DOI: 10.48011/sbse.v1i1.2270

Uma nova abordagem para o problema das imprecisões numéricas resultantes da utilização de filtros com aritmética inteira

Daniel Carrijo Polonio Araujo [*]
Gabriel de Souza Pereira Gomes [**]
Christos Aristóteles Harissis [***] **Rogério Andrade Flauzino** [*]

[*] *Esc. Eng. de São Carlos - EESC, Univ. de São Paulo - USP, SP*
(e-mail: daniel.carrijo@radicetech.com).
[**] *Radice Tecnologia, Atibaia, SP*
(e-mail: gabriel.gomes@radicetech.com)
[***] *Treetech Sistemas Digitais, Atibaia, SP*
(e-mail: christos.harissis@treetech.com.br)

Abstract: In digital signal processing applications, filtering is an essential task, especially when dealing with noise and malformations inherent to the signals. In this specific field, filtering can be done using several algorithms, both in integer (fixed-point) arithmetic and in real (floating-point) arithmetic. The objective of this article is to propose two innovative approaches to common filtering problems when calculating using integers, thus mitigating their consequences, which are often not perceived by designers or programmers, causing non expected errors the filtering process and, consequently, to the final result of the application. The analysis is performed on a *"Single Pole"* low-pass filter, enshrined in the literature and widely used in several fields of engineering, from Power Systems to Biomedical ones.

Resumo: Em aplicações de processamento digital de sinais a filtragem de sinais é uma tarefa essencial, principalmente quando lidamos com ruídos e malformações inerentes aos sinais observados. Neste campo específico, a filtragem pode ser feita utilizando-se de diversos algoritmos, tanto em aritmética inteira quando em aritmética real. O objetivo desse artigo é propor duas abordagens inovadoras para os problemas comuns de filtragem quando o cálculo é realizado com inteiros, mitigando assim as suas consequências, que muitas vezes não são percebidas pelos projetistas ou programadores podendo causar erros inesperados ao processo de filtragem, e consequentemente, ao resultado final da aplicação. A análise é realizada sobre um filtro *"Single Pole"* passa-baixa, consagrado pela literatura e muito utilizado em diversas áreas da engenharia, desde Sistemas de Potência à Biomédica.

Keywords: DSP; Low-Pass Filter; Fixed-Point Filter; Integer Arithmetic; IIR Filter.
Palavras-chaves: DSP; Filtro Passa-Baixa; Filtro com Inteiros; Aritmética Inteira; Filtro IIR.

1. INTRODUÇÃO

Com a popularização de dispositivos digitais portáteis, é cada vez mais importante o baixo consumo de energia aliado ao baixo custo de projeto. Existem outras aplicações onde os requisitos de energia ou custo não são fundamentais, mas temos requisitos muito exigentes de carga de processamento ou tempo de reposta. O requisito de processamento pode estar ligado, por exemplo, ao fato de o processador estar ocupado com uma grande quantidade de tarefas que não podem ser excluídas ou reduzidas. Por último, sistemas de tempo real baseados em microcontroladores tem requisitos exigentes de processamento, tempo de resposta e baixo custo.

Plataformas de processamento de sinais com aritmética inteira (ponto fixo ou *fixed-point*), são tipicamente mais eficientes em energia e menos dispendiosas do que as alternativas de aritmética real (ponto flutuante ou *floating-point*), graças a utilização de circuitos integrados mais simples, com baixo consumo de potência e ausência de FPU (*Floating-Point Unit* ou Unidade de Ponto Flutuante). Ademais, apesar da constante diminuição do custo de circuitos integrados com FPU, estes ainda são mais caros que os circuitos com somente aritmética inteira. Contudo, os filtros com aritmética inteira são geralmente mais difíceis de projetar.

O projeto dos filtros nasce naturalmente utilizando-se números reais, tanto para a definição dos seus coeficientes quanto para o processamento do sinal de entrada. Por isso, quando possível, é melhor implementar um filtro utilizando-se aritmética real. Contudo, existem rotinas que possuem exigentes restrições no tempo de execução, e a filtragem costuma estar entre elas. Considerando esse fator, a realização de aritmética real ocasiona em um grande

[⋆] A pesquisa e seus pesquisadores foram financeiramente suportados pela Treetech e Radice, em parceria com a SEL-EESC-USP.

comprometimento de processamento e a não garantia da entrega do resultado em tempo hábil. É sabido que a filtragem seria naturalmente mais veloz se realizada com inteiros.

O uso de inteiros resulta muitas vezes no desprezo da parte não inteira resultante das operações. Este procedimento causa um erro que influencia todas as operações subsequentes. Mesmo que o erro máximo resultante da imprecisão do cálculo inteiro seja tolerado, este erro ao longo da sequência de cálculo pode tornar o sistema de filtragem instável, podendo produzir resultados errados em sua saída. Ao se projetar filtros totalmente com inteiros (onde tanto os coeficientes do filtro quanto o processamento do sinal são realizados em inteiros), deve-se atentar alguns problemas inerentes, como o estouro das variáveis inteiras (*overflow*), a quantização dos coeficientes do filtro e do sinal de entrada, segundo Baranowski et al. (2016).

O erro de truncamento (*round-off)* com um sinal de entrada no limiar de detecção é um erro comumente negligenciado pelos projetistas, já que se poderia contar que a entrada nunca alcance este nível ou mecanismos de ganho possam ser inseridos para mitigar esta causa. Nesta forma, o erro de truncamento gera uma espécie de ruído (*round-off noise*), que se não tratado pode ocasionar deformação do sinal de saída, indo do imperceptível a inutilização total do sinal tratado pelo filtro, conforme apresentado por Prandoni and Vetterli (2008). Para os filtros classe IIR (*Infinite Impulse Response*), este problema é muito mais grave que para os classe FIR (*Finite Impulse Response*).

Na tentativa de resolver este e outros problemas de filtragem utilizando-se de inteiros, o objetivo de trabalho é pontuar esses erros que não costumam ser percebidos e propor soluções inovadoras para evitá-los, utilizando como exemplo um filtro *"single-pole"* passa-baixa, consagrado pela literatura e amplamente utilizado em diversas áreas, conforme mostrado em Smith et al. (1997), submetendo-o a dois sinais de entrada, sendo o primeiro uma onda retangular e o segundo uma onda senoidal com amplitudes variáveis, ambos com diferentes constantes de filtragem.

Neste trabalho serão abordados apenas filtros de primeira ordem. Apesar dos filtros de primeira ordem terem óbvias limitações no resultado do sinal filtrado, esta escolha se deve ao fato de que filtros de primeira ordem abrangem uma grande gama de problemas práticos com aplicação imediata, em especial quando velocidades altas de processamento são requeridas. Além disto, ordem superiores equivalentes podem ser obtidas com o cascateamento (iteração) do filtro de primeira ordem. Filtros de ordem superiores utilizando-se os métodos aqui apresentados serão apresentados e terão seu desempenho avaliado em trabalho futuro.

Nas próximas seções serão apresentados os problemas abordados e as soluções propostas. Na segunda seção, será apresentada uma resumida retrospectiva geral do projeto de filtros com inteiros, as suas implicações e desafios. Na terceira seção serão apresentados os fundamentos básicos de um filtro *single pole*, bem como sua implementação na forma recursiva. Na quarta seção serão apresentados os problemas que serão abordados e suas consequências em cada caso específico. Na quinta seção serão apresentadas as soluções para os problemas e sua forma de implementação. Na sexta seção serão realizadas a análise dos resultados. Por fim, na sétima e última seção será apresentada a conclusão deste trabalho.

2. DESAFIOS DO PROJETO DE FILTROS COM ARITMÉTICA INTEIRA

O projeto de filtros para processamento de sinais sempre foi um assunto muito debatido desde o início da eletrônica analógica, e passou por uma revolução com a chegada da era digital. Far-se-á uma breve abordagem do projeto destes filtros na era digital, com foco nos DSPs.

Usualmente, para projetar um filtro de ponto fixo, primeiramente é projetado um filtro de ponto flutuante, também conhecido como filtro de referência, que atenda e exceda aos requisitos finais. O excesso de margem garante uma conversão suave de uma representação de ponto flutuante para uma representação de ponto fixo. O próximo passo é modificar o filtro de ponto flutuante para acomodar as restrições de aritmética inteira enquanto ainda tenta atender aos requisitos. Sobre a definição dos melhores coeficientes inteiros para filtros IIR, este se apresenta como um problema matemático de difícil solução e não existe literatura significativa sobre o tema. Para os coeficientes de filtros FIR, um pouco menos complexos, existem artigos de referência, tais como Kodek (1980) e Kodek and Steiglitz (1981).

Filtros IIR com inteiros são comumente implementados em DSPs (*Digital Signal Processors*), FPGAs (*Field-Programmable Gate Arrays*) e ASICs (*Application-Specific Integrated Circuit*). Um filtro com inteiros usa aritmética inteira e é representado por uma equação com coeficientes inteiros. Se o acumulador e a saída do filtro IIR não tiverem bits suficientes para representar seus dados, ocorrerá estouro e distorcerá o sinal de saída.

O objetivo de um projeto de filtro IIR de ponto fixo é maximizar o desempenho do filtro e minimizar os efeitos da aritmética inteira Ott et al. (2019). Converter um filtro de ponto flutuante em ponto fixo pode alterar significativamente as características e o desempenho do filtro Bauer and Słowik (2017). A aritmética inteira tem diversos efeitos prejudiciais, dentre os quais, segundo Antoniou (2016), o que mais afeta o projeto dos filtros IIR é o da quantização indevida dos dados, resultando em estouros ou saída zero. Erros de arredondamento aritmético resultam de aritmética imprecisa, o que, por sua vez, reduz a precisão, devido ao acúmulo de vários erros sucessivamente ocorridos em um filtro. Além disso, a aritmética com inteiros pode levar a restrições na frequência de corte, restringindo os valores que ela pode assumir ao se construir o filtro idealmente previsto. Projetar e implementar filtros digitais IIR robustos, com a metodologia de números inteiros, é um constante desafio para projetistas.

3. FILTRO *SINGLE-POLE*

Conforme Oppenheim (1999), o filtro *single-pole* é um filtro IIR originado de uma digitalização por invariância da resposta ao impulso de filtros analógicos RC (passa-altas ou passa-baixas) de primeira ordem. Tomemos, como exemplo, o filtro de primeira ordem que tem seu circuito

analógico dado pela Figura 1, cuja função de transferência é dada por (1).

$$H(s) = \frac{1}{RCs + 1} \ , \tag{1}$$

que é escrita em termos da frequência de corte $fc = (2\,\pi\,R\,C)^{-1}$ como

$$H(s) = \frac{Y(s)}{X(s)} = \frac{2\,\pi\,f_c}{s + 2\,\pi\,f_c} \ , \tag{2}$$

cuja resposta contínua ao impulso é

$$h(t) = 2\,\pi\,f_c\,e^{-2\,\pi\,f_c\,t} \tag{3}$$

que corresponde a equação (4)

$$h[n] = 2\,\pi\,f_c\,e^{-2\,\pi\,n\,f_{cn}} \tag{4}$$

em um sistema discreto. A transformada Z da equação (4) é dada por:

$$H_K(z) = \frac{2\,\pi\,f_c}{1 - z^{-1}e^{-2\,\pi\,f_{cn}}} \ , \tag{5}$$

A equação (5) é também a função de transferência no domínio discreto, a menos de uma constante ($H(z) = K\,H_K(z)$). Deve-se portanto fazer a correção impondo que a resposta DC ($f = 0 \Leftrightarrow z = 1$) do filtro passa-baixas tenha ganho unitário $H(z)|_{z=1} = 1$, sendo assim finalmente temos a função de transferência do filtro *single-pole* passa-baixas:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - e^{-2\,\pi\,f_{cn}}}{1 - z^{-1}e^{-2\,\pi\,f_{cn}}} \tag{6}$$

e sua respectiva equação de diferenças:

$$y[n] = (1 - e^{-2\,\pi\,f_{cn}})\ x[n] + e^{-2\,\pi\,f_{cn}}\ y[n-1] \tag{7}$$

Podemos simplificar a equação (7) fazendo $(1 - e^{-2\,\pi\,f_{cn}}) = 1/K_f$, assim sendo a equação de diferenças se torna:

$$y[n] = \frac{x[n]}{K_f} + y[n-1] - \frac{y[n-1]}{K_f} \tag{8}$$

Onde K_f será chamada de constante ou ganho do filtro para fins de análise posterior. A partir da equação anterior, é possível ver que o filtro é implementado de forma recursiva. Desse modo, a saída atual depende de uma soma de pesos entre a saída anterior e a entrada atual. A resposta em frequência e a resposta de fase do filtro *single-pole* passa-baixa pode ser visualizada na figura 2. Desta análise, destacam-se dois pontos: Primeiro, que um filtro *single-pole* de um único estágio possui uma atenuação de 20db/dec. Segundo, seu ganho DC é unitário, imposto pela normalização realizada em (6).

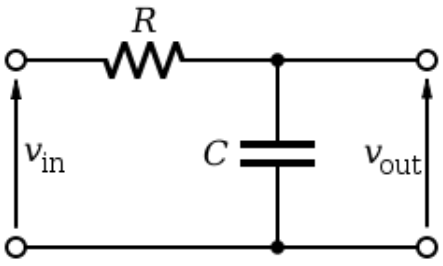

Figura 1. Circuito passa-baixas de primeira ordem

4. PROBLEMAS DA FILTRAGEM UTILIZANDO APENAS VARIÁVEIS DO TIPO INTEIRO

4.1 Problema 1: Numeradores menores que os denominadores

Para entendermos os problemas impostos pela aritmética baseada em inteiros, analisemos a equação (8). Os termos $\frac{x[n]}{K_f}$ e $\frac{y[n-1]}{K_f}$ resultarão em zero sempre que os valores de y[n-1] e x[n] forem menores que a constante do filtro (K_f). Para sinais com amplitude baixa ou amplitude comparada a amplitude do ganho do filtro, esse fato produzirá um sinal de saída totalmente diferente do esperado. Para entendermos tal problema, analisemos a figura 3. Podemos perceber que o ganho do filtro foi definido como $K_f = 3$, uma unidade maior que a amplitude do sinal. Devido ao fato de que o ganho do filtro é maior que a amplitude do sinal, a saída resultou em zero, uma vez que as parcelas fracionárias resultaram sempre em zero. Isso acontecerá para qualquer sinal sempre que o valor máximo da entrada for menor que o valor da constante do filtro utilizada. Se, por exemplo, utilizarmos uma entrada composta por um sinal senoidal e um ganho de filtro maior que o máximo valor da entrada, o mesmo resultado ocorrerá. Isso pode ser visto na figura 4.

Tal problema é crítico, uma vez que a dinâmica da entrada pode implicar muitas vezes em sinais de entrada com valores menores que a constante do filtro. A medida que a amplitude do sinal de entrada aumenta e sua ordem de grandeza se aproxima da ordem de grandeza da constante do filtro, esse problema diminui, fazendo com que a saída seja zero apenas quando a entrada e a realimentação do filtro forem menores que a constante do mesmo filtro. Na figura 5, podemos ver o agravamento desse problema para vários valores de constante do filtro. A medida que se aumenta a constante do filtro, mais a resposta em aritmética inteira desvia-se da resposta em aritmética real devido aos problemas do truncamento.

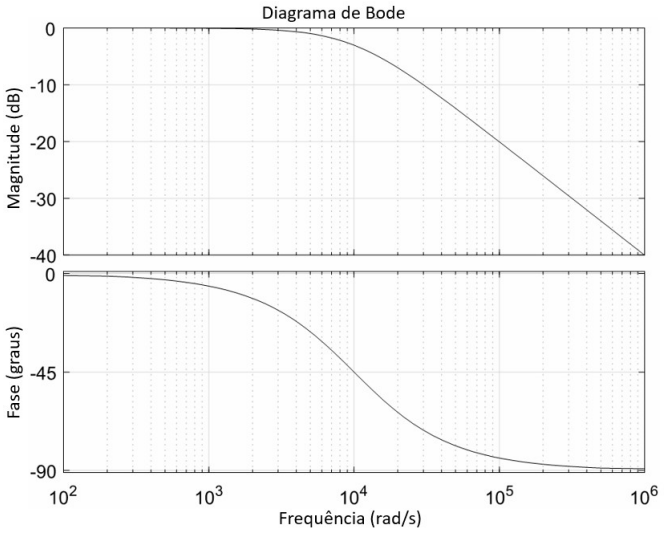

Figura 2. Resposta em frequência do filtro *single-pole* passa-baixa

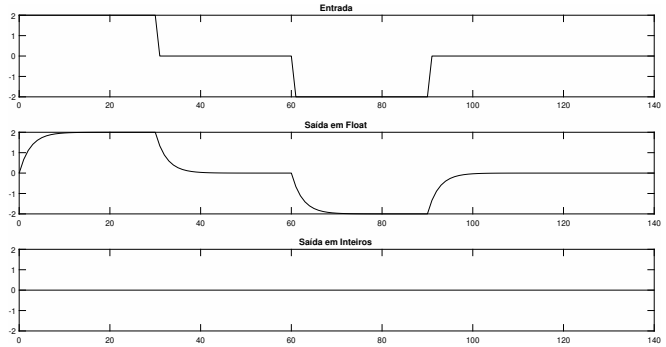

Figura 3. Numeradores maiores que os denominadores - Sinal Quadrado

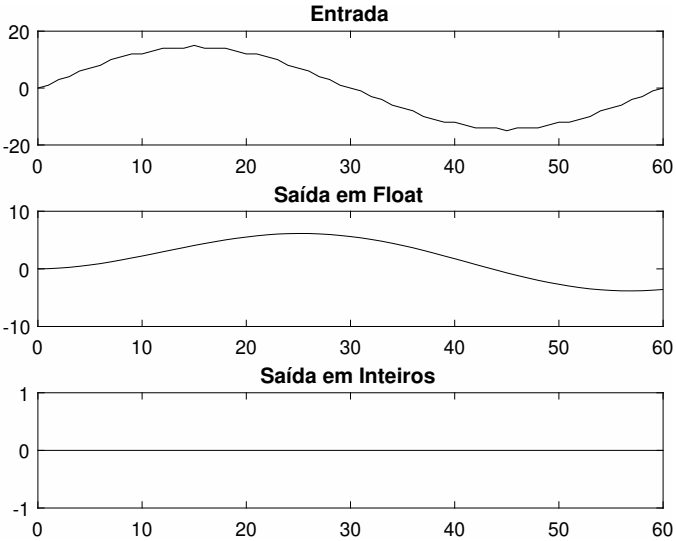

Figura 4. Numeradores maiores que os denominadores - Sinal Senoidal

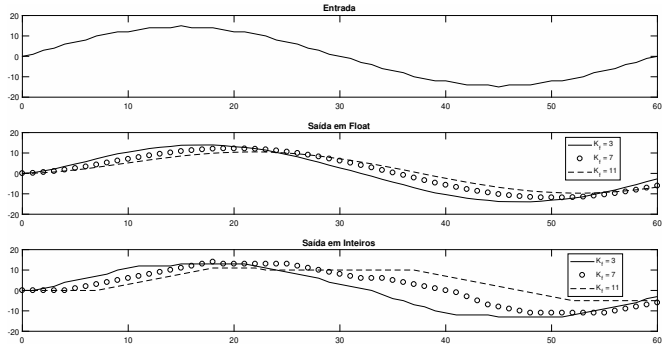

Figura 5. Numeradores maiores que os denominadores - Vários valores de constante do filtro

4.2 Problema 2: Ponto de estabilização do filtro diferente do ponto de estabilização da entrada

Um outro fato crítico causado pelo truncamento no cálculo apenas com inteiros é que o ponto de estabilização do filtro pode se tornar diferente do ponto de estabilização do sinal, que é o mesmo ponto de estabilização do resultado da filtragem em aritmética real. Isso acaba por inviabilizar muitas análises, como por exemplo a análise de *"Zero-Crossing"*. Um exemplo de estabilização em um ponto diferente pode ser verificado na figura 6. Para realização

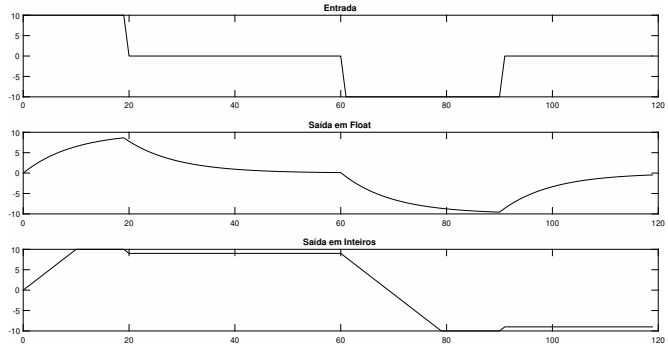

Figura 6. Problema de não estabilização no ponto adequado

da filtragem, o valor de K_f utilizado foi igual a "dois". Devido aos problemas do truncamento na forma de onda de saída do filtro com inteiros, é praticamente impossível identificar a passagem por zero do sinal filtrado. Além disso o resultado ficou bem distante da filtragem calculada em *floating-point*, perdendo boa parte das características de forma.

5. SOLUÇÕES PARA PROBLEMAS DE TRUNCAMENTO

5.1 Método 1: Ganho Quadrático

Consideremos algumas observações feitas na seção anterior: Quanto mais o valor máximo da entrada se aproxima ou ultrapassa o ganho do filtro, menos o efeito da truncamento é percebido. Partindo desse princípio, podemos multiplicar a entrada pela constante K_f do filtro, de modo que para qualquer valor maior ou igual a 1, a entrada dividida pelo ganho do filtro sempre resultará em um valor diferente de zero. Assim sendo, a equação (8) pode ser reescrita como:

$$y[n] = x[n] + y[n-1] - \frac{y[n-1]}{K_f} \ . \qquad (9)$$

Com essa multiplicação, conseguimos resolver o problema da divisão da entrada resultar em zero, o que faz com que a saída do filtro não seja nula (para uma entrada diferente de zero), conforme mostrado nas figuras 4 e 5. Todavia, podemos perceber que ainda teremos o problema da estabilização em um valor diferente do esperado, uma vez que a realimentação dividida pelo ganho do filtro pode implicar em um valor constante. Assim sendo, para evitar esse problema, podemos multiplicar novamente a entrada por K_f, de modo a evitar que a realimentação estabilize em um valor constante. De tal modo, a equação (9) se tornará:

$$y[n] = K_f x[n] + y[n-1] - \frac{y[n-1]}{K_f} \ . \qquad (10)$$

Para analisarmos a validade do método, consideremos as figuras 4 e 6 agora aplicadas com a nova equação resultando nas figuras 7 e 8 Como pode ser visto nas simulações, ambos os problemas foram resolvidos de fato. Porém, um outro problema foi inserido. O ganho DC do filtro deixou de ser unitário e passou a ser K_f^2 vezes o ganho DC do filtro padrão. Outro problema criado é que se a amplitude do sinal e o ganho forem muito grandes, a filtragem poderá resultar em um *overflow* da variável de

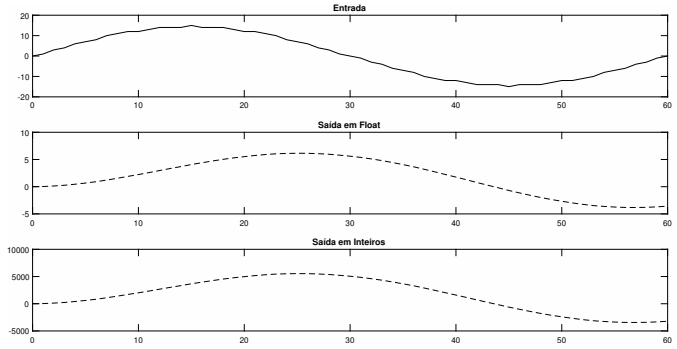

Figura 7. Problema de saída nula resolvida pelo método

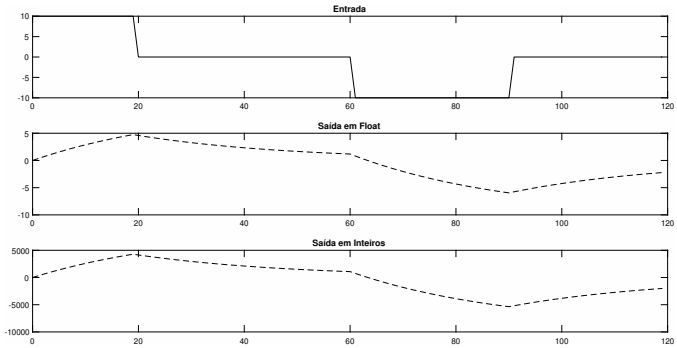

Figura 8. Problema de não estabilização no ponto adequado resolvido pelo método

saída do filtro. Por exemplo, se considerarmos a constante do filtro $K_f = 30$ e o valor máximo da amplitude igual a "1000" (o que estaria de acordo considerando uma escala completa de um conversor AD de 10 bits) poderíamos esperar valores da ordem de centenas de milhares. Se considerarmos que a variável de saída do filtro poderia ser uma variável do tipo int16 (inteiro de 16 bits), cujo a máxima representação seria 65535 em uma operação não sinalizada, essa variável teria sofrido um *overflow*. Se o objetivo é encontrar o ponto de passagem por zero de um sinal ou algo diretamente ligado a forma de onda em si e não o valor de sua amplitude, o problema do ganho DC não unitário não influenciará. Todavia, se a amplitude do sinal for um fator importante, será necessário a realização de um outro ajuste. Após o cálculo de cada amostra de saída do filtro, o resultado deverá ser dividido por K_f^2 e estocado em outra variável que sera a saída de ganho DC unitário. A filtragem então será composta por duas equações, a equação (10) adicionada da equação da saída com ganho DC unitário, que será:

$$y_{DC}[n] = \frac{y[n]}{K_f^2} \tag{11}$$

É importante salientar que o valor realimentado na equação de filtragem continua sendo $y[n-1]$ e não $y_{DC}[n-1]$. Como o truncamento é mais uma vez realizado, a saída ficará quantizada de uma maneira mais notória, uma vez que a faixa de amplitude da saída diminuirá e por consequência o número de *"quantums"* que a compõe, conforme pode ser vista na figura 9. Podemos verificar que a forma de onda deformou-se, todavia não existe mais o problema do ganho DC unitário. Ainda possuímos o problema do *overflow* da primeira saída do filtro. Portanto, ao utilizar esse método é interessante ter cuidado com os

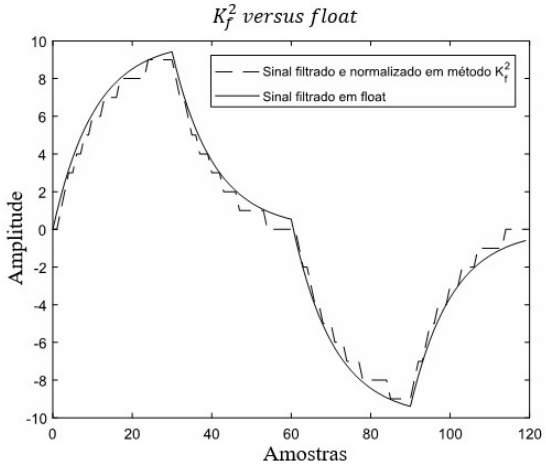

Figura 9. Saída do filtro Ganho Quadrático normalizado e do filtro em *floating-point*

valores máximos aceitos pela variável de saída. Para o caso exemplificado anteriormente, o ideal seria a utilização de uma variável de saída do tipo int32 (inteiro de 32 bits). O algoritmo para implementação do método é direto, basta multiplicar a entrada por K_f^2 e realizar a filtragem normalmente. Caso deseja-se um valor normalizado, dividir a saída do filtro pela constante do filtro K_f^2 e estocar em outra variável. Na figura 10 é representado o fluxograma do algoritmo de implementação do método.

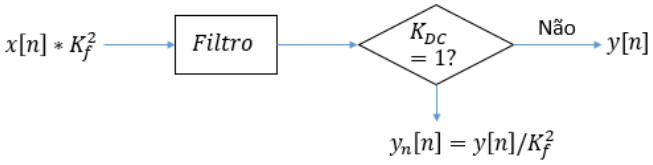

Figura 10. Fluxograma Algoritmo Ganho Quadrático

5.2 Método 2: "Soma Triangular"

O segundo método a ser apresentado pode ser representado pela equação abaixo:

$$y[n] = y[n-1] - \frac{y[n-1] - x[n] + K_f K_c(-1)^n}{K_f \qquad +K_c(-1)^n} \tag{12}$$

O método "Soma Triangular" consiste na soma de um "ganho" multiplicado por (-1) elevado ao índice da amostra, o que implica em somas e subtrações de $K_f K_c$. Se analisarmos a equação (12) do ponto de vista de uma aritmética operada com ponto flutuante, essa equação é idêntica a equação (8). Todavia, do ponto de vista de inteiros, a equação é bem diferente. O fato de se somar um valor antes de operar a divisão central da equação faz com que essa divisão tenda a não convergir para zero e não estabilizar em um valor diferente do esperado. Para que esse método de fato tenha efeito, é necessário que o produto entre o "Soma Triangular" K_c e o ganho do filtro K_f seja igual ou maior ao máximo valor da entrada, de modo que o numerador dessa divisão tenda a não ser, em modulo, um número menor que K_f. A saída devido ao truncamento também é quantizada, assim como no caso do

método anterior. As Figure 11, 12 mostram uma aplicação do método do "Soma Triangular" para as duas entradas: Pela figura 12, é possível perceber que a resposta do

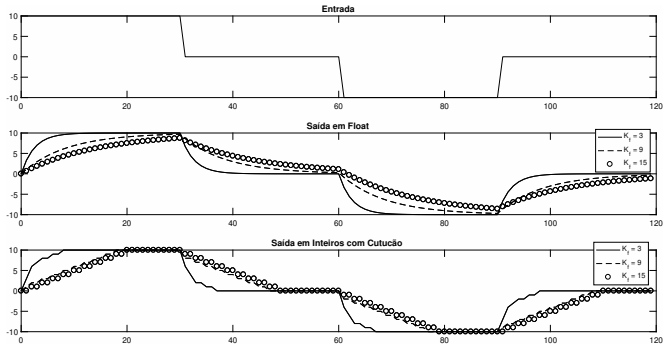

Figura 11. Resposta do filtro com Soma Triangular para valores diferentes de K_f com entrada retangular

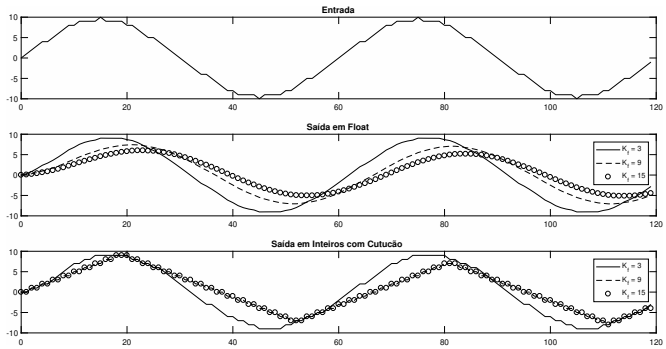

Figura 12. Resposta do filtro com Soma Triangular para valores diferentes de K_f com entrada senoidal

filtro com Soma Triangular para $K_f = 9$ e $K_f = 15$ é a mesma. Isso ocorre porque a medida que o ganho do filtro ultrapassa o máximo valor da entrada, a saída pelo método do "Soma Triangular" não sofre alteração significativa. Para resolver o problema da estabilização da saída acima exposto, podemos seguir de um modo semelhante ao feito no método do Ganho Quadrático, multiplicando $x[n]$ por K_f e depois dividindo a saída atual por K_f estocando em uma variável de saída diferente. Assim obteremos uma saída com ganho DC unitário e outra saída com ganho DC diferente de unitário, e um filtro composto de duas equações, conforme visto no método anterior. Assim sendo, as duas equações serão:

$$y[n] = y[n-1] - \frac{y[n-1] + K_f K_c(-1)^n}{K_f} + x[n] + K_c(-1)^n \quad (13)$$

$$y_{DC}[n] = \frac{y[n]}{K_f} \quad (14)$$

Assim, a resposta com ganho DC unitário do filtro com Soma Triangular se transformará conforme a figura 13. O algoritmo para implementação do mesmo é bem semelhante a do método do Ganho Quadrático, com mudanças na equação do filtro e no fato de que agora a entrada não será mais multiplicada por K_f^2, uma vez que a equação do filtro foi alterada, e a saída será dividida por K_f e não mais por K_f^2 conforme pode ser visto no fluxograma da figura 14

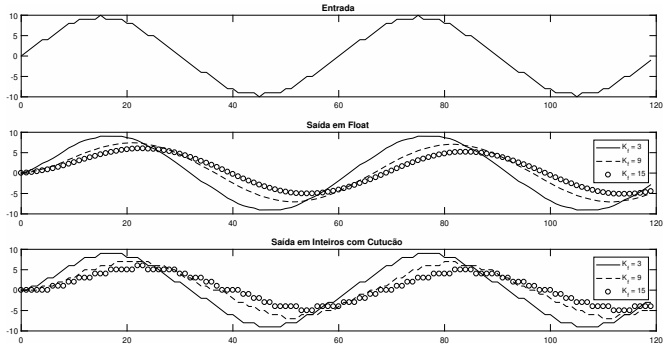

Figura 13. Resposta do filtro com Soma Triangular para valores diferentes de K_f com entrada senoidal normalizado por K_f

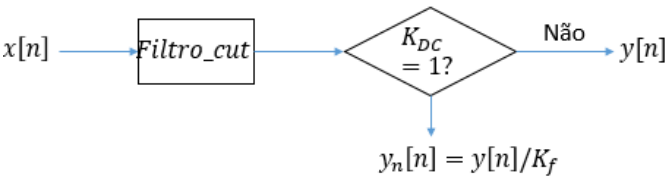

Figura 14. Fluxograma Algoritmo Soma Triangular

5.3 Método 3: Resto Cumulativo

O ultimo método a ser apresentado é chamado de resto cumulativo, que consiste em acumular o resto das divisões e dividi-los novamente assim que eles ultrapassarem o valor do ganho do filtro, e então somar ou subtrair "um" da próxima saída do filtro. Desse modo, diminui-se o erro ao longo da filtragem, porque o resto não é desprezado. A equação do filtro para o resto acumulativo pode ser visualizado abaixo:

$$y_{[}n] = y[n-1] - \frac{y[n-1] - x[n]}{K_f} + \frac{Acc}{K_f} \quad (15)$$

Onde o valor Acumulado é igual a:

$$Acc = rem\left(y[n-1] - x[n], K_f\right) + rem\left(Acc_a, K_f\right) \quad (16)$$

Onde "rem(a,b)" é o resto da divisão de "a por b" e "Acc_a" é o resto acumulado da divisão anterior. Nas figura abaixo é possível ver a resposta desse método para dois tipos de entrada e três ganhos diferentes de filtro: O algoritmo para implementação do método pode ser visualizado no fluxograma da figura 17. Os passos desde algorítimo iniciam-se com o calculo do valor acumulado atual, que é

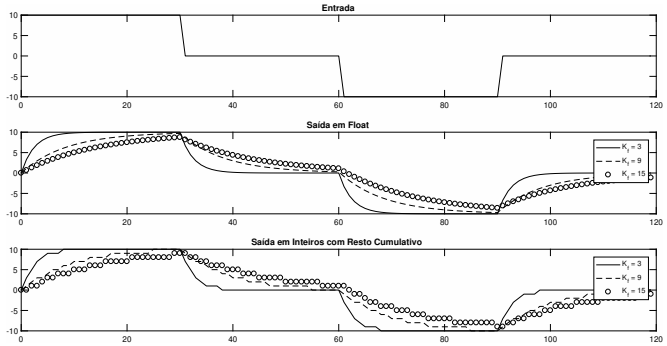

Figura 15. Resposta do filtro com Resto Acumulativo para valores diferentes de K_f com entrada retangular

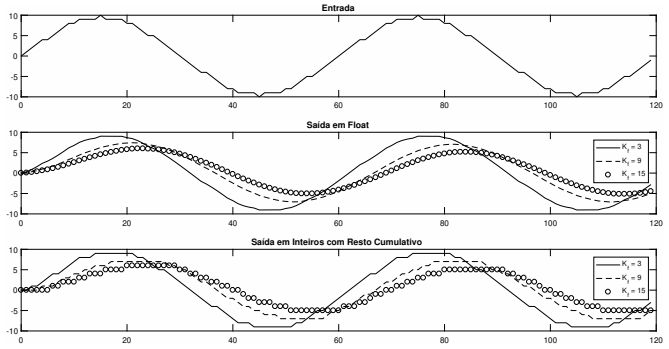

Figura 16. Resposta do filtro com Resto Acumulativo para valores diferentes de K_f com entrada senoidal

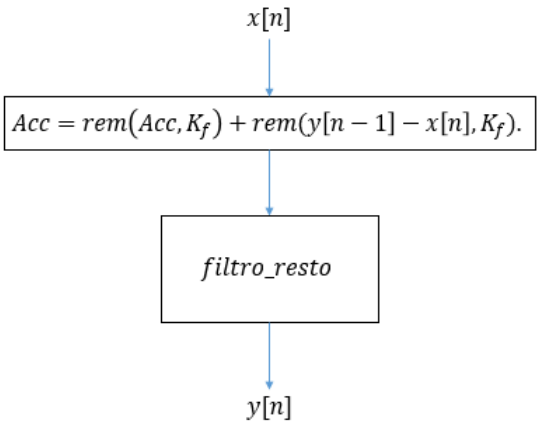

Figura 17. Fluxograma Algoritmo Resto Acumulativo

composto pelo resto da divisão da realimentação subtraída da entrada atual pela constante do filtro, somada ao resto da divisão do acumulado atual pela constante do mesmo filtro. Após isso, proceder a filtragem conforme a equação (16).

6. ANÁLISE DE RESULTADOS E COMPARAÇÕES

Para analisar os resultados e realizar comparações entre os métodos, será utilizado como referência de comparação o cálculo em *floating-point*, que é a maneira ideal de realizar as filtragens do ponto de vista matemático, e os demais cálculos serão então comparados a este por meio do Coeficiente de Correlação de Pearson, mostrado na equação abaixo:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \qquad (17)$$

Onde $\bar{x}$ é a média da primeira base de dados e $\bar{y}$ é a média da segunda base de dados as quais estão sendo comparadas. Esse coeficiente indica o quanto as variações nos dois bancos de dados estão relacionados através de uma regressão linear. Quanto mais próximo de "1", mais correlacionados estão os dados e por consequência mais suas formas se parecem. Para as análises de comparação, serão utilizadas as duas entradas dos exemplos anteriores e três valores de ganho do filtro. Os três métodos propostos e o modo sem modificação serão comparados a resposta do filtro calculado em *floating-point*. As figuras 18 e 19, em conjunto com a tabela 1 apresentam essas comparações.

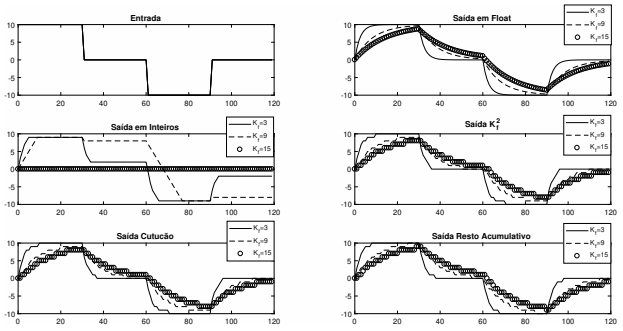

Figura 18. Resposta de todos os métodos para uma entrada retangular para $K_f = 3$, $K_f = 9$, $K_f = 15$

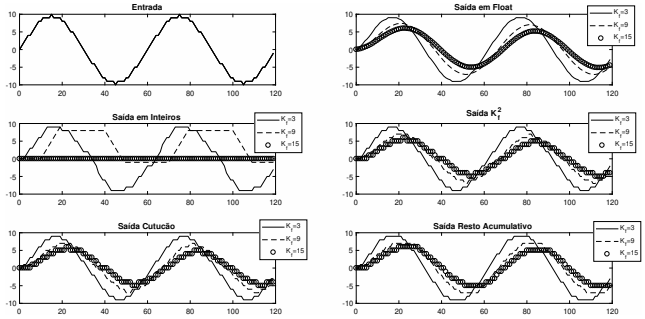

Figura 19. Resposta de todos os métodos para uma entrada senoidal para $K_f = 3$, $K_f = 9$, $K_f = 15$

Tabela 1. Coeficientes de Correlação:
Método X Ganho do Filtro X Tipo de Entrada

Método	$K_f = 3$	$K_f = 9$	$K_f = 15$
Inteiros	0,9800|0,9928	0,8038|0,6323	0,0000|0,0000
Ganho Quadrático	0,9993|0,9986	0,9977|0,9967	0,9974|0,9945
Soma Triangular	0,9993|0,9988	0,9976|0,9969	0,9975|0,9945
Resto Acc	0,9960|0,9813	0,9859|0,9711	0,9725|0,9721

Através das figuras 18 e 19, em conjunto com a tabela 1, podemos perceber que o filtro *single-pole* utilizando inteiros não apresenta uma boa resposta a medida que o ganho do filtro cresce em relação a amplitude do sinal. Para o primeiro caso, onde o sinal retangular esteve submetido a um filtro de ganho igual a 3, sendo a amplitude do sinal maior que três vezes a amplitude do filtro, a resposta com inteiros apresentou um índice de correlação razoável, porém para aplicações de alta precisão o mesmo já não serviria uma vez que seu índice de correlação foi igual a 0,98, ou seja, um erro de 2% em relação ao sinal filtrado utilizando *floating-point*. A medida que a constante do filtro aumenta em relação ao comprimento do sinal, ou seja, um filtro com frequência de corte mais baixa, a resposta do mesmo filtro piora. Para um ganho igual a 9, ainda inferior ao máximo da entrada, temos um coeficiente de correlação equivalente a 0,8038 para a entrada retangular e 0,6323 para a entrada senoidal, correspondendo a erros de $19,62\%$ e $36,77\%$ respectivamente. Erros de tal magnitude são inadmissíveis para a maioria das aplicações. Quando consideramos um ganho do filtro maior que a máxima amplitude do sinal, a saída se torna literalmente zero, como mostrado na seção 2.

Analisando os resultados obtidos pelo método "Ganho Quadrático", podemos perceber que o mesmo apresenta uma ótima resposta para todos os ganhos analisados. Além

disso, o mesmo estabiliza no mesmo ponto de estabilização do sinal filtrado em *floating-point*. O maior erro apresentado pelo mesmo método é para constante do filtro igual a 15, mesmo assim obtendo um erro menor que $0,6\%$.

O resultado obtido pelo método do "Soma Triangular" é levemente superior ao método do "Ganho Quadrático". Sua resposta tanto para a entrada retangular quanto para a entrada senoidal teve um erro menor que a resposta do Ganho Quadrático, enquanto os erros para entrada senoidal foram os mesmos. Outro ponto interessante é que o erro médio desse método é $0,06\%$ menor que o erro do método anterior.

Por fim, o método do resto acumulativo obteve um resultado intermediário quando comparado com o Ganho Quadrático e "Soma Triangular", e sua implementação é um pouco mais complexa que a dos demais. Todavia, seus resultados foram bem melhores que o cálculo tradicional com inteiros, podendo ser também uma substituição viável.

7. CONCLUSÕES

Projetar filtros IIR somente com inteiros é um desafio ainda sem solução final, já que não existe uma técnica universal de projeto que contemple todos os requisitos para as diversas aplicações. O uso de aritmética inteira no projeto de filtros IIR incorre em riscos adicionais em relação ao seu uso em aritmética real. Um destes riscos, o do erro de truncamento (*round-off)* caso o sinal de entrada esteja no limiar de detecção, foi abordado neste trabalho e solucionado com proposta de dois novos métodos, o da "Soma Triangular" e o do "Resto Acumulativo". Os métodos desenvolvidos mostraram-se eficazes, de fácil codificação e com baixo erro local ou global.

Comparando-se o cálculo tradicional com inteiros (sem o uso de nenhuma técnica de tratamento) e método do "Ganho Quadrático" (que não é nada mais que um simples contorno ao problema real) com os métodos propostos neste trabalho, é possível perceber que ambos apresentam resultados extremamente satisfatórios. O método da "Soma Triangular" apresenta o menor erro dentre todos, quando comparado com o filtro ideal (aritmética real) (0,55%). O método do "Resto Acumulativo" é mais intuitivo, pois mostra o erro sendo compensado em cada iteração, porém apresenta um desvio maior em relação ao filtro ideal (2,79%). O método do "Ganho Quadrático" é mais simples de ser implementado e é minimamente menos preciso que o da "Soma Triangular" ($0,06\%$), mas apresenta o risco de *overflow* caso a entrada seja desconhecida. Além deste, ele apresenta uma desvantagem adicional já que ao se multiplicar a entrada pelo ganho ao quadrado, os números tornam-se grandes para os cálculos, resultando em um processamento mais lento e custoso.

Os métodos propostos neste estudo ("Soma Triangular" e "Resto Acumulativo"), por não sofrerem as influências numéricas do uso de inteiros, podem ser utilizados de forma a prover uma maneira precisa para calcular a passagem por zero (saída não normalizada) ou para calcular a amplitude real do sinal filtrado (saída normalizada).

Neste trabalho foram abordados apenas implementações de primeira ordem (filtros de primeira ordem) dos métodos propostos. Filtros de ordens superiores serão abordados em trabalho futuro. Esta escolha se deve a dimensão do tema abordado, não comprometendo o conceito e avaliação dos métodos propostos. Ademais, o cascateamento de filtros de primeira ordem para que resultados similares aos de ordem superior sejam obtidos é muito comum em aplicações de *DSPs* que exigem grande velocidade de resposta, tornando os resultados aqui apresentados de imediata aplicação prática para o leitor. Exemplos práticos de utilização - inclusive a análise do caso que motivou este desenvolvimento - serão abordados em trabalho futuro.

8. AGRADECIMENTOS

Os autores agradecem a **Radice** e a **Treetech** pelo estímulo à pesquisa, que vem desde o fomento a ideias inovadoras, passando pelo ambiente empresarial que auxilia na superação dos desafios e chega ao final com o suporte integral a cadeia de PD&I, além do Prof. Rogério de Andrade Flauzino, do **Departamento de Engenharia Elétrica e de Computação - SEL, da Escola de Engenharia de São Carlos - EESC, Universidade de São Paulo - USP**, cuja participação foi essencial para a consistência dos resultados aqui apresentados.

9. ORCID IDS DOS AUTORES

Daniel Carrijo Polonio Araujo; Gabriel de Souza Pereira Gomes; Christos Aristóteles Harissis; Rogério Andrade Flauzino.

REFERÊNCIAS

Antoniou, A. (2016). *Digital signal processing*. McGraw-Hill.

Baranowski, J., Bauer, W., Zagórowska, M., and Piątek, P. (2016). On digital realizations of non-integer order filters. *Circuits, Systems, and Signal Processing*, 35(6), 2083–2107.

Bauer, W. and Słowik, W. (2017). Comparison fixed-point and floating-point implementation of noninteger filter of stm microcontroller. In *Conference on Non-integer Order Calculus and Its Applications*, 126–134. Springer.

Kodek, D. (1980). Design of optimal finite wordlength fir digital filters using integer programming techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(3), 304–308.

Kodek, D. and Steiglitz, K. (1981). Comparison of optimal and local search methods for designing finite wordlength fir digital filters. *IEEE Transactions on Circuits and systems*, 28(1), 28–32.

Oppenheim, A.V. (1999). *Discrete-time signal processing*. Pearson Education India.

Ott, G., Costa, E.A., Almeida, S.J., and Fonseca, M.B. (2019). Iir filter architectures with truncation error feedback for ecg signal processing. *Circuits, Systems, and Signal Processing*, 38(1), 329–355.

Prandoni, P. and Vetterli, M. (2008). *Signal processing for communications*. EPFL press.

Smith, S.W. et al. (1997). The scientist and engineer's guide to digital signal processing. *C.T.P.*