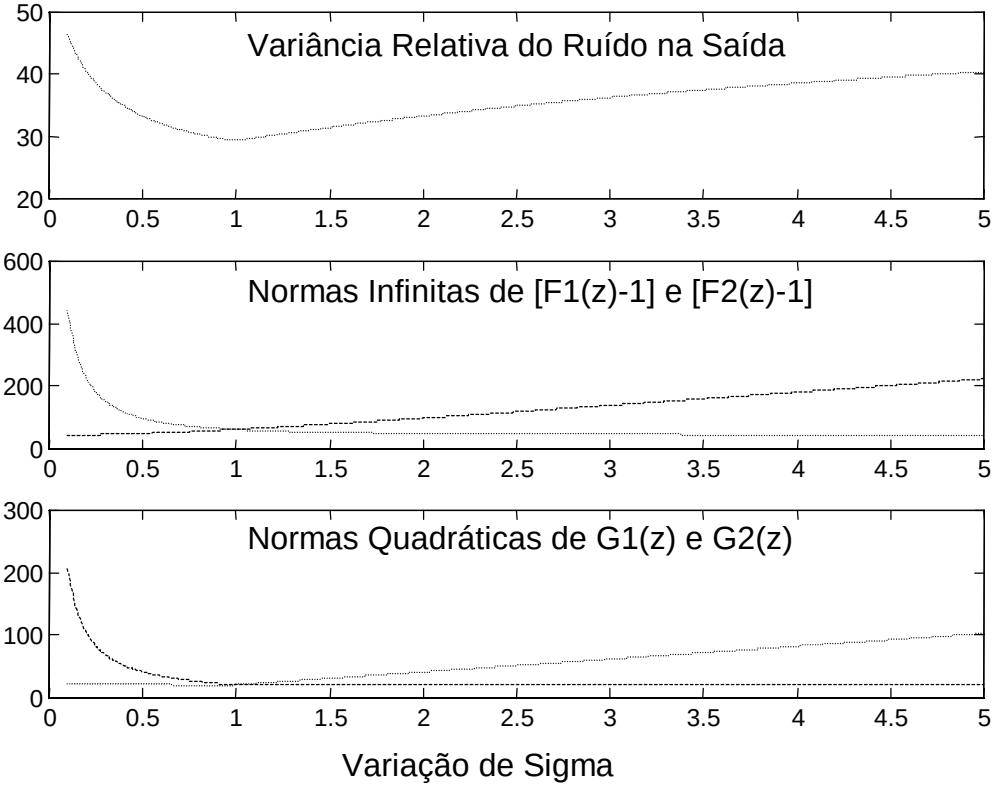

Figura 3.7 - Comportamento de um filtro Elíptico escalado em norma infinita com a variação de sigma.

3.3 - Análise de Desempenho da Nova Rede

Para analisar o desempenho da nova rede, foi feito um estudo comparativo para alguns exemplos de filtros de um bloco (Chebyschev, Buterworth e Elíptico), variando-se a banda passante dos filtros através de transformações espectrais [2]. Para todos os exemplos são usados filtros com mesma especificação: passa-baixas, largura de banda de 100 Hz, banda de rejeição em 120 Hz e *ripple* na banda passante de 0.5 dB.

Para cada transformação espectral o algoritmo varia σ e calcula o ponto de mínimo ruído, o ponto onde $\|F_1(z)\|_p = \|F_2(z)\|_p$ e o ponto onde $\|G_1(z)\|_2 = \|G_2(z)\|_2$. Os resultados para escalamento L_2 são mostrados nas Figuras 3.8, 3.9 e 3.10, onde a curva pontilhada corresponde à mínima variância do ruído na saída, a curva contínua representa a estrutura do tipo III originalmente proposta em [6] e a curva tracejada corresponde à estrutura do tipo III quando sintetizada usando à condição dada em (3.3).

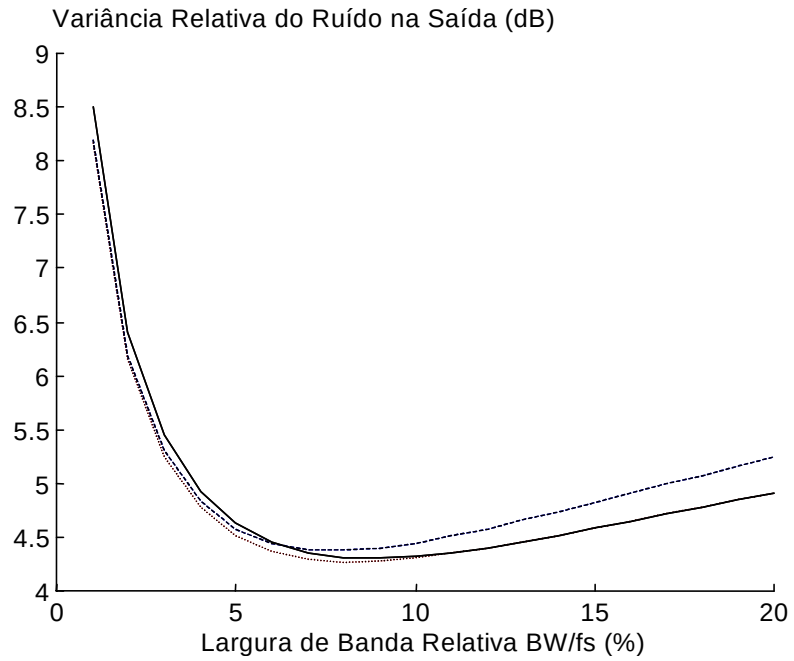

Figura 3.8 - Desempenho de ruído de um filtro Butterworth tipo III escalado em L_2.

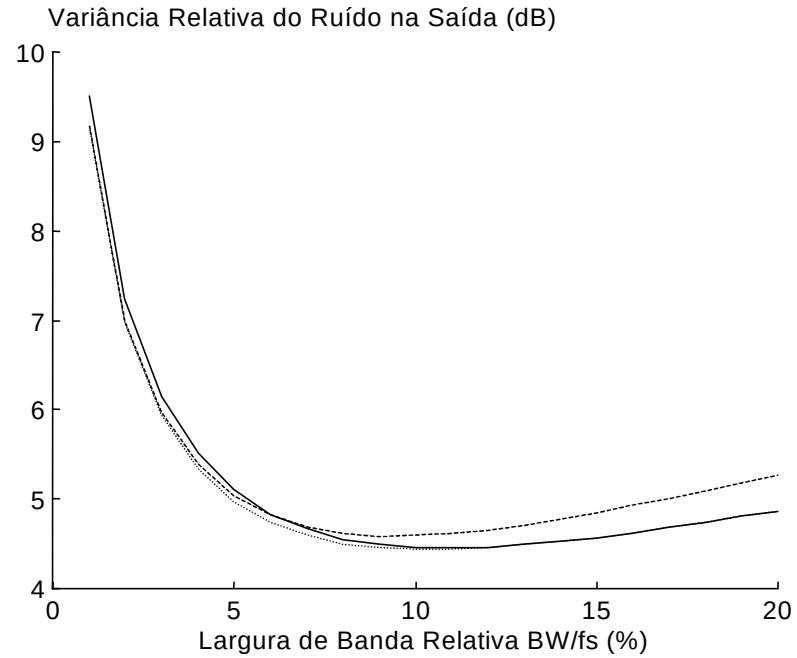

Figura 3.9 - Desempenho de ruído de um filtro Chebyshev tipo III escalado em L_2.

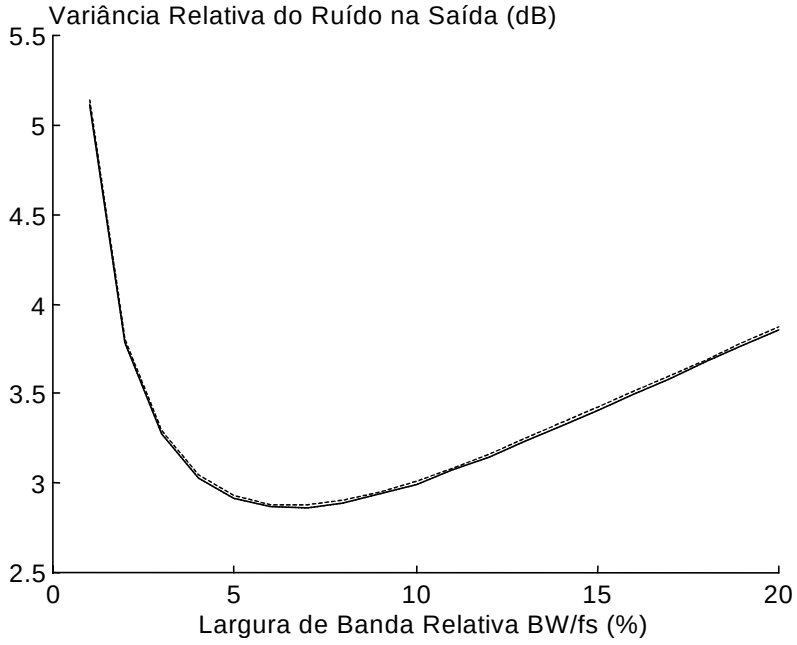

Figura 3.10 - Desempenho de ruído de um filtro Elíptico tipo III escalado em L_2.

Já as Figuras 3.11, 3.12 e 3.13 mostram gráficos similares, considerando os mesmos exemplos, agora com escalamento L_∞. Nos dois casos de escalamento, verifica-se que as estratégias propostas são extremamente adequadas, principalmente nos casos de filtros de banda passante muito estreita.

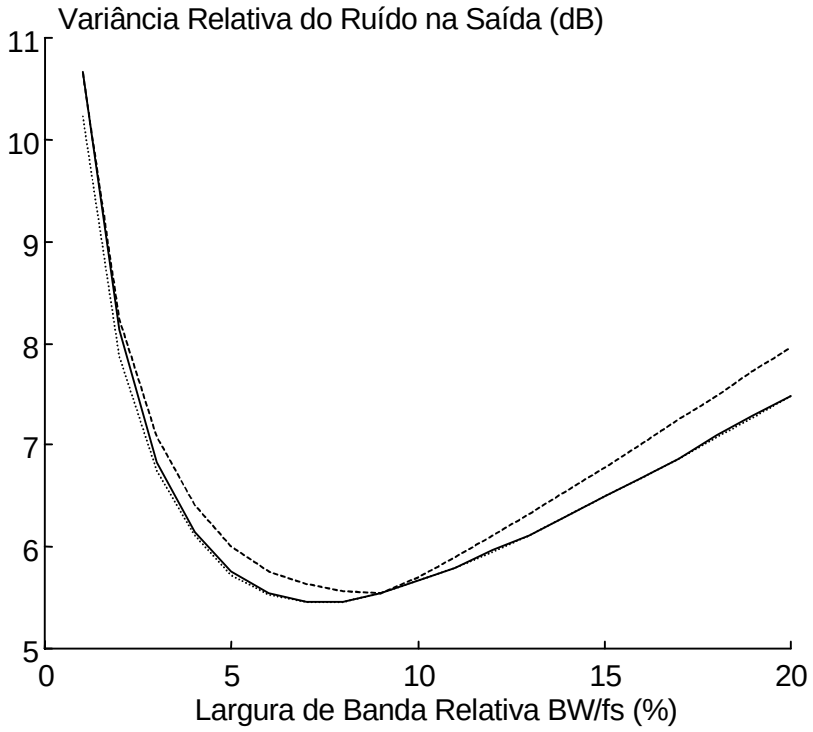

Figura 3.11 - Desempenho de ruído de um filtro Butterworth tipo III escalado em L_∞.

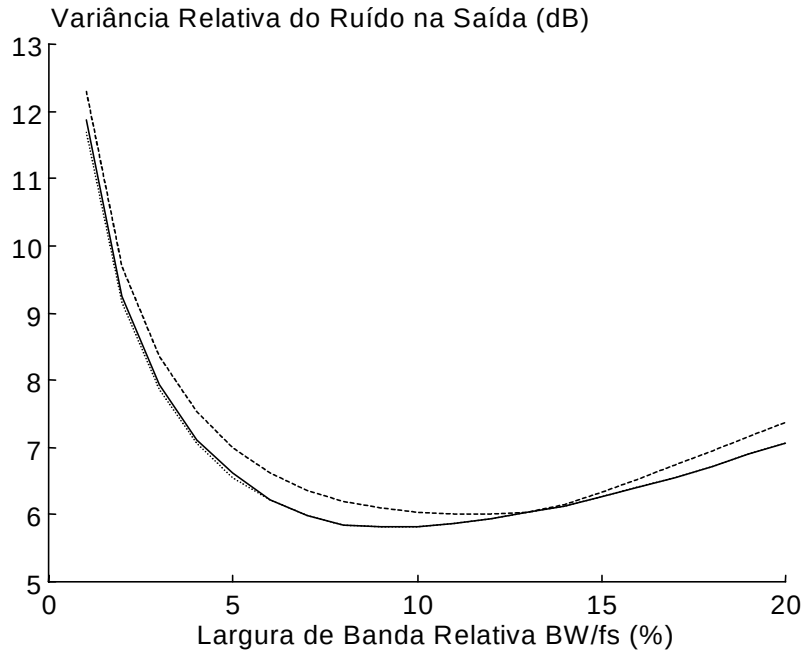

Figura 3.12 - Desempenho de ruído de um filtro Chebyshev tipo III escalado em L_∞.

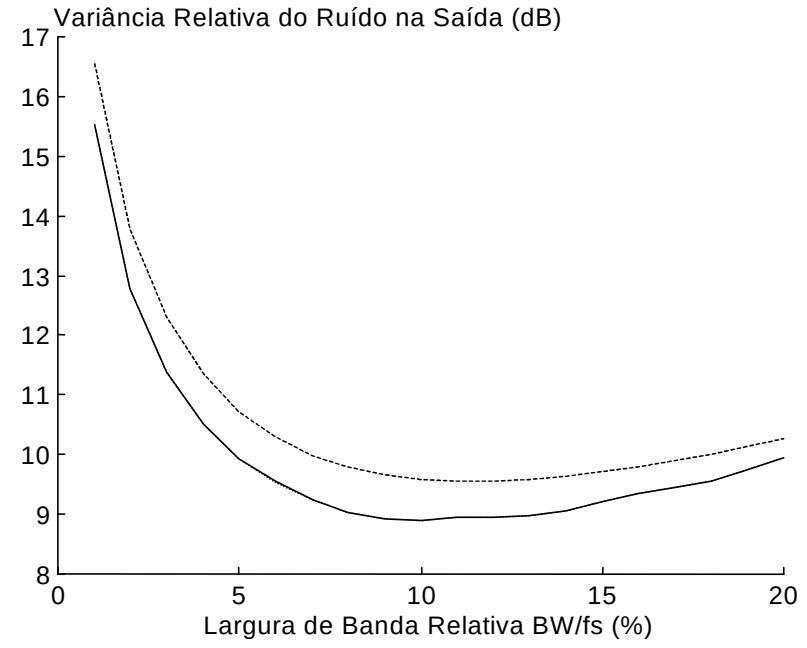

Figura 3.13 - Desempenho de ruído de um filtro Elíptico tipo III escalado em L_∞.

3.4 - Conclusão

Neste capítulo, a estrutura do tipo III originalmente proposta em [6] foi revisitada. Uma estratégia mais adequada para usar um parâmetro livre para reduzir a variância relativa do ruído na saída do filtro foi também discutida, e o resultado é que a estrutura do tipo III pode apresentar mínima variância relativa do ruído quando tal parâmetro é propriamente selecionado. Novas estratégias para escalamento em norma infinita e quadrática foram propostas, definindo-se através delas valores adequados do referido parâmetro, em cada caso de escalamento.

A nova rede aqui desenvolvida tem menor variância relativa do ruído na saída, em comparação com as redes do tipo I, II e III, originalmente propostas em [6], o que a torna muito interessante para aplicações que precisam de redes de baixo ruído e imunes a ciclos limite.

No próximo capítulo será feita uma comparação da nova estrutura aqui gerada com outros filtros já consagrados na literatura.

CAPÍTULO IV

Comparação com Outros Filtros

No capítulo anterior, uma síntese ótima da estrutura tipo III foi proposta, nos casos de escalamento em norma quadrática e em norma infinita. O que se comprovou é que é possível melhorar o desempenho da estrutura tipo III [6], trabalhando-se a variável σ.

Neste capítulo, um estudo comparativo do desempenho da nova estrutura tipo III é realizado, para comprovar-se as vantagens de sua utilização, assim como em que situações ela é recomendável. Tal estudo compara a estrutura tipo III ótima (tipo III onde se obtém o mínimo ruído) escalada em norma quadrática e norma infinita, com a estrutura tipo III original, a estrutura de mínimo ruído, e a estrutura *Error Spectrum Shaping* ótima [29]. No capítulo anterior somente filtros de um único bloco de segunda ordem foram estudados. Neste capítulo, adicionalmente, exemplos de filtros paralelos compostos por vários blocos de segunda ordem também são discutidos.

Para desenhar os vários gráficos apresentados neste capítulo foram usados algoritmos implementados em MATLAB. Os programas podem ser encontrados no final desta dissertação, em um anexo.

4.1 - Variação da Banda Passante

Para análise do comportamento do filtro, em relação à variância relativa de ruído na saída, são usadas várias larguras de banda passante. Faz-se necessário, então, variar a largura de banda passante do filtro analisado. Este estudo consiste em começar a análise com um filtro protótipo, e a partir deste, fazer transformações espectrais [2] [9] para se chegar ao próximo valor da largura de banda, e assim por diante, até se varrer toda faixa de interesse de pesquisa.

Os filtros protótipos usados nos exemplos deste capítulo são filtros passa-baixas, de largura de banda de 100 Hz, banda de rejeição iniciando-se em 120 Hz e com *ripple* na banda passante de 0.5 dB, podendo variar apenas, de um exemplo para o outro, o número de blocos de segunda ordem usados para implementação do filtro. Tais protótipos têm sua banda passante variada de 100 Hz até 2 KHz, em passos de 100 Hz, quando for o caso.

4.2 - Comparação com a Estrutura Tipo III Original [6]

No capítulo anterior, foi feita uma análise do desempenho das estruturas tipo III original e modificada, e observou-se que a nova estrutura é muito interessante para implementação de filtros digitais de banda estreita imunes a ciclos limite. Toda análise foi feita para filtros implementados com um único bloco de segunda ordem. Na prática, os filtros usuais têm mais que um bloco. Nesta seção alguns exemplos com vários blocos de segunda ordem são desenvolvidos, e seu desempenho a nível de ruído é analisado.

Os gráficos das Figuras 4.1, 4.2 e 4.3 mostram o desempenho de filtros Butterworth, Chebyschev e Elíptico, respectivamente, escalados em L_2. Em tais figuras, a linha pontilhada representa a estrutura tipo III em sua síntese ótima, como discutido no Capítulo 3, e a linha contínua representa a estrutura tipo III quando sintetizada usando a condição (3.2).

Analisando tais figuras, pode-se se notar que para filtros de banda estreita, assim como ocorria em um único bloco de segunda ordem, a síntese da estrutura tipo III proposta no Capítulo 3 efetivamente conduz a filtros com menos ruído, para escalamento em norma quadrática.

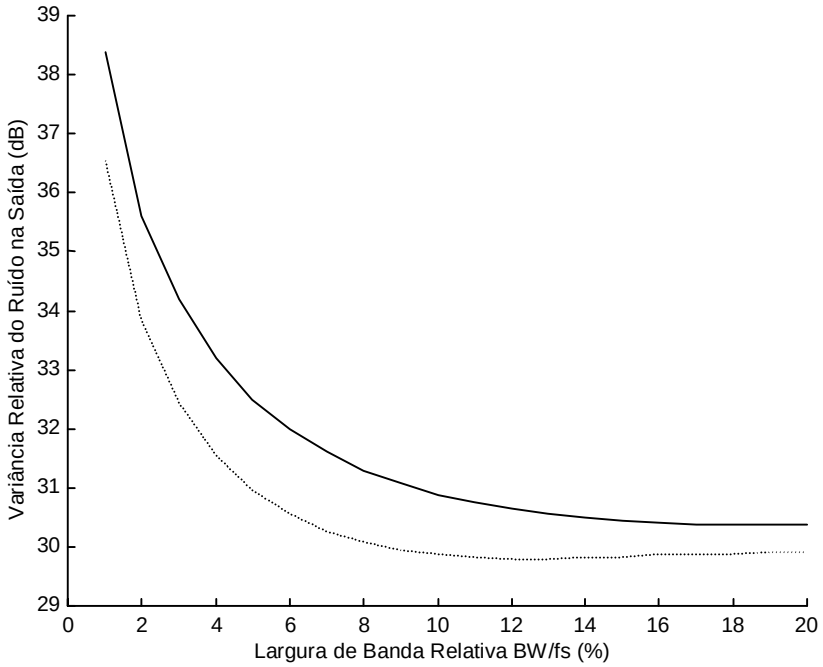

Figura 4.1: Exemplo Butterworth de ordem 10, com escalamento L_2.

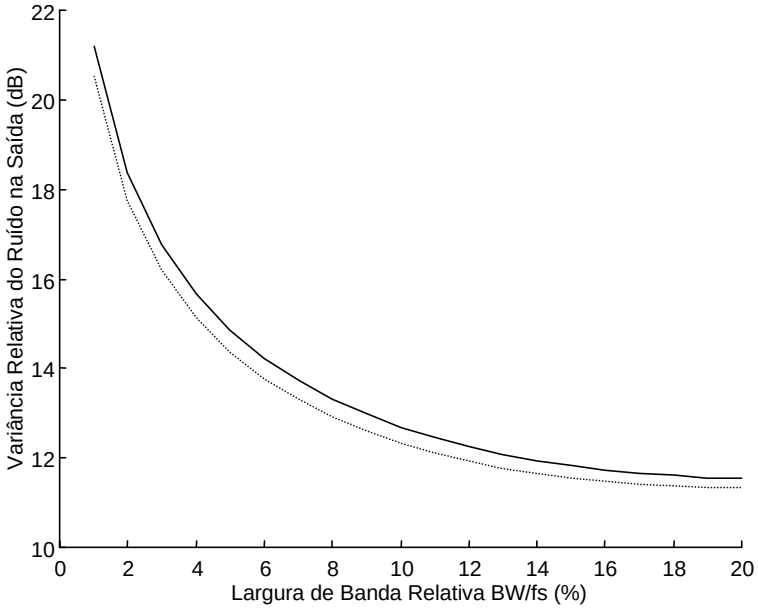

Figura 4.2 - Exemplo Chebyschev de ordem 10, com escalamento L_2.

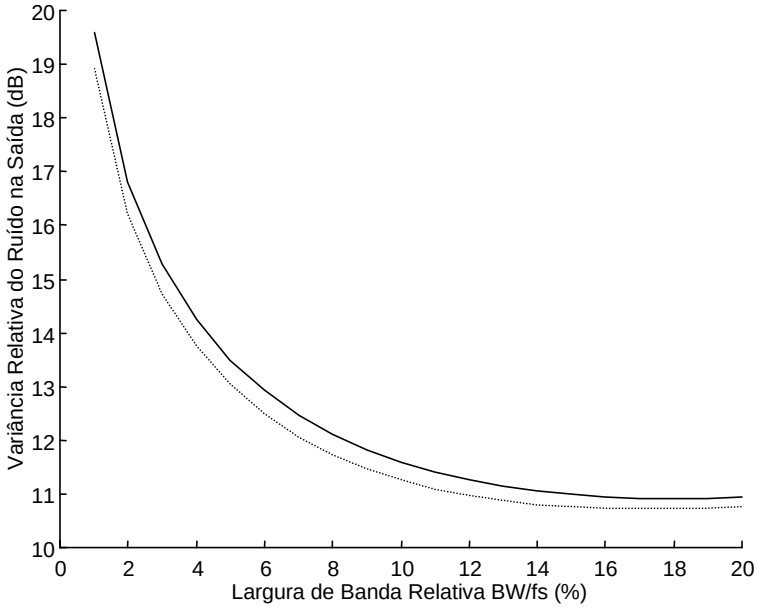

Figura 4.3 - Exemplo Elíptico de ordem 10, com escalamento L_2.

Os gráficos das Figuras 4.4, 4.5 e 4.6, respectivamente, mostram o desempenho dos filtros Butterworth, Chebyschev e Elíptico usados para gerar os gráficos das Figuras 4.1, 4.2 e 4.3, agora escalados em L_∞. Em tais figuras, novamente, a curva em linha cheia representa a estrutura tipo III sintetizada usando-se a condição (3.2), e a linha pontilhada representa a estrutura tipo III em sua síntese ótima. Uma observação de tais figuras permite notar que a

redução no ruído não é significativa, principalmente nos casos de filtros Chebyschev e elíptico, uma vez que a condição de otimalidade de ruído já é cumprida quando da síntese original da estrutura tipo III, conforme mostrado no Capítulo 3.

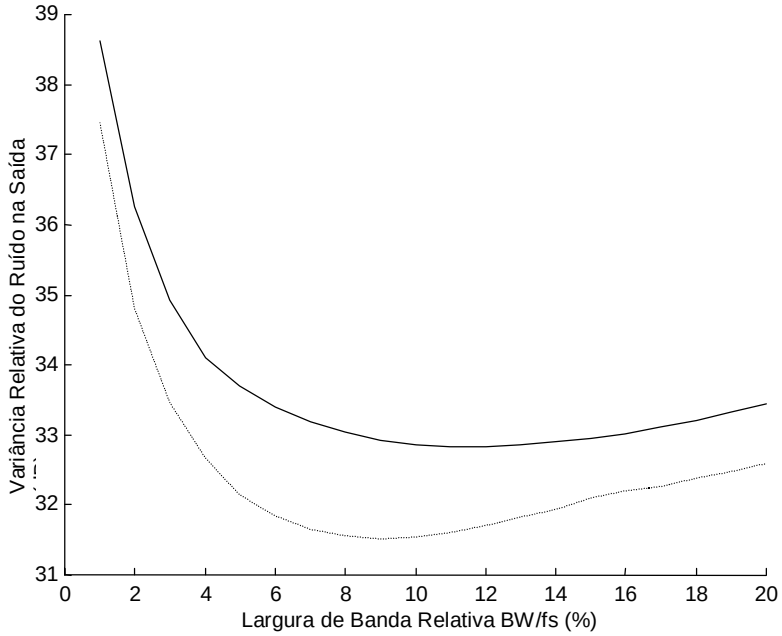

Figura 4.4 - Exemplo Butterworth de ordem 10, com escalamento L_∞.

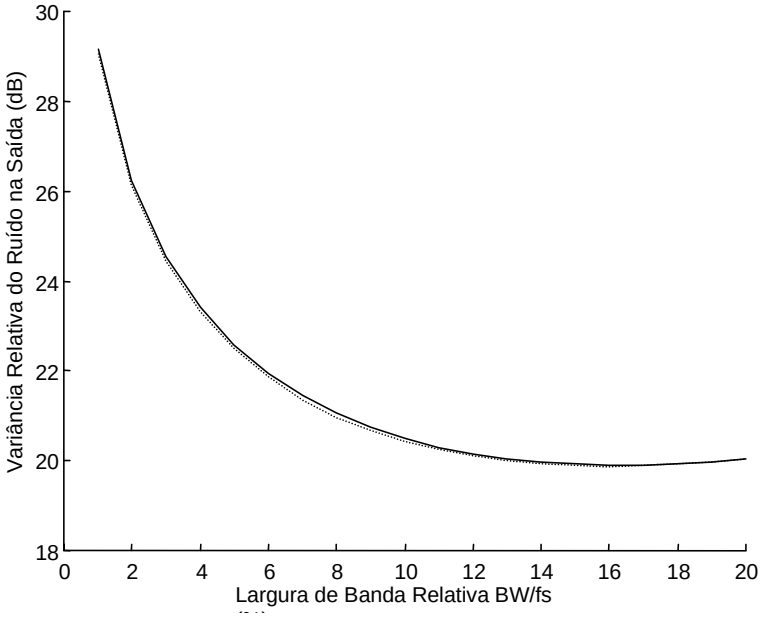

Figura 4.5 - Exemplo Chebyschev de ordem 10, com escalamento L_∞

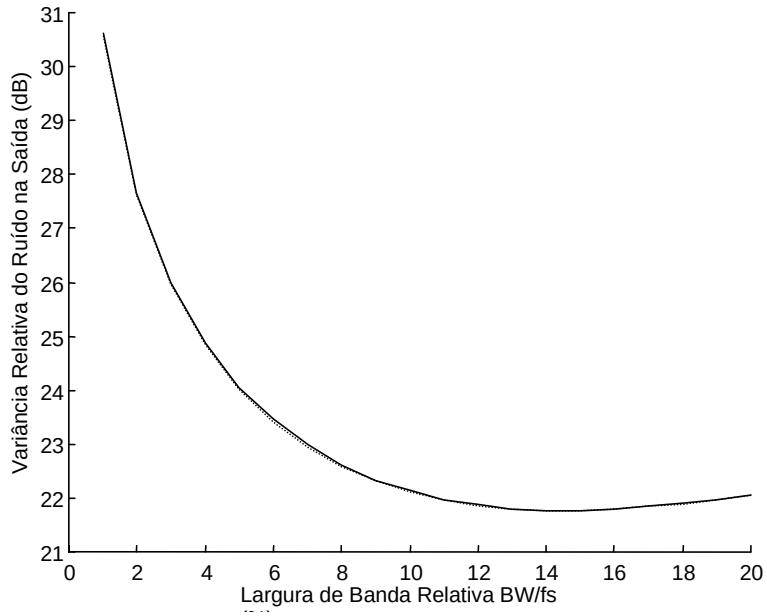

Figura 4.6 - Exemplo Elíptico de ordem 10, com escalamento L_∞

4.3 - Comparação com a Estrutura Ótima

Na seqüência, os gráficos das Figuras 4.7, 4.8 e 4.9 ilustram o desempenho das estruturas tipo III, original e modificada, em comparação com a rede de mínimo ruído [3], com escalamento L_2, para filtros Butterworth, Chebyschev e Elíptico, respectivamente, no caso de ordem 2. Por sua vez, os gráficos das Figuras 4.10, 4.11 e 4.12 ilustram o desempenho

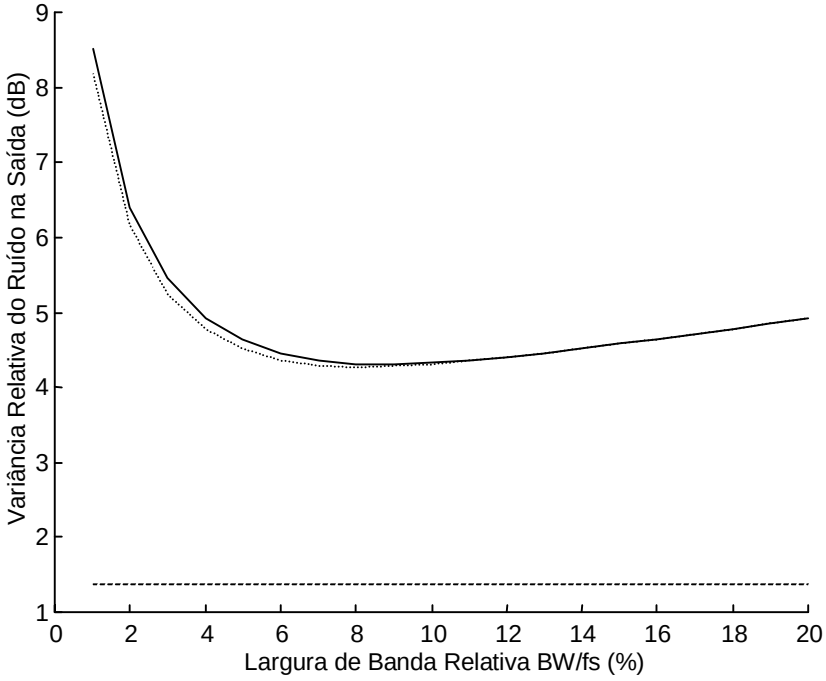

Figura 4.7 - Exemplo Butterworth, com um bloco, escalado em L_2.

das mesmas redes, agora com escalamento L_∞, para os mesmos exemplos. Em todos os casos, a linha pontilhada representa a estrutura tipo III modificada, a linha contínua representa a estrutura tipo III sintetizada usando a condição (3.2), e a linha tracejada representa a estrutura de mínimo ruído [3].

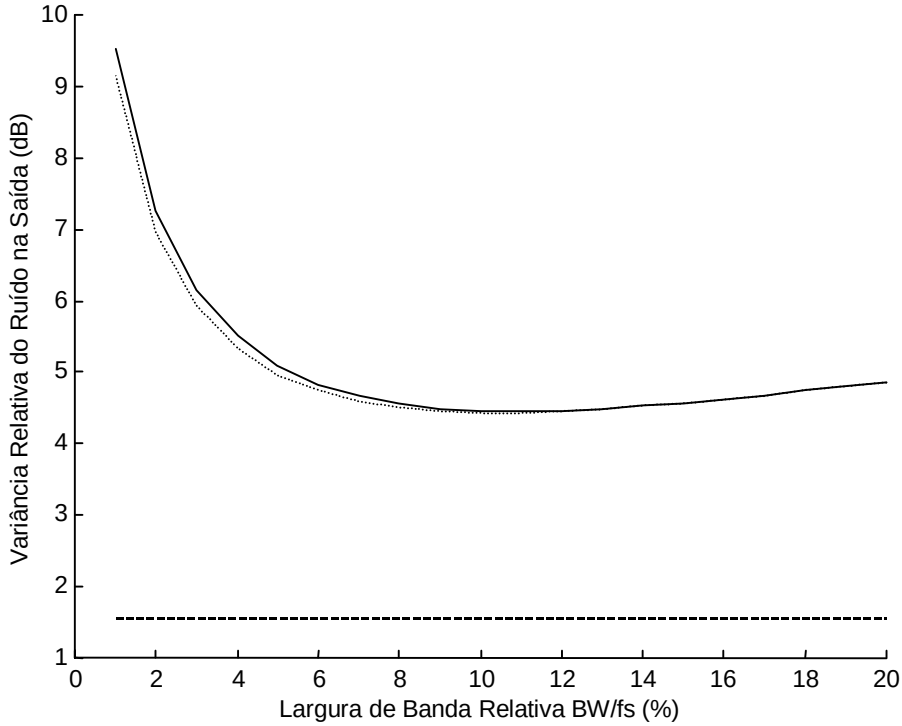

Figura 4.8 - Exemplo Chebyschev, com um bloco, escalado em L_2.

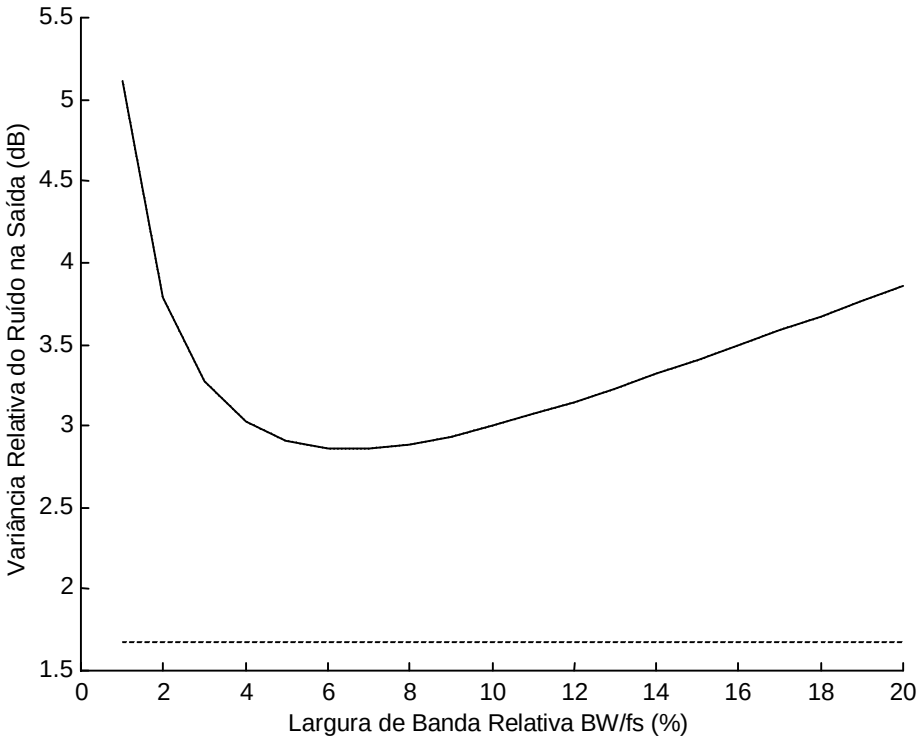

Figura 4.9 - Exemplo Elíptico, com um bloco, escalado em L_2.

Como se pode ver a partir das Figuras 4.7, 4.8 e 4.9 no caso de escalamento L_2 a variância relativa do ruído da rede ótima é bem menor que a das estruturas do tipo III, tanto na versão original quanto na modificada. Porém, é importante frisar que esta estrutura não é imune a ciclos limite devidos a entrada constante, que é justamente o caso em que a estrutura do tipo III se torna vantajosa, considerando o ruído adicional que ela introduz.

Outro detalhe interessante é que todo o estudo em [1] e [3] acerca da síntese da estrutura no espaço de estados de mínimo ruído se baseia em escalamento L_2, incluindo a demonstração da invariância do ruído com a variação da largura de banda [2]. No caso de escalamento L_∞, a estrutura gerada não mais apresenta mínimo ruído e invariância do ruído com a variação da largura de banda [1] [3] e [5], como pode ser comprovado pelas curvas das Figuras 4.10, 4.11 e 4.12. Neste caso como já foi mostrado no capítulo anterior, a estrutura tipo III ótima representa uma boa opção para a síntese dos filtros. Adicionalmente, também é possível notar que a síntese original da rede tipo III já conduz ao mínimo ruído, ou seja, a condição (3.2) já representa a síntese ótima da estrutura tipo III com escalamento L_∞, conforme foi discutido no Capítulo 3. Também vale à pena mencionar que as curvas em linha tracejada foram obtidas considerando os resultados em [5], o que faz com que o ruído na saída

do filtro seja bem próximo do mínimo. Ainda assim, para filtros de banda muito estreita, a rede tipo III tem melhor desempenho.

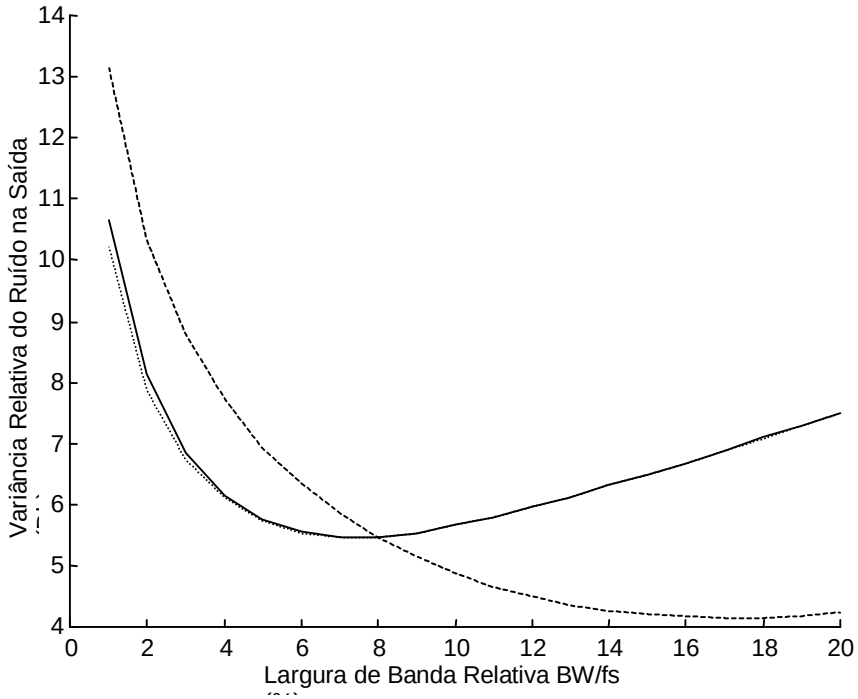

Figura 4.10 - Exemplo Butterworth, com um bloco, escalado em L_∞.

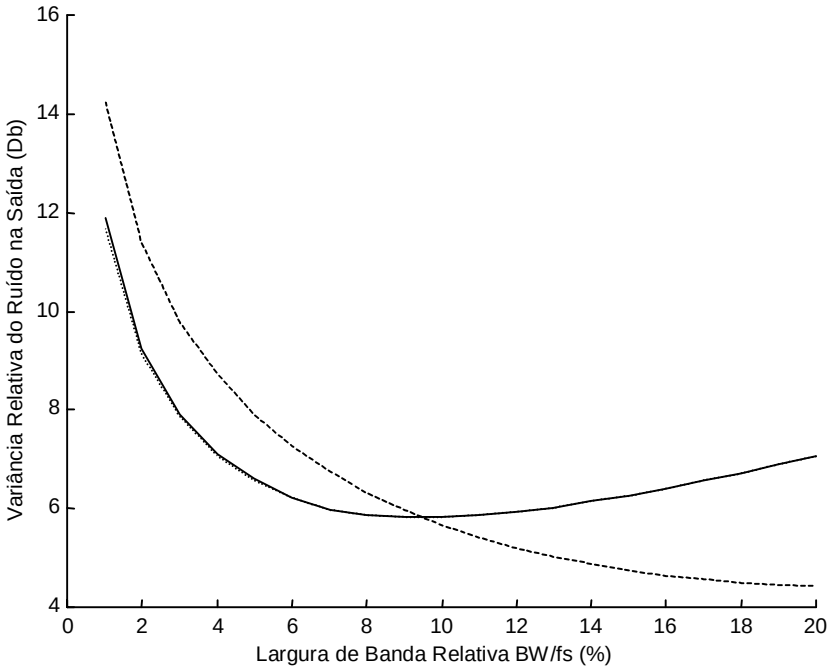

Figura 4.11 - Exemplo Chebyschev, com um bloco, escalado em L_∞.

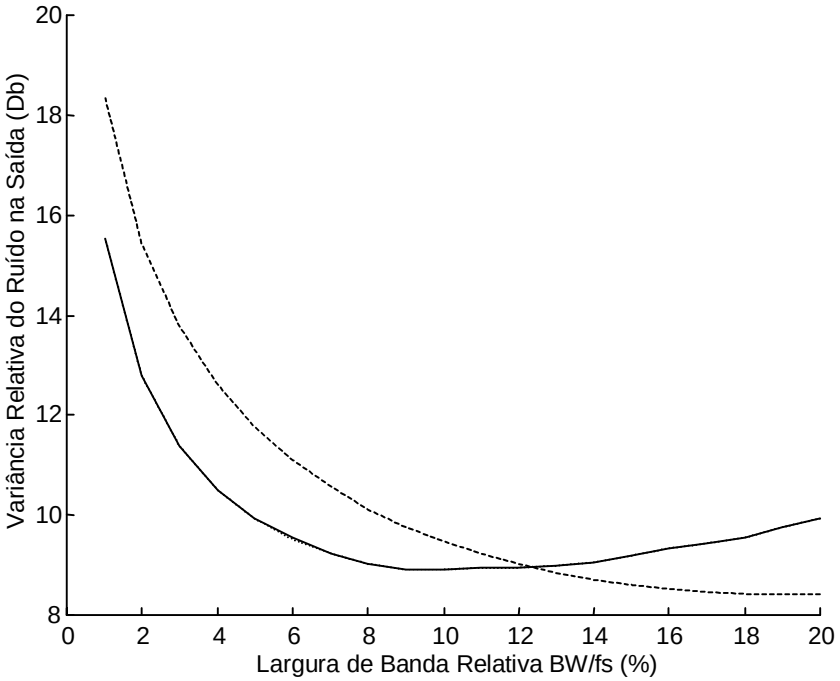

Figura 4.12 - Exemplo Elíptico, com um bloco, escalado em L_∞.

4.4 - Comparação com a Estrutura *"Error Spectrum Shaping"* [29]

Uma aproximação alternativa para a redução do ruído na saída do filtro é a aplicação de uma técnica conhecida como *error spectrum shaping* [29]. Esta técnica envolve a realimentação do erro de quantização através de uma sub-rede, de forma que tal erro é utilizado para modificar o sinal antes da quantização. Esta sub-rede de realimentação do erro possui uma versão ótima, a qual equivale a processar o sinal de entrada em precisão dupla [29] [30].

Entretanto, esta rede nada mais é do que uma versão modificada da rede direta, com o ruído bastante reduzido, porém muito susceptível à ocorrência de ciclos limite, no caso de filtros de banda estreita (pólos localizados muito próximo ao eixo real e com raio próximo de um) [9]. Para garantir a eliminação de ciclos limite em redes *error spectrum shaping*, uma solução bastante eficiente é apresentada em [30]. Ela consiste em adaptar os coeficientes da sub-rede de realimentação do erro de forma a que oscilações eventualmente presentes na saída venham a ser eliminadas. O preço para tal eliminação de ciclos limite, mais uma vez, é a deterioração da relação sinal ruído.

Sob esta ótica, redes garantidamente imunes a ciclos limite, como é o caso da rede tipo III discutida neste trabalho, tornam-se competitivas. Uma comparação dos desempenhos da rede tipo III modificada, aqui proposta, da rede *error spectrum shaping* ótima [29] e da rede *error spectrum shaping* adaptativa proposta em [30] é então mostrada na Tabela 4.1 abaixo. A figura de mérito utilizada é a relação sinal ruído, para o caso de comprimento de palavra de 16 bits. Para a rede *error spectrum shaping* adaptativa, dois casos são considerados, correspondentes às situações em que os dois coeficientes usados na sub-rede de realimentação do erro variam na mesma direção (caso A) e na direção oposta (caso B), admitindo-se 40% de variação em cada parâmetro. Para todos os casos, são usados os mesmos filtros de segunda ordem, escalados com L_2, que foram usados na seção 4.3.

Tabela 4.1 - Comparação da estrutura tipo III ótima com a rede ESS.

Exemplo	**Rede ESS Ótima**	**Rede ESS Adaptativa**		**Rede Tipo III Ótima**
		Caso A	Caso B	
Butterworth	96.3296	81.0496	72.6715	75.1184
Chebyschev	96.3296	76.5028	67.6659	72.9806
Elíptico	96.3296	70.8141	61.4806	73.1251

Como se pode observar a partir da Tabela 4.1, mesmo considerando o caso da rede *error spectrum shaping*, em sua versão imune a ciclos limite, a rede tipo III ótima ainda apresenta desempenho adequado, com uma complexidade computacional comparável à rede *error spectrum shaping*, uma vez que ambas possuem o mesmo número de multiplicadores.

CAPÍTULO V

Conclusão

Muitos trabalhos, têm sugerido estruturas para a implementação de filtros digitais que gerem baixo ruído na saída do filtro. Especialmente para o caso em que a banda passante é muito estreita (onde os pólos estão próximos do eixo real e com raios próximo da unidade), os filtros digitais de segunda ordem no espaço de estados têm se mostrado uma boa solução.

Entretanto, alguns destes filtros não são garantidamente imunes a ciclos limite devidos a entrada constante, embora o sejam para entrada zero e *overflow*. Assim, alguns esforços em pesquisa foram destinados à busca de filtros com bom desempenho em relação ao ruído e que também fossem imunes a ciclos limite no caso de entrada constante.

Um conjunto de estruturas denominadas estrutura tipo I, estrutura tipo II, e estrutura tipo III foram propostas como uma solução de compromisso entre a baixa variância relativa do ruído na saída e a imunidade a ciclos limite [6]. Como a estrutura do tipo III era computacionalmente mais complexa, sua síntese e variância do ruído não foram exploradas. Mais tarde demonstrou-se que exatamente a estrutura do tipo III é a que tem melhor desempenho em relação ao ruído [26]. Esta característica da estrutura do tipo III se dá devido aos caminhos simétricos que o sinal de entrada percorre, até as variáveis de estado.

Originalmente, a variável σ foi usada, no caso da estrutura do tipo III, para equalizar as normas das funções de transferência da entrada da rede para os nós das variáveis de estado a $x_1[n]$ e $x_2[n]$ (isto é, para fazer $\|F_1\|_2 = \|F_1\|_2$), o que é um dos princípios para minimização do ruído na saída do filtro escalado em norma quadrática. O que se mostra neste trabalho é que devido a simetria original da rede tipo III, as normas destas funções tendem a ser muito próximas, e neste caso pode-se usar σ como uma variável livre para minimizar o ruído. Através da variação de σ, foi possível a síntese de uma nova estrutura, aqui referenciada como estrutura do tipo III ótima.

A estrutura do tipo III ótima foi comparada com várias outras estruturas. O primeiro estudo compara-a com a estrutura do tipo III original, tanto para escalamento L_2 como para escalamento L_∞. O que se pode notar é que para filtros Butterworth e Chebyschev, para ambos escalamentos, a rede tipo III ótima tem um desempenho melhor. Já para o caso de filtros Elípticos, nota-se que a rede tipo III original já é muito próxima da ótima. Em um segundo estudo a nova rede é comparada com a rede ótima, consagrada na literatura. Pode-se observar

que para escalamento L_2 a rede tipo III ótima apresenta maior variância relativa de ruído na saída, mas este é o preço que se paga pela eliminação de ciclos limite no caso de entrada constante. Note-se que a rede ótima não é imune a ciclos limite devidos a entrada constante. Já para o caso de norma infinita, a rede em [3] não é ótima, como já foi comentado, e para filtros com largura de banda estreita a estrutura do tipo III ótima tem um desempenho melhor. Para finalizar, a nova rede é comparada com redes *error spectrum shaping*, e também para os casos de filtros de banda passante muito estreita, que é exatamente o propósito deste estudo, a rede tipo III ótima se mostra um boa solução.

Como resultado final, tem-se um bloco de segunda ordem no espaço de estados garantidamente livre de ciclos limite em qualquer caso, sem complexidade computacional adicional, e com baixo ruído na saída do filtro, principalmente no caso de síntese de filtros de banda estreita.

Como uma sugestão para futuros trabalhos pode-se determinar uma expressão analítica para determinar σ, e assim sintetizar a rede tipo III ótima. Neste trabalho, tal determinação foi feita sempre de forma numérica, mas, como foi mostrado, a variância relativa do ruído na saída é uma função convexa de σ, e assim pode-se estudar de forma mais detalhada tal função e determinar analiticamente o valor de tal parâmetro. Uma outra perspectiva seria a prova formal de que o ruído na saída da estrutura do tipo III ótima é de fato mínimo, para tal classe de estrutura, isto é, que existe uma estrutura imune a ciclos limite devidos a entrada zero, entrada constante e *overflow*, e que ao mesmo tempo apresenta ruído mínimo.

Referências Bibliográficas

[1] Mullis, C. T. e Roberts, R. A. (1976). "Syntesis of Minimm Roundoff Noise Fixed Point Digital Filters", IEEE Trans. on Circuits and Systems, vol. CAS-23: 551-562.

[2] Mullis, C. T. e Roberts, R. A. (1976). "Roundoff Noise in Digital Filters: Frequency Transformaations and Invariants", IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-24: 538-550.

[3] Jackson, L. B., Lindgren, A. e Kim, Y. (1979). "Optimal Syntesis of Second-Order State-Space Structures for Digital Filters", IEEE Trans. on Circuits and Systems, vol. CAS-26: 149-153

[4] Barnes, C. W. (1985). "A Parametric Approach to the Realization of Second-Order Digital Filters Sections", IEEE Trans. on Circuits and Systems, vol. CAS-32: 530-539.

[5] Bomar, B. W. (1989). "On The Design of Second Order State-Space Digital Filter Sections", IEEE Trans. on Circuits and Systems, vol. CAS-36: 542-552

.

[6] Diniz, P. S. R. e Antoniou, A. (1986). "More Economical State-Space Digital-Filter Structures Which Are Free of Costant-Input Limit Cycles", IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-34: 807-815.

[7] Jackson, L. B. (1988). "Signals and Transforms", addison Wesley Publishing Company, USA.

[8] Papoulis, A. (1994). "Probability, Random Variables and Stochastics Processes", McGray-Hill, USA.

[9] Antoniou, A. (1993). "Digital Filters: Analysis, Design and Applications", McGraw Hill, USA.

[10] Langinmaa, A. (1987). "Limit Cycles in Digital Filters", Dissertação de Mestrado, Helsinki University of Tecnhnology, Finlândia.

[11] Kawamata, M. e Higuchi, T. (1983). "A Systematic Approach to Synthesis of Limit Limit Cycle-Free Digital Filters". IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-31: 212-214.

[12] Vaidyanathan, P. P. e Liu, V. (1987). "An Improved Sufficient Condition for Absence of Cycles in Digital Filters", IEEE Trans. on Circuits and Systems, vol. CAS-34: 319-322.

[13] Mills, W. L., Mullis, C. T. e Roberts, R. A. (1978). "Digital Filter Realizations Without Overflow Oscillations", IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-26: 334-338.

[14] Jackson, L. B. (1987). "Digital Filters and Signal Processing", Kluwer Academic Publishers, USA.

[15] Oppenheim, A. V. e Schafer, R. W. (1989). "Discrete-Time Signal Processing", Prentice-Hall, USA.

[16] Jackson, L. B. (1970). "Roundoff-Noise Analysis for Fixed Point Digital Filters Realized in Cascade or Parallel Form", IEEE Trans. on Audio an Electro-Acoustics, vol. AU-18: 107-122.

[17] Bomar, B. W. (1985). "New Second-Order State-Space Structures for Realizing Low Roundoff Noise Digital Filters", IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-33: 106-110.

[18] Sarcinelli Filho, M., Cruz, C. P. da, Simmer, A. C. S. e Diniz P. S. R. (1992). "Estrutura Digital de Segunda Ordem Quase Ótima Livre de Ciclos Limite", Anais do 9° CBA, vol. 1:40-44.

[19] Mitra, S. K., Hirano, K. e Sakaguchi. (1974). "A Simple Method of Computing the Imput Quantization and Multiplication Roundoff Erros in a Digital Filter", IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-22: 326-329.

[20] Claasen, T.A.C.M., Mecklenbrauker, W.F. e Peek, J.B. (1975). "On the Stability of the Forced Response of Digital Filters with Overflow Nonlinearities", IEEE Trans. on Circuits and Systems, vol. CAS-22: 692-696.

[21] Singh, V. (1985). "A New Realizability Condition for Limit Cycle-Free State-Space Digital Filters Employing Saturation Arithmetic", IEEE Trans. on Circuits and Systens, vol. CAS-32: 1070-1071.

[22] K. M. Anspach, B. W. Bomar, R. C. Engels e R. D. Joseph, (1996). "Minimization of Fixed-Point Roundoff Noise in Extended State-Space Digital Filters", IEEE Transactions on Circuits and Systems, Part II: Analog an Digital Signal Processing, vol. 43: 193-206.

[23] Phil Lapsley, Jeff Bier, Amit Shoham e Edward A. Lee, (1997). "DSP Processor Fundamentals", IEEE Press, USA.

[24] T. Laakso e L. B. Jackson (1994). “Bounds for floating-point roundoff noise”, IEEE Transactions on Circuits and Systems - Part II: Analog and Digital Signal Processing, vol. 41: 424–426.

[25] Chimin Tsai (1997). "Floating-Point Roundoff Noises of First- and Second-Order Sections in Parallel Form Digital Filters", IEEE Transactions on Circuits and Systems - Part II: Analog and Digital Signal Processing, vol. 44: 774-779.

[26] M. Sarcinelli Filho e A. C. S. Simmer (1995). "Avaliação do Desempenho a Nível de Ruído para Estruturas Digitais de Segunda Ordem sem Ciclos Limite", SBA - Controle e Automação, vol. 6: 89-95.

[27] P. V. Amanda-Mohan (1989). "Coments on Calculation of L_∞ Norms for Scaling Second-Order State-Space Digital Filters Sections", IEEE Transactions on Circuits and Systems, vol. 36: 310-311.

[28] T. I. Laakso (1992). "Coments on Coments on Calculation of L_∞ Norms for Scaling Second-Order State-Space Digital Filters Sections", IEEE Transactions on Circuits and Systems-Part II: Analog and Digital Signal Processing,. vol. 39: 256.

[29] Higgins, W. E. e Musson Jr., D.C. (1982). "Noise Reduction Strategies for Digital Filters: Error Spectrum Shaping versus the Optimal Linear State- Space Formulation", IEEE Transactions on Acoustics, and Signal Processinh,. vol. ASSP-30: 963-973.

[30] Macedo Jr., T. C. (1990), "Modelagem do Erro Adaptativa para Redução dos Efeitos de Quantização em Filtros Digitais". Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro/RJ.

Anexo

Este programa calcula a variância relativa do ruído para filtros do Tipo I, escalado em L_2 [6].

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
%Inicialização dos parâmetros

%arq = input('Entre com caminho e o nome do arquivo: ','s');
[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

alfa1(1)= den(2)  ;
alfa2(1)= den(3)  ;
beta1(1)= num(1)  ;
beta2(1)= num(2)  ;

d = g;

% primeiro passo calculo de "a" e "Ksi"

for i = 1:1,
      a(i) = -alfa1(i)/2;
      ksi(i) = sqrt(alfa2(i)-(alfa1(i)^2)/4);
end


ruidod = 0;

  for i = 1:1,

      mi  = (alfa1(i)+2*alfa2(i))/(alfa1(i)+2);
      num = (2+alfa1(i))^2*((1+alfa2(i))*(1+mi^2)-2*alfa1(i)*mi);
      den = 8*ksi(i)^2*(1+alfa1(i)+alfa2(i));
      sig = sqrt(num/den);

  end

  for j = 1:1,

        A11(j) = a(j);
      A12(j) = -ksi(j)/sig;
      A21(j) = ksi(j)*sig;
```

```
      A22(j) = a(j);

      den1(j) = (1+alfa1(j)+alfa2(j));
      den2(j) = 2*sig*ksi(j)*(1+alfa1(j)+alfa2(j));

      C1(j) = beta1(j)+beta2(j);
      C1(j) = C1(j)/den1(j);
      C2(j) = -beta1(j)*(alfa1(j)+2*alfa2(j))+beta2(j)*(2+alfa1(j));
      C2(j) = C2(j)/den2(j);


   end

% Procurar a constante de escalonamento lamb

    for m = 1:1,

      %calculo norma de "Fa" => NormFa = 1 + NormF1

      Cm(m) = 1-A11(m);
      Dm(m) = ksi(m)^2-A11(m)+A11(m)^2;
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormF1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormF1(m) = NormF1(m)/den(m);

      NormFa(m) = 1 + NormF1(m);


     %calculo norma de "F2"

      Cm(m) = -ksi(m)*sig;
      Dm(m) = ksi(m)*sig;
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormF2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormF2(m) = NormF2(m)/den(m);


      if NormFa(m) > NormF2(m)
            lamb(m) = 1/sqrt(NormFa(m));
      else
          lamb(m) = 1/sqrt(NormF2(m));
      end

   end

   for l = 1:1,
      C1(l) = C1(l)/lamb(l);
      C2(l) = C2(l)/lamb(l);
   end

   % cálculo do ruído
```

```
   %cálculo da norma quadrática de G1 ao quadrado => NormG1
   for m = 1:1,

      Cm(m) = C1(m);
      Dm(m) = -C1(m)*a(m)+C2(m)*ksi(m)*sig;
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)- (1+dm(m)^2)*bm(m)^2;

      NormG1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormG1(m) = NormG1(m)/den(m);
   end

   %calculo da norma quadrática de G2 ao quadrado => NormG2
    for m = 1:1,

      Cm(m) = C2(m);
      Dm(m) = -C2(m)*a(m)-C1(m)*ksi(m)/sig;
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)- (1+dm(m)^2)*bm(m)^2;

      NormG2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormG2(m) = NormG2(m)/den(m);

   end

   %cálculo da norma quadrática de H ao quadrado => NormH
   for m = 1:1,

      Cm(m) = beta1(m)/lamb(m);
      Dm(m) = beta2(m)/lamb(m);
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)- (1+dm(m)^2)*bm(m)^2;

      NormH(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormH(m) = NormH(m)/den(m);
   end

   %cálculo do ruído

   for n = 1:1,
      ruidod = NormG1(n)+NormG2(n)+NormH(n);
   end

ruidod = ruidod + 1;
ruidod = 10*log10(ruidod);

hold on;
figure(1);
plot(sig,ruidod,'w+');
```

Este programa calcula a variância relativa do ruído para filtros do Tipo III, escalados em L_2 [6].

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
%Inicialização dos parâmetros

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

alfa1(1)= den(2)  ;
alfa2(1)= den(3)  ;
beta1(1)= num(1)  ;
beta2(1)= num(2)  ;

d = g;

% primeiro passo cálculo de "a" e "Ksi"

for i = 1:1,
      a(i) = -alfa1(i)/2;
      ksi(i) = sqrt(alfa2(i)-(alfa1(i)^2)/4);
end


ruidod = 0;

  for i = 1:1,

      den =2*ksi(i);
      sig = (1-alfa2(i))+sqrt((alfa2(i)+1)^2-alfa1(i)^2);
      sig = sig/den;
  end

  for j = 1:1,

     A11(j) = a(j);
      A12(j) = -ksi(j)/sig;
      A21(j) = ksi(j)*sig;
      A22(j) = a(j);

      den1(j) = 2*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);
```

```
    den2(j) = 2*sig*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);

    C1(j) = beta1(j)*(2*ksi(j)*sig+2*alfa2(j)+alfa1(j))-
    beta2(j)*(2+alfa1(j)-2*ksi(j)*sig);
    C1(j) = C1(j)/den1(j);
    C2(j) = beta1(j)*(2*ksi(j)-2*alfa2(j)*sig-
    alfa1(j)*sig)+beta2(j)*(2*sig+alfa1(j)*sig+2*ksi(j));
    C2(j) = C2(j)/den2(j);

end

for m = 1:1,

    %calculo norma de "Fa" => NormFa = 1 + NormF1

    Cm(m) = 1-A11(m)-A12(m);
    Dm(m) = A11(m)^2-A11(m)+ksi(m)^2+A12(m);
    bm(m) = -(A11(m)+A22(m));
    dm(m) = A11(m)*A22(m)-A12(m)*A21(m);

    den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

    NormF1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
    Dm(m)*Cm(m)*dm(m))*bm(m);
    NormF1(m) = NormF1(m)/den(m);

    NormFa(m) = 1 + NormF1(m);

   %calculo norma de "Fb" => NormFb = 1 + NormF2

    Cm(m) = 1-A11(m)-A21(m);
    Dm(m) = A11(m)^2-A11(m)+ksi(m)^2+A21(m);
    bm(m) = -(A11(m)+A22(m));
    dm(m) = A11(m)*A22(m)-A12(m)*A21(m);

    den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

    NormF2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
    Dm(m)*Cm(m)*dm(m))*bm(m);
    NormF2(m) = NormF2(m)/den(m);

    NormFb(m) = 1 + NormF2(m);

    if NormFa(m) > NormFb(m)
          lamb(m) = 1/sqrt(NormFa(m));
    else
        lamb(m) = 1/sqrt(NormFb(m));
    end
    %lamb(m) = 8.7017906252E-01;
 end

 for l = 1:1,
    C1(l) = C1(l)/lamb(l);
    C2(l) = C2(l)/lamb(l);
 end

 %cálculo do ruído

 %cálculo da norma quadrática de G1 ao quadrado => NormG1
 for m = 1:1,
```

```
      Cm(m) = C1(m);
      Dm(m) = C2(m)*A21(m)-C1(m)*A22(m);
      bm(m) = -(A11(m)+A22(m));
      dm(m) = A11(m)*A22(m)-A12(m)*A21(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormG1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormG1(m) = NormG1(m)/den(m);
   end

   %cálculo da norma quadrática de G2 ao quadrado => NormG2
    for m = 1:1,

      Cm(m) = C2(m);
      Dm(m) = C1(m)*A12(m)-C2(m)*A11(m);
      bm(m) = -(A11(m)+A22(m));
      dm(m) = A11(m)*A22(m)-A12(m)*A21(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormG2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormG2(m) = NormG2(m)/den(m);

   end

   %cálculo da norma quadrática de H ao quadrado => NormH
   for m = 1:1,

      Cm(m) = beta1(m)/lamb(m);
      Dm(m) = beta2(m)/lamb(m);
      bm(m) = -(A11(m)+A22(m));
      dm(m) = A11(m)*A22(m)-A12(m)*A21(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormH(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormH(m) = NormH(m)/den(m);

   end


   %calculo do ruído

   for n = 1:1,

      ruidod = NormG1(n)+NormG2(n)+NormH(n);

   end


ruidod = ruidod + 1;
ruidod = 10*log10(ruidod);

hold on;
figure(1);
plot(sig,ruidod,'wx');
```

Este programa plota a variância relativa do ruído para filtros do tipo III, escalados L_2, em função da variável σ.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
%Inicialização dos parâmetros
nvz = 20;

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

alfa1(1)= den(2)  ;
alfa2(1)= den(3)  ;
beta1(1)= num(1)  ;
beta2(1)= num(2)  ;



d= g;

% primeiro passo cálculo de "a" e "Ksi"

for i = 1:1,
      a(i) = -alfa1(i)/2;
      ksi(i) = sqrt(alfa2(i)-(alfa1(i)^2)/4);
end


% segundo passo variar o sigma a partir de um valor inicial

index = 0;
aux = 1/nvz;
for sig = 0.5:aux:4,

   index = index +1;
   sigv(index) = sig;

   % computar as matrizes A e C

   for j = 1:1,

      A11(j) = a(j);
```

```
   A12(j) = -ksi(j)/sig;
   A21(j) = ksi(j)*sig;
   A22(j) = a(j);

   den1(j) = 2*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);
   den2(j) = 2*sig*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);

   C1(j) = beta1(j)*(2*ksi(j)*sig+2*alfa2(j)+alfa1(j))-
   beta2(j)*(2+alfa1(j)-2*ksi(j)*sig);
   C1(j) = C1(j)/den1(j);
   C2(j) = beta1(j)*(2*ksi(j)-2*alfa2(j)*sig-
   alfa1(j)*sig)+beta2(j)*(2*sig+alfa1(j)*sig+2*ksi(j));
   C2(j) = C2(j)/den2(j);

end

% computar as constantes de escalamento "lamb"

for m = 1:1,

   %calculo norma de "Fa" => NormFa = 1 + NormF1

   Cm(m) = 1-A11(m)-A12(m);
   Dm(m) = A11(m)^2-A11(m)+ksi(m)^2+A12(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormF1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormF1(m) = NormF1(m)/den(m);

   NormFa(m) = 1 + NormF1(m);
  NormF11(index) = sqrt(NormFa(m));

  %calculo norma de "Fb" => NormFb = 1 + NormF2

   Cm(m) = 1-A11(m)-A21(m);
   Dm(m) = A11(m)^2-A11(m)+ksi(m)^2+A21(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormF2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormF2(m) = NormF2(m)/den(m);

   NormFb(m) = 1 + NormF2(m);
   NormF22(index) = sqrt(NormFb(m));


   if NormFa(m) > NormFb(m)
         lamb(m) = 1/sqrt(NormFa(m));
   else
       lamb(m) = 1/sqrt(NormFb(m));
   end

end
```

```
% A fim de restaurar o nível do sinal na saída substituir "C"

for l = 1:1,
   C1(l) = C1(l)/lamb(l);
   C2(l) = C2(l)/lamb(l);
end

% calculo do ruído

%calculo da norma quadrática de G1 ao quadrado => NormG1
for m = 1:1,

   Cm(m) = C1(m);
   Dm(m) = C2(m)*A21(m)-C1(m)*A22(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormG1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormG1(m) = NormG1(m)/den(m);
  NormG11(index) = sqrt(NormG1(m));
end

%cálculo da norma quadrática de G2 ao quadrado => NormG2
 for m = 1:1,

   Cm(m) = C2(m);
   Dm(m) = C1(m)*A12(m)-C2(m)*A11(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormG2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormG2(m) = NormG2(m)/den(m);
  NormG22(index) = sqrt(NormG2(m));
end

%calculo da norma quadrática de H ao quadrado => NormH
for m = 1:1,

   Cm(m) = beta1(m)/lamb(m);
   Dm(m) = beta2(m)/lamb(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormH(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormH(m) = NormH(m)/den(m);

end


%calculo do ruído

for n = 1:1,
```

```
      ruido(index) = NormG1(n)+NormG2(n)+NormH(n);

   end


ruido(index) = ruido(index) + 1;
ruido(index) = 10*log10(ruido(index));

end

%---------------------- Plotar os gráficos ------------------

figure(1);
hold on;
subplot(3,1,1), plot(sigv,ruido,'w:');
subplot(3,1,1), title('Variância Relativa do Ruído na Saída (dB)');
subplot(3,1,2), plot(sigv,NormF11,'w:');
hold on;
subplot(3,1,2), plot(sigv,NormF22,'w--');
subplot(3,1,2), title('Normas Quadráticas de [F1(z)-1] e [F2(z)-1]');
subplot(3,1,3), plot(sigv,NormG11,'w:');
hold on;
subplot(3,1,3), plot(sigv,NormG22,'w--');
subplot(3,1,3), xlabel('Variação de Sigma');
subplot(3,1,3), title('Normas Quadráticas de G1(z) e G2(z)');
```

Este programa plota a variância relativa do ruído para filtros do Tipo III, escalados em L_∞, em função da variável σ.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
%inicialização dos parâmetro
nvz = 200;

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

alfa1(1)= den(2)  ;
alfa2(1)= den(3)  ;
beta1(1)= num(1)  ;
beta2(1)= num(2)  ;



d= g;

% primeiro passo calculo de "a" e "Ksi"

for i = 1:1,
      a(i) = -alfa1(i)/2;
      ksi(i) = sqrt(alfa2(i)-(alfa1(i)^2)/4);
end

%for n = 1:nvz-8,
%     ruido(n) = 0;
%end

% segundo passo variar o sigma a partir de um valor inicial

index = 0;
aux = 1/nvz;
for sig = 0.1:aux:5,

   index = index +1;
   sigv(index) = sig;

% computar as matrizes A e C
```

```
   for j = 1:1,

        A11(j) = a(j);
      A12(j) = -ksi(j)/sig;
      A21(j) = ksi(j)*sig;
      A22(j) = a(j);

      den1(j) = 2*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);
      den2(j) = 2*sig*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);

      C1(j) = beta1(j)*(2*ksi(j)*sig+2*alfa2(j)+alfa1(j))-
      beta2(j)*(2+alfa1(j)-2*ksi(j)*sig);
      C1(j) = C1(j)/den1(j);
      C2(j) = beta1(j)*(2*ksi(j)-2*alfa2(j)*sig-
      alfa1(j)*sig)+beta2(j)*(2*sig+alfa1(j)*sig+2*ksi(j));
      C2(j) = C2(j)/den2(j);

   end

% computar a constante de escalamento "lamb"

%calculo da norma infinita de "Fa" => NormFa

   for m = 1:1,

      p = -1;
      q = -1*(1+A11(m)-A12(m));
      r = A12(m)-A11(m);
      u = A11(m)+A22(m);
      v = A11(m)*A22(m)-A12(m)*A21(m);

      a0 = (p*r)/v;
      b0 = (p+r)*q/(2*v);
      c0 = ((p-r)^2 + (q^2))/(4*v);
      d0 = (1+v)*u/(2*v);
      e0 = ((1-v)^2 + (u^2))/(4*v);

      a1 = (d0^2) - 4*e0;
      b1 = (4*c0) + (4*a0*e0) - (2*b0*d0);
      c1 = (b0^2) - (4*a0*c0);

      kquadrado1 = (-b1+sqrt(b1^2-(4*a1*c1)))/(2*a1);
      kquadrado2 = (-b1-sqrt(b1^2-(4*a1*c1)))/(2*a1);

      if kquadrado1 > kquadrado2
            k = sqrt(kquadrado1);
      else
                k = sqrt(kquadrado2);
      end

      a2 = k^2-a0;
      b2 = d0*(k^2)-b0;
      c2 = (k^2)*e0-c0;

      x1 = (-b2+sqrt(b2^2-4*a2*c2))/(2*a2);
      x2 = (-b2-sqrt(b2^2-4*a2*c2))/(2*a2);

      k1 = sqrt((a0-b0+c0)/(1-d0+e0));
      k2 = sqrt((a0+b0+c0)/(1+d0+e0));

      if k1 > k2
```

```
            NormFa(m) = k1;
      else
            NormFa(m) = k2;
      end

      if imag(x1) <= 0.0001
            if x1 > -1 & x1 < 1
                  NormFa(m) = k;
            end
            end

      if imag(x2) <= 0.0001
            if x2 > -1 & x2 < 1
                  NormFa(m) = k;
            end
            end

      NormFaa(index) = NormFa(m);

%calculo da norma infinita de "Fb" => NormFb

      p = -1;
      q = -1*(1+A11(m)-A21(m));
      r = A21(m)-A11(m);
      u = A11(m)+A22(m);
      v = A11(m)*A22(m)-A12(m)*A21(m);

      a0 = (p*r)/v;
      b0 = (p+r)*q/(2*v);
      c0 = ( (p-r)^2 +  (q^2) )/(4*v);
      d0 = (1+v)*u/(2*v);
      e0 = ( (1-v)^2 + (u^2) )/(4*v);

      a1 = (d0^2) - 4*e0;
      b1 = (4*c0) + (4*a0*e0) - (2*b0*d0);
      c1 = (b0^2) - (4*a0*c0);

      kquadrado1 = (-b1+sqrt(b1^2-(4*a1*c1)))/(2*a1);
      kquadrado2 = (-b1-sqrt(b1^2-(4*a1*c1)))/(2*a1);

      if kquadrado1 > kquadrado2
            k = sqrt(kquadrado1);
      else
                k = sqrt(kquadrado2);
      end

      a2 = k^2-a0;
      b2 = d0*(k^2)-b0;
      c2 = (k^2)*e0-c0;

      x1 = (-b2+sqrt(b2^2-4*a2*c2))/(2*a2);
      x2 = (-b2-sqrt(b2^2-4*a2*c2))/(2*a2);

      k1 = sqrt((a0-b0+c0)/(1-d0+e0));
      k2 = sqrt((a0+b0+c0)/(1+d0+e0));

      if k1 > k2
            NormFb(m) = k1;
      else
            NormFb(m) = k2;
      end
```

```
      if imag(x1) <= 0.0001
            if x1 > -1 & x1 < 1
                  NormFb(m) = k;
            end
            end

      if imag(x2) <= 0.0001
            if x2 > -1 & x2 < 1
                  NormFb(m) = k;
            end
            end

      NormFbb(index) = NormFb(m);

      if NormFa(m) > NormFb(m)
            lamb(m) = 1/NormFa(m);
      else
          lamb(m) = 1/NormFb(m);
      end

   end

% A fim de restaurar o nível do sinal na saída substituir "C"

   for l = 1:1,
      C1(l) = C1(l)/lamb(l);
      C2(l) = C2(l)/lamb(l);
   end

% calculo do ruído

%cálculo da norma quadrática de G1 ao quadrado => NormG1
   for m = 1:1,

      Cm(m) = C1(m);
      Dm(m) = C2(m)*A21(m)-C1(m)*A22(m);
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormG1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormG1(m) = NormG1(m)/den(m);
      NormG11(index) = sqrt(NormG1(m));
   end

%cálculo da norma quadrática de G2 ao quadrado => NormG2
    for m = 1:1,

      Cm(m) = C2(m);
      Dm(m) = C1(m)*A12(m)-C2(m)*A11(m);
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormG2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormG2(m) = NormG2(m)/den(m);
```

```
      NormG22(index) = sqrt(NormG2(m));
   end

%cálculo da norma quadrática de H ao quadrado => NormH
   for m = 1:1,

      Cm(m) = beta1(m)/lamb(m);
      Dm(m) = beta2(m)/lamb(m);
      bm(m) = alfa1(m);
      dm(m) = alfa2(m);

      den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

      NormH(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
      Dm(m)*Cm(m)*dm(m))*bm(m);
      NormH(m) = NormH(m)/den(m);

   end


%cálculo do ruído

   for n = 1:1,
      ruido(index) = NormG1(n)+NormG2(n)+NormH(n);
   end

ruido(index) = ruido(index) + 1;
ruido(index) = 10*log10(ruido(index));

end

%---------------------- Plotar os gráficos -------------------

figure(1);
hold on;
subplot(3,1,1), plot(sigv,ruido,'w:');
subplot(3,1,1), title('Variância Relativa do Ruído na Saída (dB)');
subplot(3,1,2), plot(sigv,NormFaa,'w:');
hold on;
subplot(3,1,2), plot(sigv,NormFbb,'w--');
subplot(3,1,2), title('Normas Quadráticas de [F1(z)-1] e [F2(z)-1]');
subplot(3,1,3), plot(sigv,NormG11,'w:');
hold on;
subplot(3,1,3), plot(sigv,NormG22,'w--');
subplot(3,1,3), xlabel('Sigma variation');
subplot(3,1,3), title('Normas Quadráticas de G1(z) e G2(z)');
hold on;
```

Este programa varia a banda passante de um filtro protótipo de 1% de banda passante e plota a variância relativa do ruído de cada transformação espectral em função da Largura de Banda Relativa, para filtros do tipo III ótimos, escalados em L_2, para vários blocos.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
%Inicialização dos parâmetros
nvz = 20;

for i = 1:20,

% inicialização dos vetores de ruído para cada freqüência

minruidototal(i)=1;
minGvtotal(i)=1;
minFvtotal(i)=1;

end

%preparado para leitura dos arquivo gerados pelo programa do Arnaldo

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,117,-1);
numblocos = fscanf(fid,'%f');
j=0;
for ii=1:numblocos,

for i=1:6,
   fseek(fid,172+j,-1);
   if i<4
         num(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end  % if i<4
   j=j+37;
end  % i=1:6 (loop de leitura de cada bloco)

alfa1(ii) = den(2)  ;
alfa2(ii) = den(3)  ;
beta1(ii) = num(2)  ;
beta2(ii) = num(3)  ;
end % ii = 1:numblocos (loop de leitura de todos os blocos)

% leitura do termo independente d

fseek(fid,172+j,-1);
d=fscanf(fid,'%f');

fclose(fid);

% Preencher o vetor alfa da transformação de Contantinides.
% estou partindo sempre de um filtro passa baixa 1% e aumentando a largura
de banda de 1 em 1.
```

```
for i = 1:20,
      al(i) = sin(-0.01*3.14);
      x = 0.01 + 2*i/100;
      xx(i) = x;
      al(i) = al(i)/sin(x*3.14);
end


for ii=1:numblocos,     %loop para variação de vários blocos

% primeiro passo calculo de "a" e "Ksi"


      a = -alfa1(ii)/2;
      ksi = sqrt(alfa2(ii)-(alfa1(ii)^2)/4);


% neste ponto  está o loop para mudança de largura de banda

for i = 1:20,
   alf(i)=i;

% segundo passo variar o sigma a partir de um valor inicial

index = 0;
aux = 1/nvz;
for sig = 0.5:aux:4,

   index = index +1;
   sigv(index) = sig;

   % computar as matrizes A e C



      A11 = a;
      A12 = -ksi/sig;
      A21 = ksi*sig;
      A22 = a;

      den1 = 2*(1+alfa1(ii)+alfa2(ii))*(ksi*sig+ksi/sig);
      den2 = 2*sig*(1+alfa1(ii)+alfa2(ii))*(ksi*sig+ksi/sig);

      C1=beta1(ii)*(2*ksi*sig+2*alfa2(ii)+alfa1(ii))-
      beta2(ii)*(2+alfa1(ii)-2*ksi*sig);
      C1 = C1/den1;
      C2 = beta1(ii)*(2*ksi-2*alfa2(ii)*sig-
      alfa1(ii)*sig)+beta2(ii)*(2*sig+alfa1(ii)*sig+2*ksi);
      C2 = C2/den2;

% Completar as matrizes para uso nas transformações espectrais (nos
produtos de Kroneker)
      if sig == 1

      A = [A11 A12; A21 A22];
      B = [(1-A11-A12) ;(1-A22-A21)];
      C = [C1 C2];

      end % if sig == 1

   % computar as constantes de escalamento "lamb"
```

```
   %calculo norma de "Fa" => NormFa = 1 + NormF1

   Cm = 1-A11-A12;
   Dm = A11^2-A11+ksi^2+A12;
   bm = alfa1(ii);
   dm = alfa2(ii);

   den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
   NormF1 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
   NormF1 = NormF1/den;

   NormFa = 1 + NormF1;
   NormF11(index) = sqrt(NormFa);

  %calculo norma de "Fb" => NormFb = 1 + NormF2

   Cm = 1-A11-A21;
   Dm = A11^2-A11+ksi^2+A21;
   bm = alfa1(ii);
   dm = alfa2(ii);

   den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
   NormF2 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
   NormF2 = NormF2/den;

   NormFb = 1 + NormF2;
   NormF22(index) = sqrt(NormFb);


   if NormFa > NormFb
         lamb = 1/sqrt(NormFa);
   else
       lamb = 1/sqrt(NormFb);
   end     % if NormFa > Norm Fb


% A fim de restaurar o nivel do sinal na saída substituir "C"


   C1 = C1/lamb;
   C2 = C2/lamb;

%cálculo do ruído

%cálculo da norma quadrática de G1 ao quadrado => NormG1


   Cm = C1;
   Dm = C2*A21-C1*A22;
   bm = alfa1(ii);
   dm = alfa2(ii);

   den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
   NormG1 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
   NormG1 = NormG1/den;

   NormG11(index) = sqrt(NormG1);
```

```
   %calculo da norma quadrática de G2 ao quadrado => NormG2

      Cm = C2;
      Dm = C1*A12-C2*A11;
      bm = alfa1(ii);
      dm = alfa2(ii);

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormG2 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormG2 = NormG2/den;

      NormG22(index) = sqrt(NormG2);

   %cálculo da norma quadrática de H ao quadrado => NormH

      Cm = beta1(ii)/lamb;
      Dm = beta2(ii)/lamb;
      bm = alfa1(ii);
      dm = alfa2(ii);

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormH = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormH = NormH/den;

%cálculo do ruído

ruido(index) = NormG1+NormG2+NormH;


end    %  for sig = 0.5:aux:4 (fecha o loop da variação de sigma)

% descobrir os mínimos

[minruido,imin]=min(ruido);
minruidv(i)= minruido;
[minG,iG]=min(abs(NormG22-NormG11));
minGv(i)=ruido(iG);
[minF,iF]=min(abs(NormF22-NormF11));
minFv(i)=ruido(iF);

% resolução via produtos de Kronecker

AA = ((al(i)*eye(2)+A))*(inv((eye(2)+al(i)*A)));
BB = ((1-al(i)^2)^0.5)*inv((eye(2)+al(i)*A))*B;
CC = ((1-al(i)^2)^(0.5))*C*inv((eye(2)+al(i)*A));

alfa1(ii) = -(AA(1,1)+AA(2,2));
alfa2(ii) = AA(1,1)*AA(2,2)-AA(2,1)*AA(1,2);
beta1(ii) = BB(1)*CC(1)+BB(2)*CC(2);
beta2(ii) = BB(1)*(CC(2)*AA(2,1)-CC(1)*AA(2,2))+BB(2)*(CC(1)*AA(1,2)-
CC(2)*AA(1,1));
a = AA(1,1);
ksi = -AA(1,2);

end    % for i=1:20 (fecha o loop das frequências)

minruidototal = minruidototal + minruidv;
minGvtotal = minGvtotal + minGv;
minFvtotal = minFvtotal + minFv;

end % for ii=1:numblocos (fecha os loops dos blocos)
```

```
% loop de cálculo da variância do ruído em dB, para cada freqüência

for i=1:20,
   minruidototal(i)=10*log10(minruidototal(i));
   minGvtotal(i)=10*log10(minGvtotal(i));
   minFvtotal(i)=10*log10(minFvtotal(i));
end %for i=1:20

%---------------------- Plotar os gráficos -------------------


figure(1);
hold on;
plot(alf,minruidototal,'k:');

plot(alf,minFvtotal,'k-');

plot(alf,minGvtotal,'w-.');

xlabel('Largura de Banda Relativa BW/fs (%)');
ylabel('Variância Relativa do Ruído na Saída (dB)');
```

```
_____________________________________________________________________

Este programa varia a banda passante de um filtro protótipo de 1% de banda
passante e plota a variância relativa do ruído de cada transformação
espectral em função da Largura de Banda Relativa, para filtros do tipo III
ótimo, escalados em L∞.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho
_____________________________________________________________________

%Inicialização dos parâmetros
nvz = 20;

for i = 1:20,

% inicialização dos vetores de ruído para cada freqüência

minruidototal(i)=1;
minGvtotal(i)=1;
minFvtotal(i)=1;

end

%preparado para leitura dos arquivo gerados pelo programa do Arnaldo

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,117,-1);
numblocos = fscanf(fid,'%f');
j=0;
for ii=1:numblocos,

for i=1:6,
   fseek(fid,172+j,-1);
   if i<4
         num(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end  % if i<4
   j=j+37;
end  % i=1:6 (loop de leitura de cada bloco)

alfa1(ii) = den(2)  ;
alfa2(ii) = den(3)  ;
beta1(ii) = num(2)  ;
beta2(ii) = num(3)  ;
end % ii = 1:numblocos (loop de leitura de todos os blocos)

% leitura do termo independente d

fseek(fid,172+j,-1);
d=fscanf(fid,'%f');

fclose(fid);

% Preencher o vetor alfa da transformação de Constantinides.
% estou partindo sempre de um filtro passa baixa 1% e aumentando a largura
de banda de 1 em 1.
```

```
for i = 1:20,
      al(i) = sin(-0.01*3.14);
      x = 0.01 + 2*i/100;
      xx(i) = x;
      al(i) = al(i)/sin(x*3.14);
end


for ii=1:numblocos,     %loop para variação de vários blocos

% primeiro passo calculo de "a" e "Ksi"


      a = -alfa1(ii)/2;
      ksi = sqrt(alfa2(ii)-(alfa1(ii)^2)/4);


% neste ponto  está o loop para mudança de largura de banda

for i = 1:20,
   alf(i)=i;

% segundo passo variar o sigma a partir de um valor inicial

index = 0;
aux = 1/nvz;
for sig = 0.5:aux:4,

   index = index +1;
   sigv(index) = sig;

   % computar as matrizes A e C



      A11 = a;
      A12 = -ksi/sig;
      A21 = ksi*sig;
      A22 = a;

      den1 = 2*(1+alfa1(ii)+alfa2(ii))*(ksi*sig+ksi/sig);
      den2 = 2*sig*(1+alfa1(ii)+alfa2(ii))*(ksi*sig+ksi/sig);

      C1 = beta1(ii)*(2*ksi*sig+2*alfa2(ii)+alfa1(ii))-
      beta2(ii)*(2+alfa1(ii)-2*ksi*sig);
      C1 = C1/den1;
      C2 = beta1(ii)*(2*ksi-2*alfa2(ii)*sig-
      alfa1(ii)*sig)+beta2(ii)*(2*sig+alfa1(ii)*sig+2*ksi);
      C2 = C2/den2;

% Completar as matrizes para uso nas transformações espectrais (nos
produtos de Kroneker)
      if sig == 1

      A = [A11 A12; A21 A22];
      B = [(1-A11-A12) ;(1-A22-A21)];
      C = [C1 C2];

      end % if sig == 1

     %computar as constantes de escalamento "lamb"
```

```
%calculo norma infinita de "Fa" => NormFa



p = -1;
q = -1*(1+A11-A12);
r = A12-A11;
u = A11+A22;
v = A11*A22-A12*A21;

a0 = (p*r)/v;
b0 = (p+r)*q/(2*v);
c0 = ((p-r)^2 + (q^2))/(4*v);
d0 = (1+v)*u/(2*v);
e0 = ((1-v)^2 + (u^2))/(4*v);

a1 = (d0^2) - 4*e0;
b1 = (4*c0) + (4*a0*e0) - (2*b0*d0);
c1 = (b0^2) - (4*a0*c0);

kquadrado1 = (-b1+sqrt(b1^2-(4*a1*c1)))/(2*a1);
kquadrado2 = (-b1-sqrt(b1^2-(4*a1*c1)))/(2*a1);

if kquadrado1 > kquadrado2
      k = sqrt(kquadrado1);
else
          k = sqrt(kquadrado2);
end

a2 = k^2-a0;
b2 = d0*(k^2)-b0;
c2 = (k^2)*e0-c0;

x1 = (-b2+sqrt(b2^2-4*a2*c2))/(2*a2);
x2 = (-b2-sqrt(b2^2-4*a2*c2))/(2*a2);

k1 = sqrt((a0-b0+c0)/(1-d0+e0));
k2 = sqrt((a0+b0+c0)/(1+d0+e0));

if k1 > k2
      NormFa = k1;
else
      NormFa = k2;
end

if imag(x1) <= 0.0001
      if x1 > -1 & x1 < 1
            NormFa = k;
      end
      end

if imag(x2) <= 0.0001
      if x2 > -1 & x2 < 1
            NormFa = k;
      end
      end

NormFaa(index) = NormFa;
```

```
%cálculo da norma infinita de "Fb" => NormFb

      p = -1;
      q = -1*(1+A11-A21);
      r = A21-A11;
      u = A11+A22;
      v = A11*A22-A12*A21;

      a0 = (p*r)/v;
      b0 = (p+r)*q/(2*v);
      c0 = ( (p-r)^2 +  (q^2) )/(4*v);
      d0 = (1+v)*u/(2*v);
      e0 = ( (1-v)^2 + (u^2) )/(4*v);

      a1 = (d0^2) - 4*e0;
      b1 = (4*c0) + (4*a0*e0) - (2*b0*d0);
      c1 = (b0^2) - (4*a0*c0);

      kquadrado1 = (-b1+sqrt(b1^2-(4*a1*c1)))/(2*a1);
      kquadrado2 = (-b1-sqrt(b1^2-(4*a1*c1)))/(2*a1);

      if kquadrado1 > kquadrado2
            k = sqrt(kquadrado1);
      else
                k = sqrt(kquadrado2);
      end

      a2 = k^2-a0;
      b2 = d0*(k^2)-b0;
      c2 = (k^2)*e0-c0;

      x1 = (-b2+sqrt(b2^2-4*a2*c2))/(2*a2);
      x2 = (-b2-sqrt(b2^2-4*a2*c2))/(2*a2);

      k1 = sqrt((a0-b0+c0)/(1-d0+e0));
      k2 = sqrt((a0+b0+c0)/(1+d0+e0));

      if k1 > k2
            NormFb = k1;
      else
            NormFb = k2;
      end

      if imag(x1) <= 0.0001
            if x1 > -1 & x1 < 1
                  NormFb = k;
            end
            end

      if imag(x2) <= 0.0001
            if x2 > -1 & x2 < 1
                  NormFb = k;
            end
            end

      NormFbb(index) = NormFb;

      if NormFa > NormFb;
            lamb = 1/NormFa;
      else
      lamb = 1/NormFb;
```

```
      end


   % A fim de restaurar o nível do sinal na saída substituir "C"


      C1 = C1/lamb;
      C2 = C2/lamb;

   % cálculo do ruído

   %cálculo da norma quadrática de G1 ao quadrado => NormG1


      Cm = C1;
      Dm = C2*A21-C1*A22;
      bm = alfa1(ii);
      dm = alfa2(ii);

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormG1 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormG1 = NormG1/den;

      NormG11(index) = sqrt(NormG1);


   %cálculo da norma quadrática de G2 ao quadrado => NormG2

      Cm = C2;
      Dm = C1*A12-C2*A11;
      bm = alfa1(ii);
      dm = alfa2(ii);

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormG2 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormG2 = NormG2/den;

      NormG22(index) = sqrt(NormG2);

   %cálculo da norma quadrática de H ao quadrado => NormH

      Cm = beta1(ii)/lamb;
      Dm = beta2(ii)/lamb;
      bm = alfa1(ii);
      dm = alfa2(ii);

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormH = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormH = NormH/den;

%cálculo do ruído

ruido(index) = NormG1+NormG2+NormH;


end    %  for sig = 0.5:aux:4 (fecha o loop da variação de sigma)

% descobrir os mínimos

[minruido,imin]=min(ruido);
minruidv(i)= minruido;
```

```
[minG,iG]=min(abs(NormG22-NormG11));
minGv(i)=ruido(iG);
[minF,iF]=min(abs(NormFaa-NormFbb));
minFv(i)=ruido(iF);

% resolução via produtos de Kronecker

AA = ((al(i)*eye(2)+A))*(inv((eye(2)+al(i)*A)));
BB = ((1-al(i)^2)^0.5)*inv((eye(2)+al(i)*A))*B;
CC = ((1-al(i)^2)^(0.5))*C*inv((eye(2)+al(i)*A));

alfa1(ii) = -(AA(1,1)+AA(2,2));
alfa2(ii) = AA(1,1)*AA(2,2)-AA(2,1)*AA(1,2);
beta1(ii) = BB(1)*CC(1)+BB(2)*CC(2);
beta2(ii) = BB(1)*(CC(2)*AA(2,1)-CC(1)*AA(2,2))+BB(2)*(CC(1)*AA(1,2)-
CC(2)*AA(1,1));
a = AA(1,1);
ksi = -AA(1,2);

end    % for i=1:20 (fecha o loop das frequências)

minruidototal = minruidototal + minruidv;
minGvtotal = minGvtotal + minGv;
minFvtotal = minFvtotal + minFv;

end % for ii=1:numblocos (fecha os loops dos blocos)

% loop de cálculo da variância do ruído em dB, para cada freqüência

for i=1:20,
   minruidototal(i)=10*log10(minruidototal(i));
   minGvtotal(i)=10*log10(minGvtotal(i));
   minFvtotal(i)=10*log10(minFvtotal(i));
end %for i=1:20

%---------------------- Plotar os gráficos -------------------


figure(1);
hold on;
plot(alf,minruidototal,'k:');
plot(alf,minFvtotal,'k-');
plot(alf,minGvtotal,'w-.');
xlabel('Largura de Banda Relativa BW/fs (%)');
ylabel('Variância Relativa do Ruído na Saída (dB)');
```

Este programa varia a banda passante de um filtro protótipo e plota a variância relativa do ruído cada transformação espectral em função da Largura de Banda Relativa, para filtros de mínimo ruído, escalados em L_2.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
%Inicialização dos parâmetros
nvz = 20;

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

alfa1 = den(2)  ;
alfa2 = den(3)  ;
beta1 = num(1)  ;
beta2 = num(2)  ;
d= g;

% Preencher o vetor alfa da transformação de Contantinides.
% estou partindo sempre de um filtro passa baixa 1% e aumentando
% a largura de banda de 1 em 1.

for i = 1:20,
      al(i) = sin(-0.01*3.14);
      x = 0.01 + 2*i/100;
      xx(i) = x;
      al(i) = al(i)/sin(x*3.14);
end

% primeiro passo calculo de "a" e "Ksi" computar (A,B,C,d)

      a = -alfa1/2;
      ksi = sqrt(alfa2-(alfa1^2)/4);

% neste ponto  está o loop para mudança de largura de banda

for i = 1:20,
   alf(i)=i;

     A11 = a;
      A12 = -ksi;
      A21 = ksi;
```

```
      A22 = a;

      aux = sqrt(beta2^2-beta1*beta2*alfa1+beta1^2*alfa2);
      B1 = sqrt((aux+beta2+a*beta1)/(2*A21));
      B2 = beta1/(2*B1);
      C1 = B2;
        C2 = B1;

      A = [A11 A12; A21 A22];
      B = [B1 ; B2];
      C = [C1 C2];


  % Fazer o escalonamento da rede via matriz T
  %calculo norma de "F1"

      Cm = B1;
      Dm = B2*A12-B1*a;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormF1 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormF1 = NormF1/den;
      NormF1 = sqrt(NormF1);


        %calculo norma de "F2"

      Cm = B2;
      Dm = B1*A21-B2*a;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormF2 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormF2 = NormF2/den;
      NormF2 = sqrt(NormF2);

      T = [NormF1 0; 0 NormF2];

% escalonamento

      Ax = inv(T)*A*T;
      Bx = inv(T)*B;
      Cx = C*T;

% Novas variáveis escalonadas

      A11 = Ax(1,1);
      A12 = Ax(1,2);
      A21 = Ax(2,1);
      A22 = Ax(2,2);
      B1  = Bx(1);
      B2  = Bx(2);
      C1  = Cx(1);
      C2  = Cx(2);


%calculo da norma quadrática de G1 ao quadrado => NormG1
```

```
      Cm = C1;
      Dm = C2*A21-C1*A22;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormG1 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormG1 = NormG1/den;
      NormG11(i) = sqrt(NormG1);

%calculo da norma quadrática de G2 ao quadrado => NormG2

      Cm = C2;
      Dm = C1*A12-C2*A11;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormG2 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormG2 = NormG2/den;
      NormG22(i) = sqrt(NormG2);

%calculo do ruído

      ruido(i) = NormG1+NormG2;


ruido(i) = ruido(i) + 1;
ruido(i) = 10*log10(ruido(i));

% resolução via produtos de Kronecker

AA = ((al(i)*eye(2)+A))*(inv((eye(2)+al(i)*A)));
BB = ((1-al(i)^2)^0.5)*inv((eye(2)+al(i)*A))*B;
CC = ((1-al(i)^2)^(0.5))*C*inv((eye(2)+al(i)*A));

alfa1 = -(AA(1,1)+AA(2,2));
alfa2 = AA(1,1)*AA(2,2)-AA(2,1)*AA(1,2);
beta1 = BB(1)*CC(1)+BB(2)*CC(2);
beta2 = BB(1)*(CC(2)*AA(2,1)-CC(1)*AA(2,2))+BB(2)*(CC(1)*AA(1,2)-
CC(2)*AA(1,1));
a = AA(1,1);
ksi = -AA(1,2);

end

%---------------------- Plotar os gráficos -------------------

figure(1);
hold on;
plot(alf,ruido,'k--');
xlabel('Relative Bandwidth BW/fs (%)');
ylabel('Relative Output Roundoff Noise Variance (dB)');
```

Este programa varia a banda passante de um filtro protótipo e plota a variância relativa do ruído cada transformação espectral em função da Largura de Banda Relativa, para filtros de mínimo ruído, escalados em L_∞.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
%Inicialização dos parâmetro
nvz = 20;

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

alfa1 = den(2)  ;
alfa2 = den(3)  ;
beta1 = num(1)  ;
beta2 = num(2)  ;
d= g;

% Preencher o vetor alfa da transformação de Contantinides.
% partindo sempre de um filtro passa baixa 1% e aumentando a largura de
banda de 1 em 1.

for i = 1:20,
      al(i) = sin(-0.01*3.14);
      x = 0.01 + 2*i/100;
      xx(i) = x;
      al(i) = al(i)/sin(x*3.14);
end

% primeiro passo calculo de "a" e "Ksi" computar (A,B,C,d)

      a = -alfa1/2;
      ksi = sqrt(alfa2-(alfa1^2)/4);

% neste ponto  está o loop para mudança de largura de banda

for i = 1:20,
   alf(i)=i;

     A11 = a;
      A12 = -ksi;
      A21 = ksi;
```

```
      A22 = a;

      aux = sqrt(beta2^2-beta1*beta2*alfa1+beta1^2*alfa2);
      B1 = sqrt((aux+beta2+a*beta1)/(2*A21));
      B2 = beta1/(2*B1);
      C1 = B2;
        C2 = B1;

      A = [A11 A12; A21 A22];
      B = [B1 ; B2];
      C = [C1 C2];

%calculo norma de "F1"

      Cm = B1;
      Dm = B2*A12 - B1*A22;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormF1 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormF1 = sqrt(NormF1/den);

%calculo norma de "F2"

      Cm = B2;
      Dm = B1*A21 - B2*A11;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormF2 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormF2 = sqrt(NormF2/den);

% Fazer o escalamento da rede via matriz T

   T = [NormF1 0; 0 NormF2];

% escalamento

      Ax = inv(T)*A*T;
      Bx = inv(T)*B;
      Cx = C*T;

% Novas variáveis escaladas

      A11 = Ax(1,1);
      A12 = Ax(1,2);
      A21 = Ax(2,1);
      A22 = Ax(2,2);
      B1  = Bx(1);
      B2  = Bx(2);
      C1  = Cx(1);
      C2  = Cx(2);

% O Procedimento agora é fazer um reescalamento usando norma infinita

%calculo norma infinita de "F1" => NormFa


      p = 0;                        % Termo em Z2
```

```
      q = -1*(B1);                  % Termo em Z1
      r = B2*A12-B1*A22;            % Termo em Z0
      u = A11+A22;                  % alfa1
      v = A11*A22-A12*A21;          % alfa2


      a0 = (p*r)/v;
      b0 = (p+r)*q/(2*v);
      c0 = ((p-r)^2 + (q^2))/(4*v);
      d0 = (1+v)*u/(2*v);
      e0 = ((1-v)^2 + (u^2))/(4*v);

      a1 = (d0^2) - 4*e0;
      b1 = (4*c0) + (4*a0*e0) - (2*b0*d0);
      c1 = (b0^2) - (4*a0*c0);

      kquadrado1 = (-b1+sqrt(b1^2-(4*a1*c1)))/(2*a1);
      kquadrado2 = (-b1-sqrt(b1^2-(4*a1*c1)))/(2*a1);

      if kquadrado1 > kquadrado2
            k = sqrt(kquadrado1);
      else
                k = sqrt(kquadrado2);
      end

      a2 = k^2-a0;
      b2 = d0*(k^2)-b0;
      c2 = (k^2)*e0-c0;

      x1 = (-b2+sqrt(b2^2-4*a2*c2))/(2*a2);
      x2 = (-b2-sqrt(b2^2-4*a2*c2))/(2*a2);

      k1 = sqrt((a0-b0+c0)/(1-d0+e0));
      k2 = sqrt((a0+b0+c0)/(1+d0+e0));

      if k1 > k2
            NormFa = k1;
      else
            NormFa = k2;
      end

      if imag(x1) <= 0.0001
            if x1 > -1 & x1 < 1
                  NormFa = k;
            end
            end

      if imag(x2) <= 0.0001
            if x2 > -1 & x2 < 1
                  NormFa = k;
            end
            end

%calculo norma infinita de "F2" => NormFb

      p = 0;                        % Termo em Z2
      q = -1*(B2);                  % Termo em Z1
      r = B1*A21-B2*A11;            % Termo em Z0
      u = A11+A22;                  % alfa1
      v = A11*A22-A12*A21;          % alfa2
```

```
      a0 = (p*r)/v;
      b0 = (p+r)*q/(2*v);
      c0 = ( (p-r)^2 +  (q^2) )/(4*v);
      d0 = (1+v)*u/(2*v);
      e0 = ( (1-v)^2 + (u^2) )/(4*v);

      a1 = (d0^2) - 4*e0;
      b1 = (4*c0) + (4*a0*e0) - (2*b0*d0);
      c1 = (b0^2) - (4*a0*c0);

      kquadrado1 = (-b1+sqrt(b1^2-(4*a1*c1)))/(2*a1);
      kquadrado2 = (-b1-sqrt(b1^2-(4*a1*c1)))/(2*a1);

      if kquadrado1 > kquadrado2
            k = sqrt(kquadrado1);
      else
                k = sqrt(kquadrado2);
      end

      a2 = k^2-a0;
      b2 = d0*(k^2)-b0;
      c2 = (k^2)*e0-c0;

      x1 = (-b2+sqrt(b2^2-4*a2*c2))/(2*a2);
      x2 = (-b2-sqrt(b2^2-4*a2*c2))/(2*a2);

      k1 = sqrt((a0-b0+c0)/(1-d0+e0));
      k2 = sqrt((a0+b0+c0)/(1+d0+e0));

      if k1 > k2
            NormFb = k1;
      else
            NormFb = k2;
      end

      if imag(x1) <= 0.0001
            if x1 > -1 & x1 < 1
                  NormFb = k;
            end
            end

      if imag(x2) <= 0.0001
            if x2 > -1 & x2 < 1
                  NormFb = k;
            end
            end

% Fazer o escalonamento da rede via matriz T

      T = [NormFa 0; 0 NormFb];

% escalamento

      Ax = inv(T)*A*T;
      Bx = inv(T)*B;
      Cx = C*T;

% Novas variáveis escaladas

      A11 = Ax(1,1);
      A12 = Ax(1,2);
```

```
      A21 = Ax(2,1);
      A22 = Ax(2,2);
      B1  = Bx(1);
      B2  = Bx(2);
      C1  = Cx(1);
      C2  = Cx(2);



%calculo da norma quadrática de G1 ao quadrado => NormG1

      Cm = C1;
      Dm = C2*A21-C1*A22;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormG1 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormG1 = NormG1/den;
      NormG11(i) = sqrt(NormG1);

%calculo da norma quadrática de G2 ao quadrado => NormG2

      Cm = C2;
      Dm = C1*A12-C2*A11;
      bm = alfa1;
      dm = alfa2;

      den = 1-2*dm^2+dm^4+2*dm*(bm^2)-(1+dm^2)*bm^2;
      NormG2 = (Cm^2+Dm^2)*(1-dm^2)-2*(Dm*Cm-Dm*Cm*dm)*bm;
      NormG2 = NormG2/den;
      NormG22(i) = sqrt(NormG2);

%calculo do ruído

      ruido(i) = NormG1+NormG2;


ruido(i) = ruido(i) + 1;
ruido(i) = 10*log10(ruido(i));

% resolução via produtos de Kronecker

AA = ((al(i)*eye(2)+A))*(inv((eye(2)+al(i)*A)));
BB = ((1-al(i)^2)^0.5)*inv((eye(2)+al(i)*A))*B;
CC = ((1-al(i)^2)^(0.5))*C*inv((eye(2)+al(i)*A));

alfa1 = -(AA(1,1)+AA(2,2));
alfa2 = AA(1,1)*AA(2,2)-AA(2,1)*AA(1,2);
beta1 = BB(1)*CC(1)+BB(2)*CC(2);
beta2 = BB(1)*(CC(2)*AA(2,1)-CC(1)*AA(2,2))+BB(2)*(CC(1)*AA(1,2)-
CC(2)*AA(1,1));
a = AA(1,1);
ksi = -AA(1,2);

end

%---------------------- Plotar os gráficos -------------------
plot(alf,ruido,'k--');
```

Este programa calcula a relação sinal ruído da rede *Error Spectrum Shaping* adaptativa.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho

```
% Rede Error-Spectrum Shaping
%
%   lambda = 1/||H(z)||
%
%   Sn = (q1^2*||H1(z)||^2 + q2^2*||H2(z)||^2)/12  (qi^2/12 para
arredondamento)
%                                                  (qi^2/3  para
truncamento em magnitude)
%   q1=(2^(-L))         q2=(2^(-2L))
%
%   H1'(z) = (z^2 + Beta1*z + Beta0)/(z^2 + b1*z + b0) (H1(z) =
H1'(z)/lambda)
%
%   Rede Error Spectrum Shaping ótima : b1 = Beta1; b2 = Beta2 => H1'(z) =
1
%
%   H2'(z) = 1/(z^2 + b1*z + b0) (H2(z) = H2'(z)/lambda)
%
%   Logo,
%
%   Sn = (q1^2 + q2^2*||H2'(z)||^2)/(lambda^2)
%
%   H(z) = g + g*(a1*z + a0)/(z^2 + b1*z + b0)
%
% Leitura do arquivo de dados
%

[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

% Definição do comprimento de palavra

L=input(' Entre com o comprimento de palavra (n° de bits total - 1): ');


% Calculo da Norma quadratica de H(z) e de lambda (lambda = 1/||H(z)||)
```

```
I1 = g^2;

Cm = num(1);
Dm = num(2);
bm = den(2);
dm = den(3);

aux = (1 - dm^2)^2 - bm^2*(1 - dm)^2;

I3 = ((Cm^2+Dm^2)*(1-dm^2) -2*bm*Dm*Cm*(1-dm))/aux;

Norm_H = sqrt(I1 + I3);     %   = (1/lambda)

% Calculo da Norma quadratica de H1'(z)

V = input('Entre com o percentual de variação dos parâmetros ótimos (%):
');

I1 = 1;

Cm = -den(2)*(V/100);
Dm = den(3)*(V/100);
bm = den(2);
dm = den(3);

aux = (1 - dm^2)^2 - bm^2*(1 - dm)^2;

I3 = ((Cm^2+Dm^2)*(1-dm^2) -2*bm*Dm*Cm*(1-dm))/aux;

Norm_H1 = I1 + I3;     %   = (1/lambda)

% Calculo da norma quadrática de H2'(z) ao quadrado

Cm = 0;
Dm = 1;
bm = den(2);
dm = den(3);

aux = (1 - dm^2)^2 - bm^2*(1 - dm)^2;

I3 = ((Cm^2+Dm^2)*(1-dm^2) -2*bm*Dm*Cm*(1-dm))/aux;;

Norm_H2 = I3;

% Cálculo da relação sinal ruído

q1 = 2^(-L);
q2 = 2^(-2*L);

ruido_a = (q1^2*(Norm_H1) + q2^2*Norm_H2)*(Norm_H^2)/12;   % valor do ruído
absoluto

sinalruido = 4*(2^(2*L))/(Norm_H1 + (2^(-2*L))*Norm_H2);
sinalruidodB = 10*log10(sinalruido)
```

```
Este programa calcula a relação sinal ruído da rede tipo III ótima.

Autor: Marcelo Oliveira Camponêz
Orientador: Mário Sarcinelli Filho
```

```
%inicializacao dos parametros
nvz = 20;

%leitura do arquivo de dados
[f,p]= uigetfile('*.hzc','Escolha o Arquivo');
arq = [p f];
fid=fopen(arq,'r');
fseek(fid,41,-1);
g=fscanf(fid,'%f');
j=0;
for i=1:6,
   fseek(fid,100+j,-1);
   if i<4  numc(4-i)=fscanf(fid,'%f');
      else
         den(7-i)=fscanf(fid,'%f');
   end
   j=j+25;
end

num=g*[numc(2)-den(2) numc(3)-den(3)];
fclose(fid);

%calculos
alfa1(1)= den(2)  ;
alfa2(1)= den(3)  ;
beta1(1)= num(1)  ;
beta2(1)= num(2)  ;

d= g;

% primeiro passo calculo de "a" e "Ksi"

for i = 1:1,
      a(i) = -alfa1(i)/2;
      ksi(i) = sqrt(alfa2(i)-(alfa1(i)^2)/4);
end

% segundo passo variar o sigma a partir de um valor inicial

index = 0;
aux = 1/nvz;
for sig = 0.5:aux:4,

   index = index +1;
   sigv(index) = sig;

% computar as matrizes A e C

   for j = 1:1,

   A11(j) = a(j);
      A12(j) = -ksi(j)/sig;
      A21(j) = ksi(j)*sig;
```

```
   A22(j) = a(j);

   den1(j) = 2*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);
   den2(j) = 2*sig*(1+alfa1(j)+alfa2(j))*(ksi(j)*sig+ksi(j)/sig);

   C1(j) = beta1(j)*(2*ksi(j)*sig+2*alfa2(j)+alfa1(j))-
   beta2(j)*(2+alfa1(j)-2*ksi(j)*sig);
   C1(j) = C1(j)/den1(j);
     C2(j) = beta1(j)*(2*ksi(j)-2*alfa2(j)*sig-
   alfa1(j)*sig)+beta2(j)*(2*sig+alfa1(j)*sig+2*ksi(j));
   C2(j) = C2(j)/den2(j);

end

% computar as constantes de escalamento "lamb"

for m = 1:1,

   %calculo norma de "Fa" => NormFa = 1 + NormF1

   Cm(m) = 1-A11(m)-A12(m);
   Dm(m) = A11(m)^2-A11(m)+ksi(m)^2+A12(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormF1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormF1(m) = NormF1(m)/den(m);

   NormFa(m) = 1 + NormF1(m);
   NormF11(index) = sqrt(NormFa(m));

%calculo norma de "Fb" => NormFb = 1 + NormF2

   Cm(m) = 1-A11(m)-A21(m);
   Dm(m) = A11(m)^2-A11(m)+ksi(m)^2+A21(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormF2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormF2(m) = NormF2(m)/den(m);

   NormFb(m) = 1 + NormF2(m);
   NormF22(index) = sqrt(NormFb(m));


   if NormFa(m) > NormFb(m)
         lamb(m) = 1/sqrt(NormFa(m));
   else
       lamb(m) = 1/sqrt(NormFb(m));
   end

end

%A fim de restaurar o nivel do sinal na saída substituir "C"
```

```
for l = 1:1,
   C1(l) = C1(l)/lamb(l);
   C2(l) = C2(l)/lamb(l);
end

%calculo do ruído

%calculo da norma quadrática de G1 ao quadrado => NormG1
for m = 1:1,

   Cm(m) = C1(m);
   Dm(m) = C2(m)*A21(m)-C1(m)*A22(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormG1(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormG1(m) = NormG1(m)/den(m);
   NormG11(index) = sqrt(NormG1(m));
end

%calculo da norma quadrática de G2 ao quadrado => NormG2
for m = 1:1,

   Cm(m) = C2(m);
   Dm(m) = C1(m)*A12(m)-C2(m)*A11(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormG2(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormG2(m) = NormG2(m)/den(m);
   NormG22(index) = sqrt(NormG2(m));
end

%calculo da norma quadrática de H ao quadrado => NormH
for m = 1:1,

   Cm(m) = beta1(m)/lamb(m);
   Dm(m) = beta2(m)/lamb(m);
   bm(m) = alfa1(m);
   dm(m) = alfa2(m);

   den(m) = 1-2*dm(m)^2+dm(m)^4+2*dm(m)*(bm(m)^2)-(1+dm(m)^2)*bm(m)^2;

   NormH(m) = (Cm(m)^2+Dm(m)^2)*(1-dm(m)^2)-2*(Dm(m)*Cm(m)-
   Dm(m)*Cm(m)*dm(m))*bm(m);
   NormH(m) = NormH(m)/den(m);
   NormHz(index) = NormH(m);
end


%calculo do ruído

for n = 1:1,

   ruido(index) = NormG1(n)+NormG2(n)+NormH(n);    %+ruido(index);
```

```
   end


ruido(index) = ruido(index) + 1;

end

%cálculo do ruído mínimo e determinação do valor de ||H(z)|| correspondente

[minruido,indice] = min(ruido);
NormQHz=NormHz(indice);
signal=NormQHz/3;
L=input('Entre com o número de bits da mantissa: ');
noise=minruido*((2^(-2*L))/12);
SNR = 10*log10(signal/noise)
```