Abstract

[7]

Dedication

To mum and dad

Declaration

I declare that..

Acknowledgements

I want to thank...

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Revisão bibliográfica

Neste capítulo serão abordadas as técnicas anteriormente mencionadas, que são o foco deste trabalho, PLL (Phase-Locked Loop), análise de prony, ADALINE e RLS (Recursive Least Mean Squares), bem como as convencionais DFT e STFT. É importante ressaltar que esta será apenas uma abordagem superficial, não entrando, portanto, com profundidade nos assuntos tratados.

1.1 Análise espectral

Análise espectral considera o problema de encontrar a energia de um sinal, finito, distribuída em função de frequência, o que pode ser feito com métodos paramétricos ou não paramétricos. Métodos paramétricos assumem conhecimento do modelo com o qual se gerou o sinal em análise. Por exemplo, em alguns capítulos, como o de análise de Prony, é assumido que o sinal é composto unicamente por un número conhecido de exponenciais complexas amortecidas. Para solucionar um problema de estimação paramétrica, o que se deve fazer é encontrar os parâmetros supostos. Por outro lado, métodos não paramétricos se baseiam unicamente em transformações do domínio temporal para o domínio da frequência, utilizando filtros que nos dão a energia do sinal contida em uma determinada faixa do espectro, caso da DFT. Quando temos uma boa estimação de nosso modelo, os métodos paramétricos tendem a dar melhores resultados, enquanto que quando pouco se sabe sobre o sinal analisado, ou se o mesmo não está bem modelado, os não paramétricos podem ser a melhor opção.

Algumas vezes é conveniente tratar os sinais de maneira determinística, e o será feito neste trabalho, entretanto, como abordagem mais geral, se pode tratá-los com enfoque probabilístico, isto é: admitir que não se pode de maneira alguma prever os valores assumidos por um sinal ao longo do tempo, mas se pode estimar suas características por meios estatísticos. É de se notar também que embora sinais reais sejam o caso mais comum na maioria das aplicações e em nosso caso são praticamente regra, não necessariamente o tratamento de sinais complexos é muito mais complicado. Apenas por uma questão de praticidade e para deixar o trabalho mais conciso, não serão demonstrados os teoremas utilizados para entradas complexas quando não for necessário.[11]

1.2 Densidade espectral de energia de um sinal determinístico

Seja $y_c(t)$ um sinal contínuo no tempo e $y[n],\ n = 0, \pm1, \pm2, ...$ um sinal discreto tal que $y[n] = y_c(nT_s)$. Considerando que $y[n]$ tem energia finita. Pode-se dizer então que a DTFT de y está definida como:

$$Y(w) = \sum_{n=-\infty}^{+\infty} y[n]e^{-jwn} \tag{1.1}$$

Onde w é denominada frequência angular medido em radianos/segundo, e pode ser relacionada com seu valor físico da seguinte maneira $\bar{w} = wT_s$. A densidade espectral de energia do sinal $y(t)$ pode ser dada então por:

$$S(w) = |Y(w)|^2 \tag{1.2}$$

Pode-se interpretar este resultado de algumas formas, mas uma bem simples é lembrar que a DTFT de uma cossenoide é igual a soma de duas deltas refletidas no espectro de frequência:

$$\sum_{n=-\infty}^{+\infty} cos(2\pi f\ n)e^{-jwn} = \pi\delta_w(2\pi f) + \pi\delta_w(-2\pi f)\ \ w\ \in [0, 2\pi] \tag{1.3}$$

Uma relação muito parecida se estabelece no caso de uma senoide. Desta maneira, se imaginamos $yc(t)$ composta unicamente de uma soma de senoides, seu espectro fica muito bem explicitado com a equação XX.

Uma outra relação importante se dá utilizando a autocorrelação de $y[n]$:

$$\rho_y[k] = \sum_{n=-\infty}^{+\infty} y[n]y^*[n-k] \tag{1.4}$$

$$\begin{aligned}\sum_{k=-\infty}^{+\infty} \rho_y[k]e^{-jwk} &= \sum_{n=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} y[n]y^*[n-k]e^{-jwk} = \\ &\sum_{n=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} y[n]y^*[n-k]e^{-jwn}e^{jw(n-k)} = \\ \left[\sum_{n=-\infty}^{+\infty} y[n]e^{-jwn}\right]&\left[\sum_{s=-\infty}^{+\infty} y[s]e^{-jws}\right]^* = Y(w)Y(w)^*\end{aligned} \tag{1.5}$$

Vê-se então que a DTFT da autocorrelação de um sinal é igual sua densidade espectral de energia.

1.2.1 Densidade espectral de potência

A maioria dos sinais com os quais se trabalha não têm um modelo puramente determinístico que os reproduza, não se sabe o valor exato que tomarão no futuro, e tampouco se pode estender seu valor até o infinito, ou analisar infinitas amostras do mesmo. Para os casos reais, é mais conveniente lidar com sequencias aleatórias

ao longo do tempo, em que cada realização da sequência tem associada uma distribuição de probabilidade (até o momento de sua ocorrência, quando esta colapsa em algum valor). Para a maioria dos casos, consideraremos os sinais como processos estacionários em sentido amplo [8]. Para este trato aleatório, não é regra que nossos sinais tenham energia finita, mas é possível ao invés disso, estimar sua densidade de potência.

Assume-se agora que $y[n]$ é uma sequência de variáveis aleatórias, com o mesmo domínio da sequência anterior, e que $E[y[n]] = 0$ para todo n. Tendo todas as variáveis aleatórias média nula. A função de correlação de $y[n]$ está definida como:

$$r_y[k] = E[y[n]\, y[n-k]^*] \tag{1.6}$$

Definimos então a densidade espectral de potência como:

$$\phi(w) = \sum_{k=-\infty}^{+\infty} r_y[k] e^{-jwk} \tag{1.7}$$

Uma segunda definição para a PSD (*Power Spectrum Density*) pode ser dada como a DTFT do sinal com o número amostras tendendo ao infinito da seguinte forma:

$$\phi(w) = \lim_{N\to\infty} \frac{1}{N} \left| \sum_{k=0}^{N-1} y[k] e^{-jwk} \right|^2 \tag{1.8}$$

Ambas definições são equivalentes. Será exposto mais adiante que uma das formas principais de se calcular a PSD é por meio de estimações da função de autocorrelação do sinal em questão.

1.2.2 Periodograma e correlograma

Uma forma bastante simples de estimar a PSD seria:

$$\hat{\phi_p}(w) = \frac{1}{N} \left| \sum_{k=0}^{N-1} y[k] e^{-jwk} \right|^2 \tag{1.9}$$

A qual é chamada periodograma. Outra forma poderia ser:

$$\hat{\phi_c}(w) = \sum_{k=-(N-1)}^{N-1} r_y[k] e^{-jwk} \tag{1.10}$$

A qual é chamada correlograma. Dois estimadores para a autocorrelação serão apresentados abaixo, considerando $n = 0, 1, 2, ..., N-1$:

$$\hat{r}_y(k) = \frac{1}{N-k} \sum_{n=k}^{N-1} y[n]\, y^*[n-k] \tag{1.11}$$

$$\hat{r}_y(k) = \frac{1}{N} \sum_{n=k}^{N-1} y[n]\, y^*[n-k] \tag{1.12}$$

O primeiro estimador é chamado estimador padrão não viesado, e o segundo é um estimador viesado. Lembrando que é assim chamado (viesado) um estimador cuja esperança não é o que visa estimar. Fato é também que $\hat{\phi}_p$ é igual a $\hat{\phi}_p$ se $\hat{r_y}$ for estimada com estimador viesado.

Muitas vezes o estimador em XX é preterido ao de XX, porque para muitos sinais a autocorrelação para valores grandes de k (valores considerados longe do 0) é muito baixa, e portanto não ajudaria fazer a correção do fator de normalização. Entretanto esse pode não ser bem o caso, porque os sinais muitas vezes são considerados senoides puras, as quais apresentam correlação em trechos periódicos. Em realidade, não serão discutidos estimadores de correlação. A seguir serão analisados métodos relacionados a DFT. As vantagens e desvantagens destes métodos todos podem ser vistos à luz de processos estocásticos, ou pela análise das características do método em si. Tentaremos mostrar um pouco de ambos, os utilizando da maneira que for mais conveniente.

1.3 DFT

A transformada discreta de Fourier (DFT) de uma sequência discreta x[n] de tamanho N é definida como[5]:

$$X[k] = \sum_{n=0}^{n=N-1} x[n]e^{(-2jnk\pi/N)} \ , \ k = 0, 1, 2, ..., N-1 \tag{1.13}$$

Pode-se dizer que a sequência X[k] é a DFT de x[n]. As duas sequências tem o mesmo tamanho, e X[k] representa o mapeamento das frequências de x[n] supondo estas estacionárias e com período igual ao analisado. Sendo Ts o tempo de amostragem da sequência x[n], temos a seguinte relação:

$$R_s = \frac{1}{T_s N} \tag{1.14}$$

Onde Rs é igual a resolução de X[k]. O mapeamento é dado como:

$$f_k = R_s k \ , \ k = 0, 1, 2, ..., N/2 \tag{1.15}$$

Percebe-se que a DFT age como uma série de Fourier, onde valores absolutos de X[k] são vistos como a amplitude de uma determinada frequência múltipla da componente fundamental f1, que é igual a resolução Rs. Para um sinal contendo apenas harmônicos de f1, e eventualmente algum valor médio, é possível extrair perfeitamente seus valores de amplitude e fase. Entretanto, para um sinal que possui componentes inter-harmônicos, não é se pode fazer o mesmo. Neste caso ocorre o chamado *espalhamento*. Já se pode notar algumas deficiências da DFT. Esta também apresenta problemas caso não seja amostrado um período exato do sinal analisado, ou um múltiplo inteiro de um período, podendo a DFT levar a crer, por exemplo, que existe um valor médio no sinal, e outros conteúdos equivocados. Este fenômeno é denominado *spectral leackage*[5].

Como a DFT pressupõe um sinal estacionário, ela não é adequada para análise de sinais variantes no tempo. Para tanto é possível utilizar de uma DFT de janela deslizante. Para uma janela de tamanho N:

$$X[k,m] = \sum_{n=0}^{N-1} x[m-n]e^{(-2jnk\pi/N)} \ , \ k = 0,1,2,...,N-1 \tag{1.16}$$

Desta forma, para cada m¿N, tem-se uma janela de tamanho N onde será analisado o sinal. Existem algoritmos mais eficazes para efetuar este tipo de cálculo de forma recursiva, sem precisar calcular toda a DFT como se estivéssemos diante de uma janela completamente nova:

$$X[k,m] = C\{X[k,m-1]e^{j2\pi k/N} + (x[m] - x[m-N])e^{j2\pi k/N}\} \ , \ k = 1,2,...,N/2 \tag{1.17}$$

C=1/N para k=N/2 e igual a 2/N para os demais valores. O valor de k foi restringido levando-se em conta a simetria dos sinais reais.

Pode-se usar diferentes tipos de janelas além da anterior, que é uma janela quadrada, onde todos os termos da sequência tem igual peso no cálculo da DFT. O uso de outras janelas ajuda na amenização do *leackage*. Algumas janelas estão expostas abaixo.

Janela triangular:

$$w_{tri}(n) = 1 - \frac{|2n - N + 1|}{N-1} \ , \ 0 \leq n \leq N-1 \tag{1.18}$$

Janela de Hamming:

$$w_{hm}(n) = 0.54 - 0.46cos(\frac{2\pi n}{N-1}) \ , \ 0 \leq n \leq N-1 \tag{1.19}$$

O uso deste tipo de janelamento atenua eventuais descontinuidades nas extremidades do sinal, auxiliando na medição dos parâmetros. Aplicar uma janela ao sinal significa deslizar a sequência formada por w por x[n], multiplicando termo a termo. Desta forma poderíamos reescrever a equação 1.17 como:

$$X[k,m] = \sum_{n=0}^{n=N-1} x[m-n] * w(n) * e^{(-2jnk\pi/N)} \ , \ k = 0,1,2,...,N-1 \tag{1.20}$$

1.4 Análise de Prony

A análise de Prony foi desenvolvida em 1795 de modo a explicar a espanção de gases. Ela se assemelha em parte a DFT, entretanto se propõe a ajustar uma soma de exponenciais complexas amortecidas a uma sequência de dados igualmente espaçados, ao passo de que a DFT apenas estima as exponeciais complexas e em subdivisões predeterminadas da frequência de amostragem. A análise de prony não é somente uma técnica de análise de sinais, mas também de indentificação de sistemas amplamente utilizada em sistemas de potência, área biomédica, processamento de fala, decaimento radioativo entre outas. A análise de Prony é conhecida por não se comportar muito bem quando um sinal contém ruído, a técnica não faz distinção entre sinal e ruído, e também ajusta as exponenciais às pertubações presentes. Seguindo o proposto em [3], tem-se o seguinte para um sinal $y[k]$ imaginando M exponenciais para se ajustarem a 2M amostras, respeitando o teorema da amostragem:

$$y[k] = \sum_{m=1}^{M} A_m e^{(\alpha_m + j2\pi f_m)(k-1)\Delta t + j\sigma_m} \ , \ k = 1, 2, ..., 2M \tag{1.21}$$

Onde fm é a frequência das exponenciais, Δt é o tempo de amostragem, α_m é o coeficiente de amortecimento, e σ_m é o ângulo de defasagem. Para o método proposto, estamos apenas interessados em saber quais são as frequências presentes no sinal. Se imaginamos nosso sistema sendo auto regressivo (AR) de ordem M, em qualquer instante de k é possível prever o valor $y[k]$ considerando apenas as M amostras anteriores deste mesmo sinal. Pode-se então montar uma equação de diferenças, que neste caso também é um modelo de predição linear [9]:

$$y[k] = \sum_{m=1}^{M} a_m y[k-m] \tag{1.22}$$

É sabido que exponenciais complexas são soluções para tal modelo. Tendo em mãos os valores dos coeficientes a é possível montar o polinômio característico da equação, e suas raízes (exponenciais complexas) são soluções para nosso problema. De posse dos coeficientes a devemos então encontrar as raízes do polinômio seguinte:

$$P(z) = z^M - a_1 z^{M-1} - ... - a_M z^0 \tag{1.23}$$

Como os coeficientes a são reais, as raízes complexas estão em pares complexos conjugados, desta maneira cada par complexo representa uma possível senoide já que $cos(wt) = \frac{e^{jwt}+e^{-jwt}}{2}$:

$$f_m = atan(Im\{z_m\}/Re\{z_m\})2\pi fs \tag{1.24}$$

Onde ${}_zm$ é uma das raízes e fs é a frequência de amostragem. As frequências f_m são possíveis soluções do sistema, não necessariamente elas estarão presentes. Qualquer combinação linear dessas senoides também é solução do sistema AR planteado. Para encontrar a forma da solução real, é necessário fazer uso das condições iniciais conhecidas de nosso sistema. No caso, as próprias amostras $y[k]$.

Os coeficientes de amortecimento α_m são o módulo das raízes complexas que foram encontradas. E se alguma raíz não é complexa, isto somente significa que uma exponencial é solução para a equação.

$$\alpha_m = |zm| \tag{1.25}$$

1.4.1 Solução do modelo de predição linear

Existem diversas formas de solucionar um modelo deste tipo, são apresentadas algumas opções abaixo [8]:

Sistema linear

Conhecendo os valores $y[k]$ que são amostras de nosso sistema analisado, pode-se montar um sistema linear considerando a equação anterior:

$$y[k] = \sum_{m=1}^{M} a_m y[k-m] \tag{1.26}$$

$$\begin{bmatrix} y[k-1] & y[k-2] & \dots & y[k-M] \\ y[k-2] & y[k-3] & \dots & y[k-M-1] \\ \vdots & & & \vdots \\ y[k-M] & y[k-M-1] & \dots & y[k-2M+1] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} y[k] \\ y[k-1] \\ \vdots \\ y[k-M+1] \end{bmatrix} \tag{1.27}$$

Ou de forma simplificada:

$$\boldsymbol{Y_k a} = \boldsymbol{y_k} \tag{1.28}$$

A forma mais simples de resolver este sistema é inverter a matriz $\boldsymbol{Y_k}$:

$$\boldsymbol{a} = \boldsymbol{Y_k}^{-1} \boldsymbol{y_k} \tag{1.29}$$

Um dos problemas com esta solução é que ela é computacionalmente custosa, já que não está implementada de maneira recursiva, mas pior que isso é o fato de que normalmente há ruído no sistema, e desta maneira se ajusta os valores de $\boldsymbol{a}$ ao ruído também, o que em geral não é o desejado, e neste caso específico, certamente não é. Desta forma, cada vez que calculamos o vetor $\boldsymbol{a}$, ele pode sair completamente diferente do anterior, levando à estimações equivocadas. Modelando $y[k]$ como:

$$y[k] = \sum_{m=1}^{M} a_m y[k-m] + \xi_k \tag{1.30}$$

Em que ξ_k é ruído branco, tem-se um modelo estocástico do sinal, que possibilita pensar em soluções mais arrojadas. As equações que caracterizam a solução do modelo AR também são conhecidas como equações de Youle-Walker e têm como solução ótima o chamado algoritmo de Levinson-Durbin [15] o qual não será abordado neste trabalho.

1.4.2 Filtros adaptativos

O Elemento linear adaptativo, ou filtro FIR adaptativo, também chamado de ADALINE, é uma rede neural artificial composta de uma única camada e com função de ativação linear. Sendo $\boldsymbol{x}$ as entradas, $\boldsymbol{w}$ os pesos, b o valor de bias, o ADALINE pode ser implementado matricialmente da seguinte maneira:

$$y = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} + b = \boldsymbol{x}^T \boldsymbol{w} + b \tag{1.31}$$

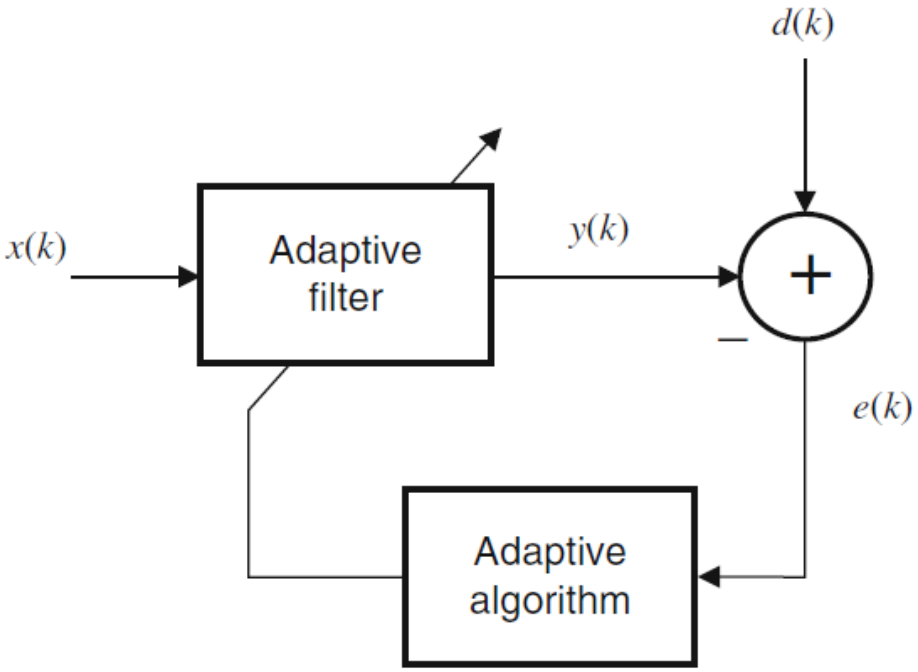


Figura 1.1: legenda aqui

Filtros adaptativos são considerados sistemas não lineares, portanto a análise de seu comportamento é mais complexa que de a filtros com coeficientes fixos. Por outro lado, pelo fato de serem auto-projetados, por um ponto de vista mais prático, eles podem ser considerados menos complicados que os convencionais em termos de projeto[6].

O desenho usual de um filtro adaptativo pode ser visto na figura 1.1, onde k é o número da iteração, $x(k)$ é a entrada do filtro, $y(k)$ é o sinal de saída, $d(k)$ é o valor de referência e $e(k) = d(k) - y(k)$ é o valor de erro, necessário para o algoritmo de adaptação do filtro. O algoritmo de adaptação é o prcesso usado para ajustar os coeficientes do filtro de modo a minimizar o erro de acordo com os critétios preestabelecidos. A escolha do algoritmo determina muitos aspectos do processo de adaptação, como a existência de soluções subótimas e complexidade computacional.

1.4.3 Adaptação via gradiente descendente

Uma das formas mais antigas e consagradas de otimizar uma função dentro dos métodos clássicos é seguir a direção do gradiente desta função, com relação as variáveis de interesse, para atualizar os valores destas variáveis de forma iterativa.

Seja um sistema descrito pela série temporal $u[n] n = 0, 1, 2..$ e um filtro de resposta impulsiva real $w_0, w_1 ... w_{M-1}$:

$$y[n] = \sum_{k=0}^{M-1} u[n-k] w_k \tag{1.32}$$

Todo este desenvolvimento pode ser feito de maneira bastante similar considerando entradas complexas e filtros com coeficientes complexos, mas por questões de simplicidade será focado o caso em que ambos são reais.Supondo que se deseja que a

saída deste filtro seja uma outra série temporal igualmente real $d[n]$, assim o erro seria:

$$e[n] = d[n] - y[n] \tag{1.33}$$

Considerando o caso de u estocástico, $e[n]$ é uma variável aleatória. Deseja-se então de minimizar a esperança de $e^2[n]$. Seja a função de custo:

$$J = E[e^2[n]] = E[(d[n] - y[n])^2] \tag{1.34}$$

Definindo nosso operador gradiente ∇_k:

$$\nabla_k J = -2E[e[n]u[n-k]] \tag{1.35}$$

Para encontrar o ótimo com relação a esta função de custo, $\nabla_k J$ deve ser igual a zero para todo k, como a esperança do produto de duas variáveis aleatórias é sua correlação, $u[n-k]$ e $e[n]$ devem estar descorrelacionadas para todo k. A este resultados chamamos princípio da ortogonalidade. Um corolário deste princípio é que y também deve ser ortogonal, ou descorrelacionada com $e[n]$.

É mais conveniente trabalhar com a notação matricial destes resultados. Ao vetor coluna que contém as variáveis aleatórias $u[n-k]$, de k=0 a K=M-1, se chama $\boldsymbol{u}_n$ o vetor de coeficientes w, igualmente:

$$\boldsymbol{u}_n = \begin{bmatrix} u[n] & u[n-1] & \dots & u[n-M+1] \end{bmatrix}^T$$

$$\boldsymbol{W}_n = \begin{bmatrix} w_0 & w_1 & \dots & w_{M-1} \end{bmatrix}^T$$

Desenvolvendo a equação anterior, obtém-se:

$$\nabla J = -2E[e[n]\boldsymbol{u}_n] = 2E[\boldsymbol{u}_n \boldsymbol{U}_n^T \boldsymbol{W} - d[n]\boldsymbol{u}_n] \tag{1.36}$$

Desta última equação $E[d[n]\boldsymbol{U}_n]$ é o vetor de correlação cruzada entre $d[n]$, ao qual se chama $\boldsymbol{r_{du}}$ e as amostras atrasadas do sinal de entrada. Destacamos também que $E[\boldsymbol{U}_n \boldsymbol{U}_n^T]$, a qual chamaremos $\boldsymbol{R_{uu}}$, é a matriz MxM de autocorrelação do mesmo sinal. Sendo assim:

$$\boldsymbol{R}_{uu}\boldsymbol{w} - \boldsymbol{r}_{du} = 0 \tag{1.37}$$

De onde se conclui que o vetor coeficientes ótimos $\boldsymbol{w}_{opt}$ é:

$$\boldsymbol{w}_{opt} = \boldsymbol{R}_{uu}^{-1}\boldsymbol{r}_{du} \tag{1.38}$$

Se u é um processo estacionário em sentido amplo, então é possível coletar amostras suficientes para fazer uma boa estimação das matrizes acima, encontrando assim um filtro muito mais apropriado para os propósitos deste trabalho. Desta maneira, se estima o conteúdo espectral de u com precisão. Reparemos que este método não é propriamente um gradiente descendente já que encontramos a solução em um passo apenas.

A solução acima tem sua utilidade, entretanto, se a natureza do sinal mudar, não teremos mais uma boa aproximação do mesmo, neste caso, é melhor resolver estas equações de forma recursiva, de modo que o filtro possa se adaptar constantemente à mudanças ocorridas.

1.4.4 O algoritmo LMS

Para atualizar o vetor de pesos $\boldsymbol{w}_n$ na direção oposta a do gradiente, definimos um coeficiente de aprendizagem μ. A cada iteração vamos fazer o seguinte:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu \nabla J$$
$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \mu(\boldsymbol{r}_{du} - \boldsymbol{R}_{uu}\boldsymbol{w}_{n+1}) \tag{1.39}$$

O que diferencia o LMS de outros métodos de gradiente descendente é a forma de estimar as matrizes $\boldsymbol{R}_{uu}$ e $\boldsymbol{r}_{du}$. Estas serão estimadas da seguinte forma:

$$\hat{\boldsymbol{R}}_{uu} = \boldsymbol{u}_n \boldsymbol{u}_n^T \tag{1.40}$$
$$\hat{\boldsymbol{r}}_{du} = \boldsymbol{u}_n d[n] \tag{1.41}$$

O que pode soar uma heresia, mas faz sentido se lembra que apenas temos M amostras de sinal.

1.4.5 O algoritmo NLMS

Uma melhoria no algoritmo anterior pode ser proposta. Multiplicando a parte que está entre parênteses na equação 1.39:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \mu \boldsymbol{R}_{uu}^{-1}(\boldsymbol{r}_{du} - \boldsymbol{R}_{uu}\boldsymbol{w}_{n+1}) = \boldsymbol{w}_{n-1} + \mu(\boldsymbol{w}_{opt} - I\boldsymbol{w}_{n-1}) = \boldsymbol{w}_{opt} \tag{1.42}$$

Em teoria, este algoritmo convergiria em apenas uma iteração. Resta o empecilho de estimar tal matriz. Com estimador anterior isto não é possível, porque a matriz estimada $\hat{\boldsymbol{R}}_{uu}$ não é inversível por ser o produto de dois vetores. Mas podemos fazer uma aproximação da forma $\hat{\boldsymbol{R}}_{uu} + \boldsymbol{I}\epsilon$. Que certamente é inversível e confere uma inversa útil, para valores pequenos de ϵ. E após alguma álgebra [2], que pode ser encontrada em [8], chegamos ao seguinte:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \mu \frac{\boldsymbol{u}_n}{|\boldsymbol{u}_n|^2 + \epsilon}(d[n] - \boldsymbol{u}_n \boldsymbol{w}_{n-1}) = \boldsymbol{w}_{n-1} + \mu \frac{\boldsymbol{u}_n}{|\boldsymbol{u}_n|^2 + \epsilon} e[n] \tag{1.43}$$

A este algoritmo se chama LMS normalizado, ou NLMS. Há ainda o estudo de convergência para valores pequenos de μ, que também pode ser encontrado em [6] e [8], tanto para o LMS quanto para o NLMS. Mas de modo geral o NLMS deve convergir respeitadas todas as condições estabelecidas anteriormente e escolhendo valores de $\mu < 1$.

1.4.6 RLS

O método RLS (*Recursive Least Squares*), ou mínimos quadrados recursivo, é um método de adaptação que visa a minimização da soma dos quadrados da diferença entre o sinal de referência e o sinal de saída do filtro em questão (o erro), assim como os anteriormente mencionados. O RLS pode ser obtido a partir do LMS (*Least Mean Squares*), e é conhecido por possuir rápida convergência, tendo boa performance em sistemas variantes no tempo, como o caso de nosso interesse. Isto vem ao custo de certa complexidade computacional aliada a problemas de estabilidade [6].

Voltando ao modelo do filtro adaptativo da seção anterior, havia sido feita uma estimação bastante simplória de $\boldsymbol{R}_{uu}$ para o LMS, e um pouco mais desenvolvida no NLMS, agora se pode imaginar uma outra: nos casos anteriores somente se aproveitava as M últimas amostras de nosso sinal para cada iteração, mas se por exemplo faz-se uma média das $\boldsymbol{R}_{uu}^{(k)}$ nos aproximamos mais da matriz verdadeira de autocorrelação. Podemos escolher então um novo estimador para $\boldsymbol{R}_{uu}$:

$$\boldsymbol{S}_d^{-1} = \hat{\boldsymbol{R}}_{uu} = \frac{1}{k+1}\sum_{i=0}^{k} \lambda^{k-i}\boldsymbol{U}_i\boldsymbol{U}_i^* \tag{1.44}$$

Esta definição em realidade é uma média ponderada em uma série geométrica, e se $\lambda < 1$ as últimas amostras contam mais que as primeiras. Reparemos que se $\lambda = 1$ tem-se um estimador consistente da matriz. À λ se chama fator de esquecimento.

Outra forma de se chegar ao RLS é por sua função objetivo dada por:

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i}\varepsilon^2(i) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \boldsymbol{x}(k)\boldsymbol{w}(k)] \tag{1.45}$$

Onde ε é o erro a posteriori no instante i. É possível estimar a matriz $\boldsymbol{R}_{uu}$ de forma recursiva, para evitar cálculos desnecessários, e melhor que isso, sua inversa também pode ser calculada de maneira recursiva, o que é mais importante para nós. A inversa pode ser substituída na expressão obtida para o NLMS. A dedução no entanto não nos interessa tanto no trabalho, ela pode ser encontrada em [8] e [6]. Ficamos com o resultado final:

$$\boldsymbol{\phi}_k^{-1} = \lambda^{-1}\left[\boldsymbol{\phi}_{k-1} - \frac{\boldsymbol{\phi}_{k-1}\boldsymbol{u}_k^T\boldsymbol{u}_k\boldsymbol{\phi}_{k-1}}{\lambda + \boldsymbol{u}_k\boldsymbol{\phi}_{k-1}\boldsymbol{u}_k^T}\right] \tag{1.46}$$

Os pesos então podem ser atualizados da seguinte maneira:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \boldsymbol{\phi}_n^{-1}e[n]\boldsymbol{u}_n \tag{1.47}$$

Uma recomendação de inicialização de $\boldsymbol{\phi}_0$ é $\boldsymbol{\phi}_0 = \delta\boldsymbol{I}$, sendo δ o inverso da potência estimada do sinal.

1.5 PLL

Um Phase Locked Loop digital, assim como sua versão analógica, visa determinar os parâmetros de um processo estocástico como uma onda senoidal. Desta forma o PLL tenta estimar parâmetros de Amplitude, fase e frequência. Temos diversas variantes digitais e analógicas [1], a variante utilizada neste trabalho foi proposta por Ziarani em 2004 [12]. Uma breve demostração será realizada abaixo. Seja $u(t)$ uma função variante no tempo, e $y(t) = Asin\phi(t)$ um sinal periódico senoidal sendo A a amplitude do sinal e $\phi(t)$ sua fase, tendo este sinal uma frequência constante, podemos escrever $\phi(t) = wt + \delta$, sendo w igual a frequência e δ um valor de fase constante. Escrevendo de maneira mais geral:

$$u(t) = \sum_{i=0}^{\infty} A_i sin\phi_i(t) + n(t) \tag{1.48}$$

Onde $n(t)$ representa um distúrbio, como um ruído. Na realidade, todos os parâmetros podem variar com o tempo, é conveniente escrever então um conjunto $\Psi(t) = [A(t), w(t), \delta(t)]$ o qual contém todos os parâmetros de um possível sinal $y(t)$.

Definimos a função d:

$$d^2(t, \Psi(t)) = [u(t) - y(t, \Psi(t))]^2 = e^2(t) \tag{1.49}$$

Assim sendo o conjunto $\Psi(t)$ ótimo o que minimiza a função d^2 que será adotado como função de custo $J(\Psi(t), t)$. Os parâmetros $\Psi(t)$ podem ser estimados por meio de gradiente descendente.

$$\frac{d\Psi(t)}{dt} = -\boldsymbol{\mu}\frac{\partial J(\Psi(t), t)}{\partial \Psi(t)} \tag{1.50}$$

$$\boldsymbol{\mu} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \tag{1.51}$$

Com $\frac{d\Psi(t)}{dt}$ denotando o vetor na direção do qual são atualizados os valores de $\Psi(t)$ a cada iteração. E $\boldsymbol{\mu}$ a matriz diagonal contendo as constantes de atualização referentes a cada parâmetro. Como estaremos lidando com estimações agora, usaremos a notação $\hat{\Psi}(t)$:

$$\begin{bmatrix} \frac{d\hat{A}(t)}{dt} \\ \frac{d\hat{w}(t)}{dt} \\ \frac{d\hat{\delta}(t)}{dt} \end{bmatrix} = -\boldsymbol{\mu} \begin{bmatrix} \frac{\partial e^2(t)}{\partial \hat{A}} \\ \frac{\partial e^2(t)}{\partial \hat{w}} \\ \frac{\partial e^2(t)}{\partial \hat{\delta}} \end{bmatrix} \tag{1.52}$$

Obtém-se então o seguinte:

$$\frac{d\hat{A}(t)}{dt} = 2m_1 e(t) sin(\int_0^t \hat{w}(\tau)d\tau + \hat{\delta}(t)) \tag{1.53}$$

$$\frac{d\hat{w}(t)}{dt} = 2m_2 e(t)\hat{A}(t) t cos(\int_0^t \hat{w}(\tau)d\tau + \hat{\delta}(t)) \tag{1.54}$$

$$\frac{d\hat{\delta}(t)}{dt} = 2m_3 e(t)\hat{A}(t) cos(\int_0^t \hat{w}(\tau)d\tau + \hat{\delta}(t)) \tag{1.55}$$

$$\frac{d\hat{\phi}(t)}{dt} = \hat{w}(t) + \frac{\hat{\delta}(t)}{dt} \tag{1.56}$$

Por conta do fator temporal t que aparece solto na equação referente a $\hat{w}$, esse sistema é variante no tempo, o que o torna instável. No entanto uma solução heurística é substituir t por uma constante $m4$, que pode ser absorvida pela constante $m2$, tornando o sistema invariante no tempo e fazendo com que este sistema de equações seja útil na prática. Desta maneira tem-se o seguinte:

$$\frac{d\hat{A}(t)}{dt} = 2\mu_1 e(t) sin(\int_0^t \hat{w}(\tau)d\tau + \hat{\delta}(t)) \tag{1.57}$$

$$\frac{d\hat{w}(t)}{dt} = 2\mu_2 e(t)\hat{A}(t)tcos(\int_0^t \hat{w}(\tau)d\tau + \hat{\delta}(t)) \tag{1.58}$$

$$\frac{d\hat{\phi}(t)}{dt} = \hat{w}(t) + 2\mu_3 e(t)\hat{A}(t)cos(\int_0^t \hat{w}(\tau)d\tau + \hat{\delta}(t)) \tag{1.59}$$

Usando o método *Euler Forward*:

$$A[n+1] = A[n] + \mu_1 T_s e[n] sen(\phi(t)) \tag{1.60}$$

$$w[n+1] = w[n] + \mu_2 T_s e[n] cos(\phi(t)) \tag{1.61}$$

$$\phi[n+1] = \phi[n] + T_s w[n] + \mu_3 T_s e[n] cos(\phi(t)) \tag{1.62}$$

Está definido o conjunto de equações discretas que serão utilizadas para rastrear frequências no restante do trabalho.

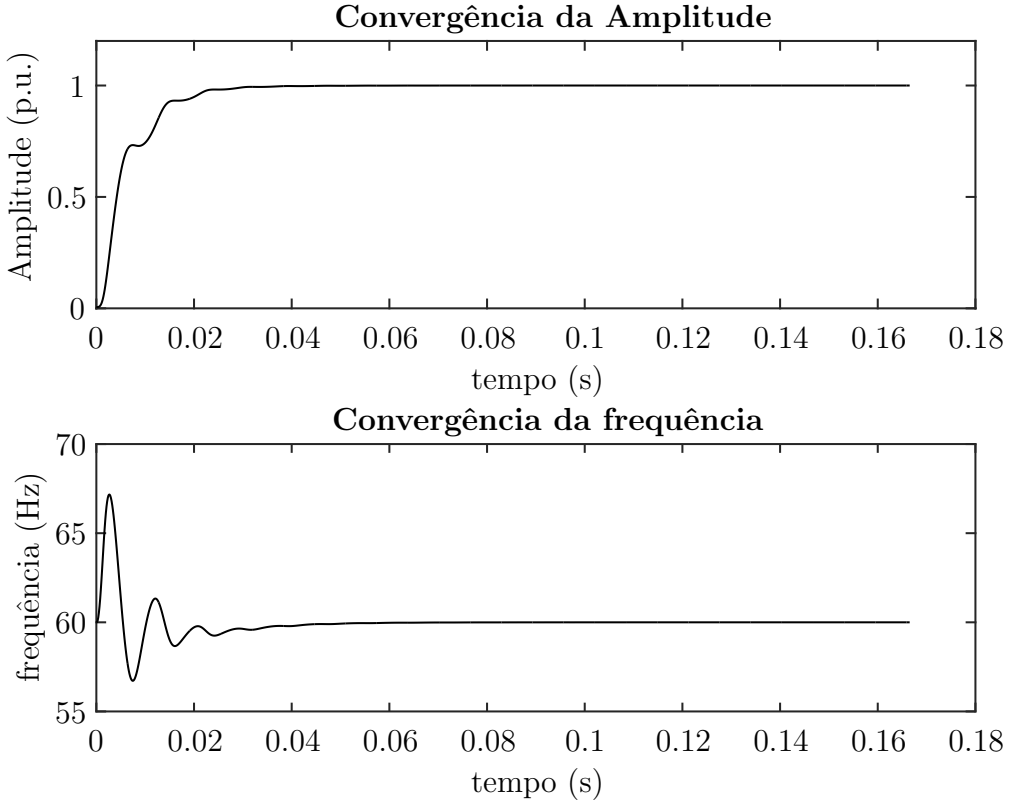


Figura 1.2: Convergência do PLL com a presença de apenas a fundamental

1.6 Processamento Multitaxa

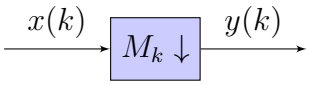
Em determinadas situações, é desejável alterar a taxa de amostragem de um sinal para realizar certos tipos de processamento. Por exemplo, veremos em alguns resultados nas próximas sessões que o algoritmo PLL descrito anteriormente é sensível a taxa de amostragem em relação a frequência rastreada, portanto é conveniente mantê-la mais ou menos constante. Também veremos que processar sinais na taxa

necessária para abranger todo o espectro que se quer analisar é extremamente custoso computacionalmente e não é garantia de melhores resultados. Por isso deve-se considerar o uso de uma estrutura multitaxa que diminua a frequência base.

Serão analisadas a seguir as implicações de um abaixamento na frequência de amostragem:

1.6.1 Downsampling

Uma estrutura downsampling tem a forma seguinte:



Representamos a estrutura como um sistema que reproduz em sua saída apenas amostras onde k é múltiplo de M_k. Podemos escrever $y[n] = x[nM_k]$ $n = 1, 2, 3...$ Considerando que $x[k]$ foi amostrado com a frequência de Nyquist fs, o efeito final é o mesmo que se tivéssemos amostrado x com fs/M_k, podemos dizer que vamos observar aliasing em nosso resultado final. Haverá portanto sobreposição no espectro de frequência referente a $X(w)$ quando analisarmos $Y(w)$. Entretanto, é possível prever onde estarão localizadas as componentes espectrais presentes no sinal completo que estamos subamostrando, e embora não possamos saber exatamente qual é o valor correspondente as frequências originais, podemos saber no mínimo qual é sua soma.

Uma demonstração formal dos efeitos observados pode ser encontrada em [10], aqui faremos uma abordagem mais intuitiva. Dada uma componente espectral de $x[k]$ referente a w_0. Supõe-se que é feita uma subamostragem neste sinal, e agora sua frequência de amostragem que era f_s se torna f_s/M_k. Agora, se $w_0 < f_s/(2Mk)$ essa as componentes dessa frequência não saem do lugar, elas continuam onde estavam anteriormente. Isso não significa que seu espectro não seja alterado nas frequências mais baixas que a nova taxa de amostragem, somente quer dizer que elas sofrem interferência no mesmo lugar onde estavam antes. Agora, se a frequência de interesse é maior que a nova frequência de amostragem dividida por 2, então ela não poderá de forma alguma ser encontrada no mesmo local, pois simplesmente não há como visualizá-la com DFT ou DTFT. Mas então onde vão parar essas frequências? Bom, onde pararia qualquer uma que sofre aliasing. Ela dá voltas no círculo de frequências. O circulo de frequências sempre vai de 0 a 2π, e as frequências sempre caminham nele no sentido anti-horário. Para cada vez que a frequência de interesse (a qual chamaremos f_i) for maior que fs, a frequência em questão dá uma volta no círculo. Um algoritmo simples será apresentado quando expusermos o método do PLL multitaxa, mas por hora podemos saber que o ângulo referente à nova frequência será o resto da divisão f_i/f_s e multiplicamos por 2π.

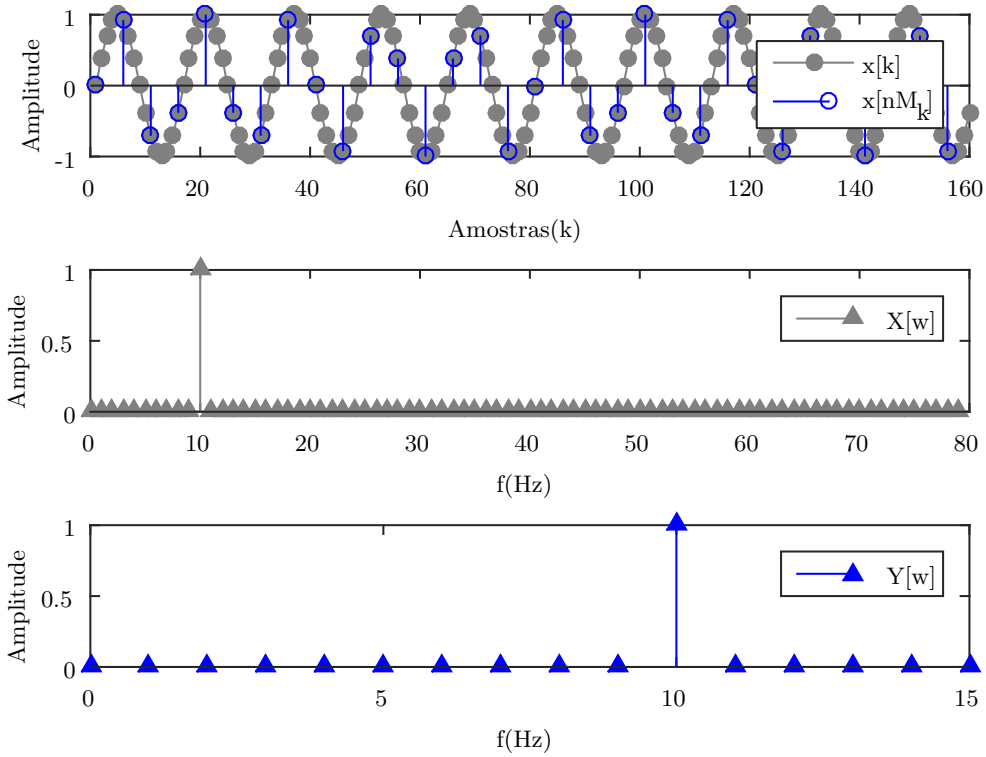


Figura 1.3: Downsampling para $f_s = 160Hz$, $f_0 = 10Hz$, $M_k = 5$

No exemplo da figura 1.3, temos $fs = 160Hz$ e $M_k = 5$, a frequência de interesse é 10Hz, a frequência de amostragem depois da subamostragem seria ainda maior que $2f_i$, então ela não se move, como observado na figura. Entretanto, no caso de $M_k = 10$ a coisa muda, calculando:

$$w_i = rest(f_i\ M_k,\ f_s)2\pi \tag{1.63}$$

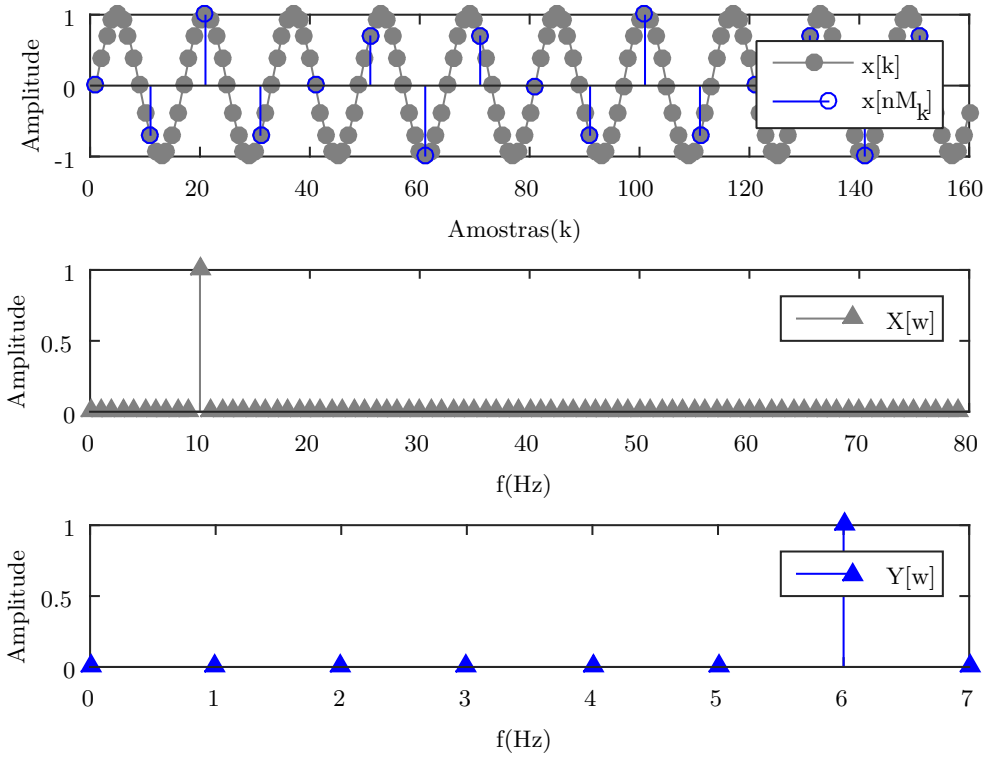


Figura 1.4: Downsampling para $f_s = 160Hz$, $f_0 = 10Hz$, $M_k = 10$

Como resultado tem-se $\frac{5}{4}\pi$ que é maior que é ligeiramente maior que π portanto já se pode saber que esta componente sofre aliasing, como se nota na figura 1.4.

Capítulo 2

Estrutura PLL-Multitaxa

Como visto na seção referente ao PLL, esse método é capaz de minimizar o erro quadrático médio entre um sinal $y(t)$ e uma componente senoidal para ao menos um mínimo local. Neste capítulo ganhará forma uma estrutura reunindo o PLL e o processamento multitaxa com o objetivo de reduzir a complexidade computacional e conseguir um algoritmo mais versátil.

2.1 Desenpenho do PLL

Nas figuras seguintes pode-se observar como converge o algoritmo em diferentes situações, todas foram simuladas para um sinal de 180 V de amplitude e constantes 300, 500, e 6, com frequência de amostragem igual a 7680 Hz, partindo de condições inicias $f_i = 60Hz$ e $A = 0$. Percebe-se pela simulação que o algoritmo converge rapidamente mesmo com ruído, entretanto, na presença de harmônicos com a mesma quantidade de energia, a convergência já é bastante comprometida. Ainda assim, em valor médio, a estimação se mostra correta. Conclui-se que é possível estimar harmônicos diretamente com o algoritmo PLL obtido, entretanto é conveniente aliá-lo a outras técnicas para que se diminua o erro da estimação. Os resultados das simulações podem ser vistos nas figuras 2.1 e 2.2.

2.2 banco de filtros

A solução encontrada em [3] para parte dos problemas anteriormente citados é o uso de um preprocessamento com filtros passa-banda, para melhorar a relação sinal ruído, e posteriormente subamostragem, para diminuir a complexidade computacional, de modo também que não seja necessário mudar as constantes do algoritmo.

O conjunto de filtros utilizado é uma cascata de dois filtros IIR com a seguinte função de transferência:

$$H_{bp}(z) = \frac{1-\alpha}{2}\frac{1-z^{-2}}{1-\beta(1-\alpha)z^{-1}+\alpha z^{-2}} \tag{2.1}$$

$$\beta = cos(w_0) \tag{2.2}$$

O parâmetro α modifica a seletividade do filtro e está entre 0 e 1, para que este seja estável. Quanto mais próximo de 1, mais seletivo é o filtro. O parâmetro β

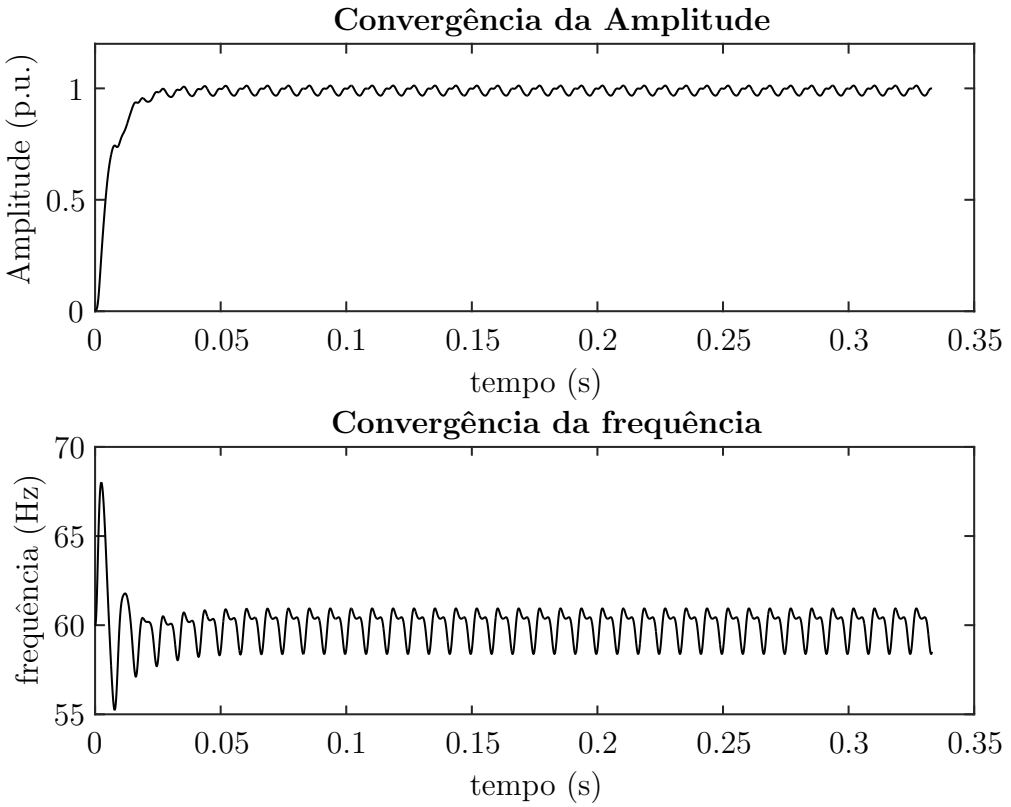


Figura 2.1: Convergência na presença do 3º e 5º harmônicos; SNR=10 dB

modifica a frequência central do filtro de acordo com a equação 2.5, onde w_0 é a frequência normalizada de acordo com a amostragem.

Este filtro é uma boa escolha por alguns motivos:

- Ele rechaça completamente a componente DC do sinal.
- Tem atraso de fase nulo na frequência central.
- É paramétrico, suas características dependem dos parâmetros α e β, os quais modificam propriedades muito específicas do filtro, sendo entAõ muito fácil utilizá-lo e modificá-lo em tempo real.

Tem-se talvez como única desvantagem o atraso de grupo que é máximo para a frequência central, e quanto mais seletivo é o filtro, maior é esse atraso. Deve-se também ter atenção com a frequência de amostragem, pois quanto maior, menos seletivo um mesmo α seria. Se por exemplo, se mantém α e aumenta f_s, frequências que antes estavam normalizadas mais longe de nossa frequência central anterior, agora estariam mais próximas, por assim dizer. Dessa forma quanto maior é f_s mais seletivo deve ser o filtro para a separação das mesmas frequências. Isso acaba se tornando um jogo de balanceamento destes parâmetros, pois como vimos anteriormente, um α maior também eleva o atraso de grupo, entretanto se tem mais amostras em menos tempo devido ao aumento em f_s. Todos estes efeitos estão muito bem descritos por J. Rodrigues em seu trabalho [4]. Ao final utilizaremos $\alpha = 0.975$ e dois filtros em cascata para o restante das simulações.

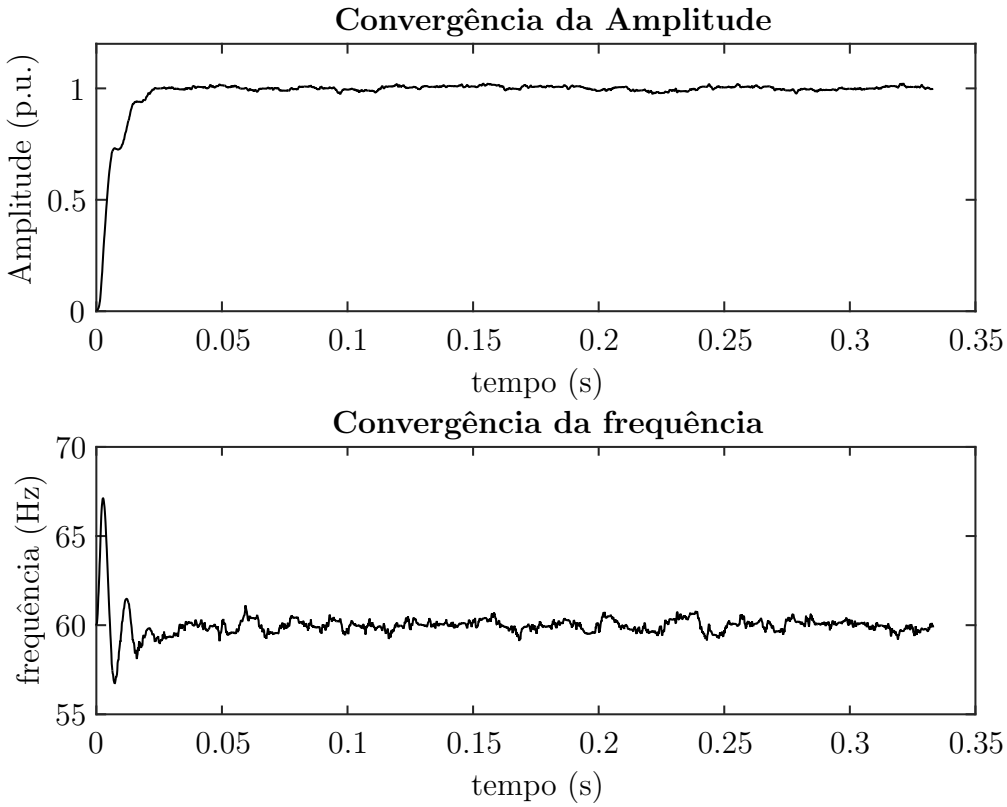


Figura 2.2: Convergência na presença de ruído; SNR=10 dB

2.3 Uso da estrutura multitaxa

Depois de passado pelo filtro passa-banda, pode-se realizar a subamostragem do sinal, uma vez que em tese eliminamos os harmônicos mais distantes quase que completamente, e estes não interferirão na estimação. Dentro revisão bibliográfica foi discutido o perfil desta interferência e também como encontrar a posição de uma frequência depois de esta sofrer aliasing. Agora é feita a exposição do algoritmo em C de uma função simples capaz de calcular esta frequência, que pode ser até mais explicativo que o texto anterior:

```
float freq_finder(const float Fs,
                  const float f_init, int *flag){

  float f_aparente=f_init;
  *flag=1; //valor padrao para a variavel

  //verifica se a frequencia investigada sofre aliasing
  if(f_init>(Fs/2)){
    //calcula o resto da divisao entre as frequencias
    f_aparente=fmod(f_init/Fs, 1);
    if(f_aparente<=0.5){
      //se o resto e menor ou igual a 1/2, estamos
      //no semicirculo positivo
      f_aparente=f_aparente*Fs;
```

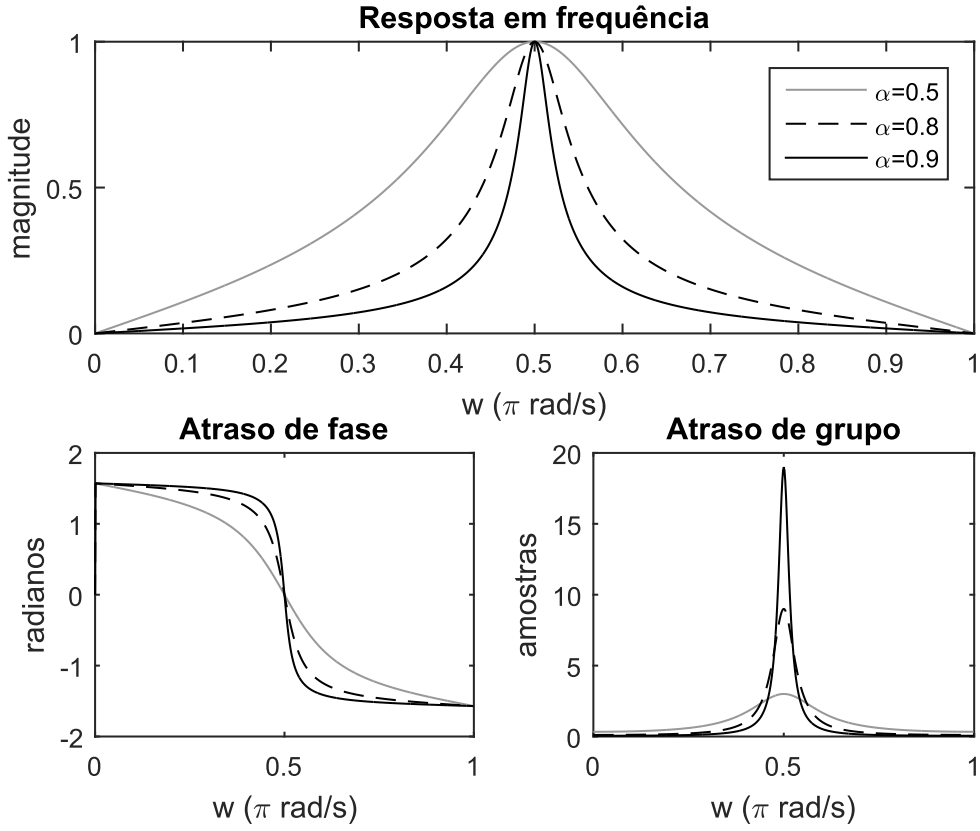


Figura 2.3: Características do filtro com w_0=0.25

```
    }
    else{
      //caso contrario, estamos no semicirculo negativo
      f_aparente=(1-f_aparente)*Fs;
      *flag=-1;
    }
  }

  return f_aparente;
}
```

A função recebe como argumentos a frequência de amostragem Fs, a frequência rastreada f_init, e o ponteiro para a variável flag, que indica se a frequência encontrada estava no semicírculo positivo, quando vale 1, ou negativo, quando vale -1 ($\theta < \pi$ ou $\theta > \pi$). É importante guardar esta informação porque necessitamos dela para recuperar a frequência original estimada.

É possível observar na figura 2.4 a localização de duas frequências f1 e f2 no círculo. Suponhamos que a frequência de amostragem é $f_s = 240Hz$, $f_2 = \frac{240}{2}\frac{3}{4}Hz = 90\,Hz$ e $f_2 = \frac{240}{2}\frac{4}{3}\,Hz = 160\,Hz$, encontramos facilmente suas colocações no círculo utilizando o algoritmo citado. Agora reparemos que diferentes frequências serão mapeadas nas mesmas posições do círculo, por exemplo $f_1 = \frac{240}{2}\frac{11}{4}\,Hz = 330\,Hz$ também é mapeada no mesmo ângulo, então não existe uma função inversa que devolva as frequências mapeadas para seus valores reais.

Também é importante se atentar ao fato de que quando se aumenta f_1 levemente,

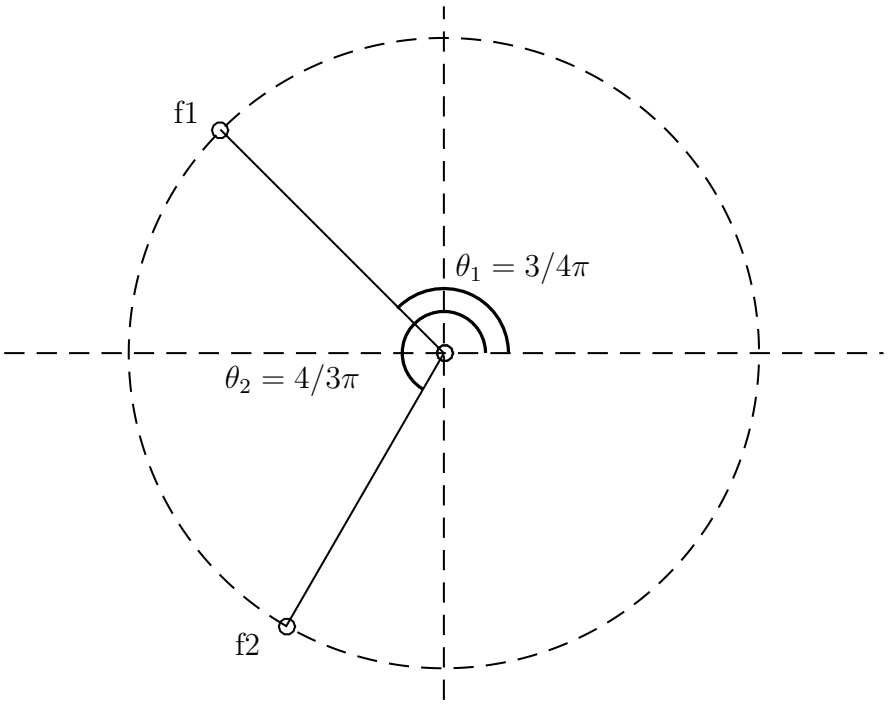


Figura 2.4: círculo de frequências

se está aumentando sua frequência aparente, mas quando se faz o mesmo com f_2 se está diminuindo sua frequência aparente. Por isso é importante guardar a variável 'flag', ela diz se acréscimos positivos em frequência aparente condizem à acréscimos ou decréscimos na frequência real, e uma vez que não existe uma função inversa que nos devolva a frequência real dada a aparente, deve-se utilizar da variação na estimação aparente e o valor inicial mapeado para recuperar a estimação real.

Uma coisa que se deve ter em mente é que algumas frequências se localizarão muito próximas a zero, e terão frequência aparente muito pequena. Isso dificulta muito a análise, é desejável testar a localização antes de iniciar o algoritmo e fazer uma mudança no valor de subamostragem caso necessário.

$$\hat{f} = f_{init} + \Delta f_{aparente}\, flag \tag{2.3}$$

2.4 variação da frequência central do banco de filtros

A estrutura multitaxa consiste em passar o sinal de entrada $x(t)$ pelo banco de filtros, sub-amostrar e passar este sinal para os respectivos PLLs. Como o filtro utilizado é bastante seletivo, a frequência estimada deve controlar o banco de filtros centrando a frequência corretamente. A maneira mais básica de realizar este controle seria alimentar o banco diretamente com as frequências obtida no PLL, entretanto este método é inviável. Vimos na seção sobre o algoritmo PLL utilizado

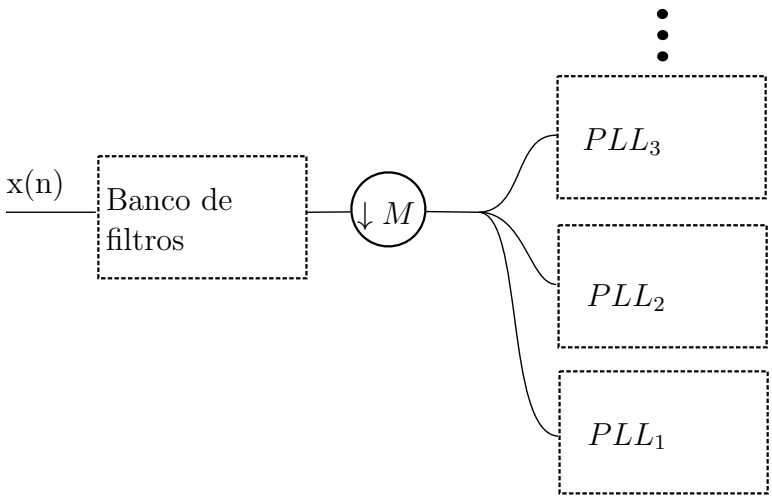


Figura 2.5: esquema multitaxa PLL

que este é altamente não linear, e complexo de se analisar a convergência. Fazer uma realimentação deste tipo muda o sistema e pode torná-lo instável, ou prejudicar sua convergência. A alternativa encontrada por J. Rodrigues [4] é utilizar a média das últimas L estimações, assim o filtro passa-banda se move de maneira mais suave e não prejudica tanto o PLL. Desta maneira:

$$w_0[k] = \frac{1}{L} \sum_{i=k-L+1}^{k} \hat{w}[i] \tag{2.4}$$

Uma outra opção é fazer a estimação com uma série geométrica, ou soma convexa, que pode ser calculada de forma recursiva:

$$w_0[k] = w_0[k-1](1-\lambda) + \lambda \hat{w}[k] \tag{2.5}$$

Onde λ é uma constante entre 0 e 1.

Simulando as três opções, a alimentação direta da estimação realmente se mostra mais instável, e com maior tempo de convergência, enquanto que o método geométrico em geral é levemente mais estável e de mais rápida convergência que o de média, além de ser mais fácil de computar. Todos foram testados para as mesmas constantes do exemplo anterior, amplitude de 180 V e 60 Hz iniciais, a constante de subamostragem escolhida foi $M_k = 16$. Depois de 1 segundo de simulação, é aplicado um degrau na frequência de -2 Hz. Também é passado um filtro média móvel de 16 amostras nos resultados finais de $\hat{w}$ e $\hat{A}$.

2.5 Síntese da estrutura PLL Multitaxa

1. Passamos o sinal por um banco de filtros.
2. Abaixamos a frequência em M_k amostras.
3. Calculamos a frequência aparente f_a da componente a rastrear.
4. Inicializamos o PLL com esta f_a e a guardamos para recompor a original.

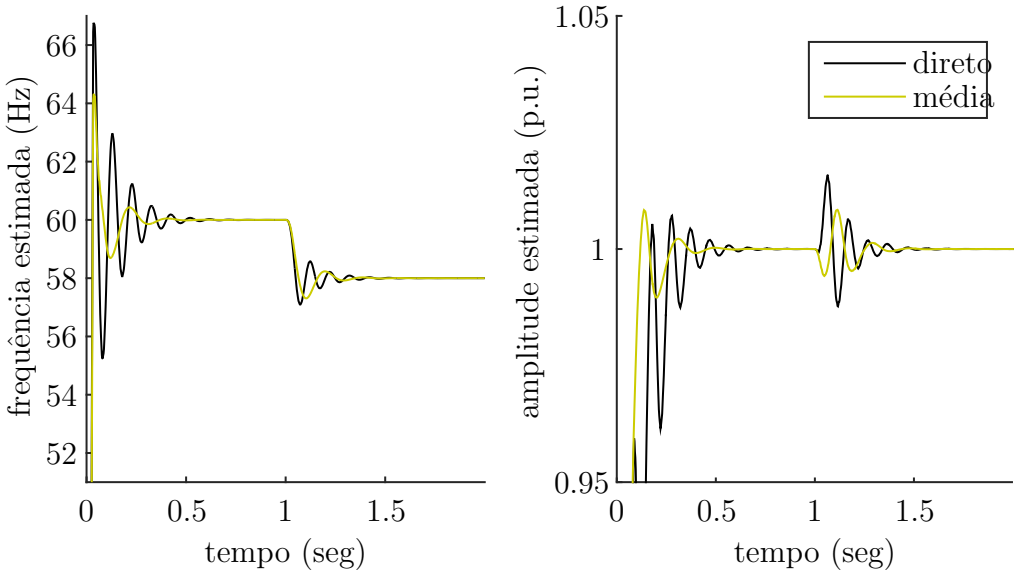


Figura 2.6: Comparação entre o método de média (com 24 amostras) e o de alimentação direta

5. Fazemos as estimações de w e A, usamos a equação 2.5 e utilizamos a série geométrica para realimentar o banco de filtros com a frequência calculada.

2.6 simulações

Para as simulações computacionais foram seguidos determinados critérios:

- Convergência em amplitude: Foram consideradas as últimas 32 estimações, assim que a média do erro quadrático destas fosse menor que 4e-4 (que é um erro menor que 2% do sinal), consideramos que o algoritmo convergiu.
- O mesmo para a convergência em amplitude, mas com umbral de 1e-4.
- Os cálculos de erro quadrático médio são feitos considerando o valor das estimações depois de 1 segundo. Quando o algoritmo já convergiu.

Os parâmetros da simulação são $f0 = 60Hz$, 128 pontos por ciclo de $f0$, $Fs = 7380\,Hz$, $\alpha = 0.975$ e dois filtros em cascata no banco. A constante de subamostragem é 16, para todos os harmônicos e inter-harmônicos. A amplitude da fundamental é 180 V, e para todos os harmônicos suas amplitudes são a da fundamental dividido pela ordem do respectivo harmônico. Foi passado um filtro média móvel de ordem 16 em todas as estimações de modo a suavizar a visualização dos resultados.

2.6.1 harmônicos ímpares

Abaixo estão os resultados para a simulação com os harmônicos ímpares de 1 a 15. Verificamos algumas coisas:

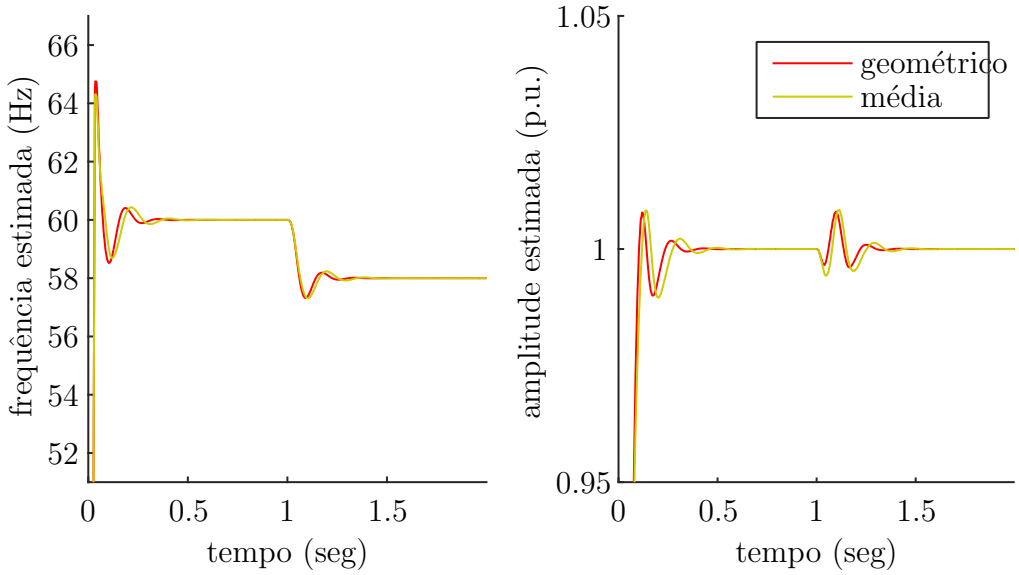


Figura 2.7: Comparação entre o método de média (com 24 amostras) e o geométrico ($\lambda = 0.9$)

- o valor de offset para a fundamental é relativamente grande, e isto se explica principalmente pela subamostragem, uma vez que alguns harmônicos não são completamente atenuados se sobrepõe à fundamental. Isso poderia ser melhorado com o aumento da seletividade do filtro, entretanto já discutimos os problemas que isto pode ocasionar. Seu uso ou não depende da natureza do problema em questão.
- o erro em frequência é em geral mais baixo que o de amplitude. E também é seu tempo de convergência.
- o tempo de convergência é difícil de prever, não é certo que os harmônicos de ordem mais elevada ou os de maior energia convergirão mais rapidamente que os demais.
- o algoritmo se comporta muito bem com ruído, principalmente a estimação de frequência.

Harmônico	Erro em frequência (%)	Erro em amplitude (%)	Tempo de convergência W (s)	tempo de convergência Amp. (s)
1	6,14E-07	1,344	0,144	0,098
3	2,08E-06	1,427	0,150	0,135
5	2,16E-06	1,338	0,035	0,088
7	3,05E-05	1,441	0,090	0,194
9	1,60E-05	0,913	0,035	0,090
11	4,93E-05	1,146	0,035	0,221
13	2,01E-04	0,136	0,035	0,090
15	4,57E-03	0,126	0,035	0,327

Tabela 2.1: Tabela para a simulação sem ruído

Harmônico	MSE em frequência (%)	MSE em amplitude (%)
1	4,21E-05	2,32E-04
3	6,05E-07	3,16E-04
5	4,23E-07	5,42E-04
7	3,70E-07	1,01E-03
9	5,56E-07	1,45E-03
11	3,26E-07	1,96E-03
13	7,37E-07	3,65E-03
15	7,76E-07	4,56E-03

Tabela 2.2: Tabela para a simulação com ruído (σ^2=10)

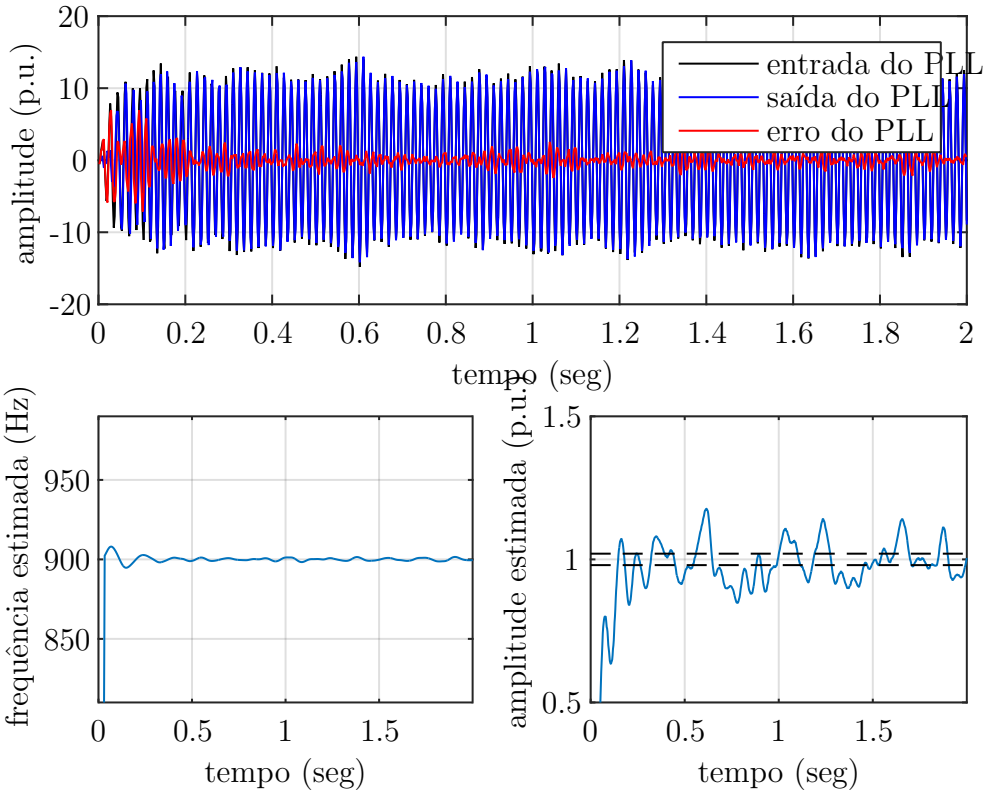


Figura 2.8: convergência do 15º harmônico na presença de ruído

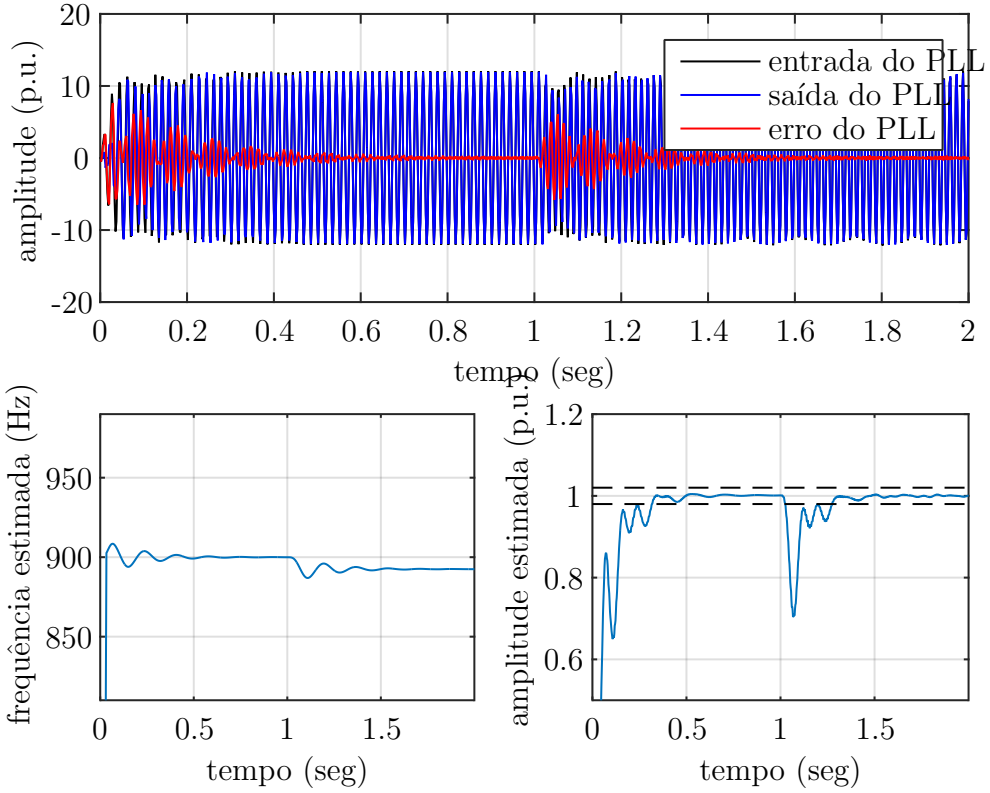


Figura 2.9: convergência do 15º harmônico com degrau em frequência

2.6.2 inter-harmônicos

Foram simuladas estimações com inter-harmônicos, seguindo as mesmas prescrições do caso anterior. Vemos que os tempos de convergência não são muito afetados, entretanto o erro quadrático médio para a fundamental aumenta um pouco. Isto se deve a um leve batimento que o harmônico 3.2 causa na fundamental, fazendo com que a estimação oscile um pouco. Entretanto se olhamos uma média grande o suficiente destas estimações, podemos nos livrar do batimento.

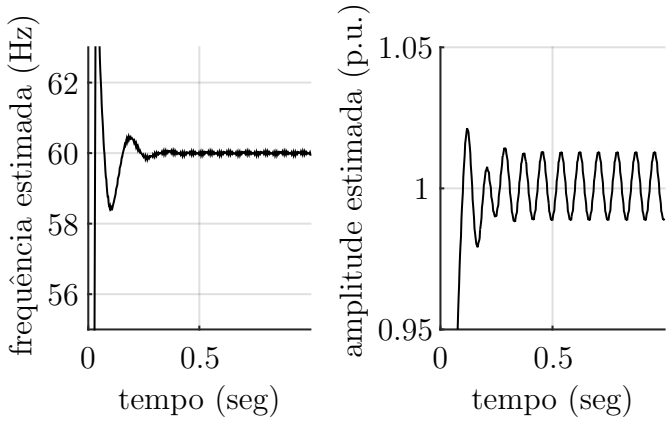


Figura 2.10: batimento observado na fundamental

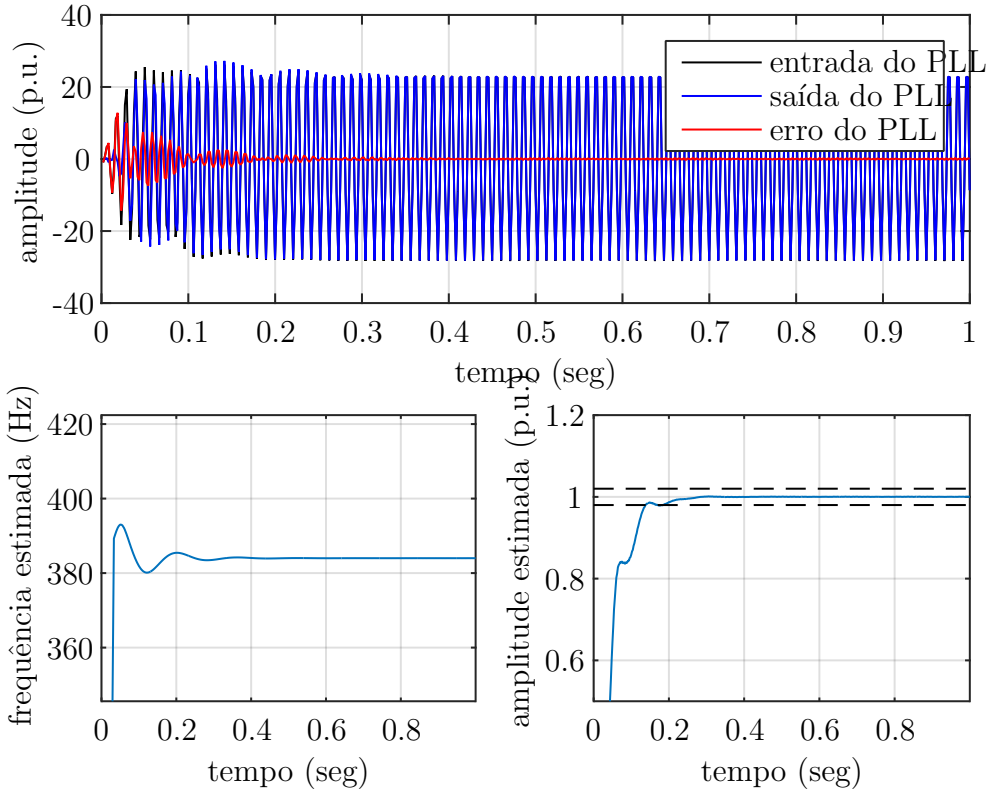


Figura 2.11: convergência do inter-harmônico 6.4

Harmônico	convergencia em freq. (s)	convergencia em amp (s)	MSE em frequência (%)	MSE em amplitude (%)
1	0,177083	0,1375	2,91E-05	1,66E-04
3,2	0,179167	0,175	6,73E-08	1,06E-05
6,4	0,114583	0,19375	2,46E-09	3,65E-06
11,3	0,11875	0,352083	5,04E-08	3,59E-06

Tabela 2.3: Tabela para a simulação de inter-harmônicos

2.7 esforço computacional

Considerando um sinal amostrado em 7380 Hz e uma subamostragem de 16 amostras, será feita a estimativa do esforço computacional considerando operações realizadas a cada iteração de soma, multiplicação e funções trigonométricas, sem considerar deslocamentos de buffers e alocação de memória:

2.7.1 Banco de filtros

Cada estrutura correspondente ao filtro apresentado na equação [2.1] representa 5 multiplicações e 4 somas. Sendo dois filtros, totaliza 10 multiplicações e 8 somas por cada amostra.

2.7.2 PLL

Cada estrutura PLL como apresentada na equação [só dá pra citar quando juntar os capítulos] representa 11 multiplicações, 5 somas e 4 trigonométricas, que são executadas a cada 16 amostras.

2.7.3 Atualização de frequência

Cada atualização de frequência custa 2 multiplicações, uma soma e uma trigonométrica.

2.7.4 Totais

Abaixo temos uma tabela com todos os cálculos necessários para 1 segundo de sinal, por cada frequência rastreada. Percebemos que mais de 90% dos cálculos são gastos pelo filtro digital.

Estrutura	Somas	Multiplicação	Trigonométricas
Banco	61440	76800	0
PLL	2400	5280	1920
Atualização	480	920	480
Total	64320	83040	2400

Capítulo 3

Estimação de frequências

Foi visto na seção sobre o PLL-Multitaxa que este é um método com grande eficácia no rastreio de frequências, entretanto é necessário saber onde elas estão, e isto o método não nos diz. Seria possível por exemplo utilizá-lo como um filtro de partículas e largar PLLs aleatoriamente pelo espectro do sinal, entretanto existem algumas formas de obter boas ideias de onde estão as componentes mais relevantes do sinal. Uma delas é a DFT ou a STFT com janelamento para reduzir o espalhamento pelo espectro, desta forma podemos identificar regiões do espectro onde sabemos que há energia relevante, sem saber no entanto a localização exata da componente, ou as componentes as quais pertence esta energia. Outro método visto na revisão foi o de Prony, do qual se fez uso no trabalho [5]. Neste trabalho utilizou-se da otimização via RLS para resolver o problema da predição linear e calculou-se as raízes do polinômio característico do modelo Auto-Regressivo do sinal, assim foi possível obter as componentes relevantes com certa precisão.

Neste mesmo trabalho, posteriormente se faz uso do RLS novamente para estimar a amplitude e fase de cada uma das componentes, alimentando o RLS com sinais senoidais em quadratura de modo que o filtro estimado fosse a amplitude de senoides e cossenoides [obs: Vou colocar uma diagrama disso depois].

Nesta seção faremos a exposição de um método baseado no trabalho citado com algumas modificações.

3.1 Solução em predição linear

O método planteado na seção de Análise de Prony visa encontrar os coeficientes w_m tais que:

$$u[k] = \sum_{m=1}^{M} w_m u[k-m] \tag{3.1}$$

Que ao final é o equivalente a encontrar a matriz de autocorrelação e o vetor de correlação cruzada do sinal $u[k]$ na forma:

$$\boldsymbol{w}_{opt} = \boldsymbol{R}_{uu}^{-1} \boldsymbol{r}_{du} \tag{3.2}$$

Onde desta vez o sinal de referência $d[k]$ ganha um caráter especial uma vez que é $u[k+1]$. Estas equação pode ser resolvida utilizando o método de Levinson–Durbin,

para um determinado conjunto de dados. Mas também pode ser resolvida com os filtros adaptativos a cada iteração, fazendo um rastreio online destes parâmetros.

3.2 Solução em RLS e NLMS

Como visto anteriormente, o RLS faz uma estimação da matriz $\boldsymbol{R}_{uu}^{-1}$ a cada iteração e geralmente é o método que nos vai dar menor MSEe em menos tempo, entretanto ele tem algumas deficiências:

- Se a matriz $\boldsymbol{R}_{uu}$ é singular ou não é bem ajustada, podemos ter problemas numéricos significativos, como a ordem dos elementos da matriz ser muito desbalanceada ou grande demais. Isso é bastante notável em nosso problema de estudo, pois em geral não se sabe a ordem do sistema que se está analisando, se este sistema tem uma ordem pequena e é estimado com um filtro grande, a matriz $\boldsymbol{R}_{uu}$ certamente será singular. Algo que pode ser feito para minimizar este problema é a adição de ruído.
- Em geral é mais lento para reagir a variações no sistema que o LMS.
- Tem uma complexidade computacional consideravelmente maior.
- Sua convergência é de certa forma caótica, perturbando o cálculo das raízes.

Deve-se também destacar que tanto o RLS quanto o NLMS se beneficiam de ordens de filtro próximas a quantidade de amostras por ciclo da fundamental. Isto se deve ao fato de que quanto menor são as variações dos sinais dentro do buffer $\boldsymbol{U}$ mais singular é a matriz de autocorrelação.

Fazendo uma simulação com os harmônicos de 1 até 7, com a amplitude igual ao inverso de sua ordem e ruído gaussiano com $\sigma = 0.02$, comparamos os resultados da divisão do menor pelo maior autovalor da matriz de autocorrelação estimada do sinal com diferentes valores de amostragem e ordem da matriz:

	M=16	M=64
f0x16	1.3e-03	2.1e-04
f0x64	2.7e-04	8.6e-05

A tabela mostra o que já era esperado, e o que pode ser confirmado com outras simulações mais adiante, obtém-se resultados melhores com uma ordem de modelo menor e com menores taxas de amostragem.

3.3 Simulações

Foram realizadas simulações com os seguintes parâmetros: $f_s = 32f0$, $M = 32$, harmônicos de 1 a 15, pares e ímpares todos com amplitude unitária. E é considerada uma variação aleatória nestas frequências somando-lhes um valor aleatório de uma V.A gaussiana com determinada amplitude. Apresentamos na tabela abaixo os resultados de porcentagem de erro para o RLS e NLMS. Também foi agregado ao sinal um ruído gaussiano com $\sigma = 0.02$. Depois de 15 ciclos do sinal é tomado o resultado:

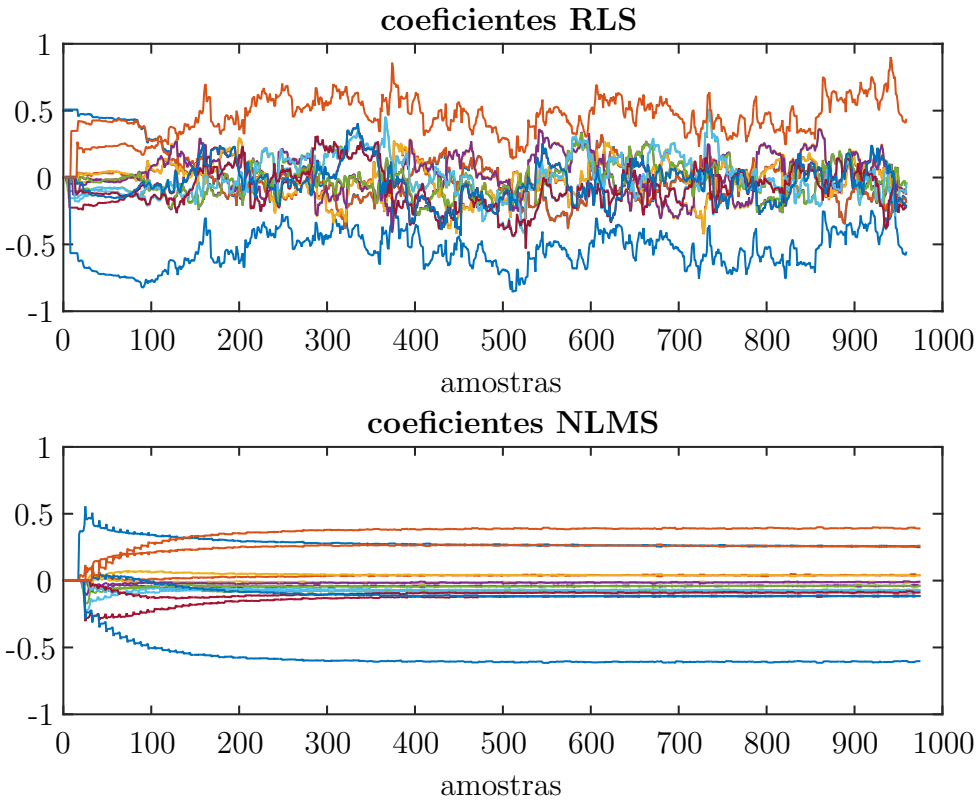


Figura 3.1: Convergência dos coeficientes RLS e NLMS na presença dos harmônicos 1, 3, 5 e 7; M=16

	NLMS	RLS
G(0, 0.1)	0	0
G(0, 0.25)	1.9	2.3
G(0, 0.5)	18.4	3.4
G(0, 0.7)	22.8	6.8
G(0, 1)	28.3	10.2

Tabela 3.1: resultados em %

É considerado um erro quando não se encontra dentre os valores estimados nenhum correspondente com distância menor que 0.1 entre os valores corretos, de maneira também que uma vez pareada uma estimação com um valor real, este valor real não mais pode ser pareado com outra estimação. Assim podemos ter uma boa ideia da capacidade do algoritmo de classificar frequências diferentes, mas que estão bastante próximas.

Também foram realizadas simulações para o caso de amplitude variável dos harmônicos, como uma V.A uniforme de 0 a 1.

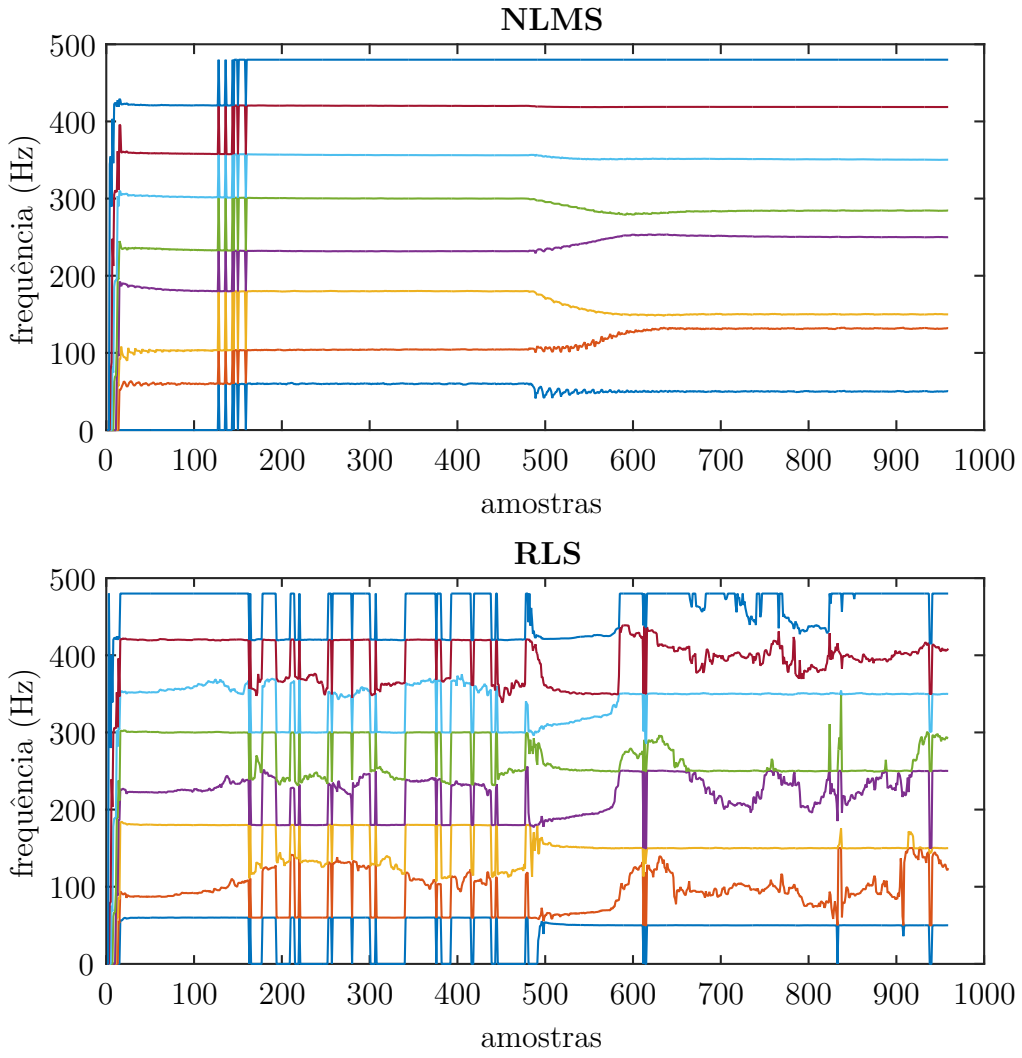


Figura 3.2: Convergência do RLS e NLMS vista na estimação das frequências, com um degrau de 10 Hz em 500 amostras

	NLMS	RLS
G(0, 0.1)	3.4	0.6
G(0, 0.25)	14.6	2.0
G(0, 0.5)	28.4	8.5
G(0, 0.7)	31.3	12.9
G(0, 1)	34.5	16.6

Tabela 3.2: Simulação com amplitude variável, resultados em %

3.4 frequências próximas

Frequências próximas em geral são um problema e o algoritmo tem bastante dificuldade em identificá-las. Muitas vezes estas são classificadas como sendo apenas uma. Uma maneira de resolver este problema é fazendo subamostragem do sinal e

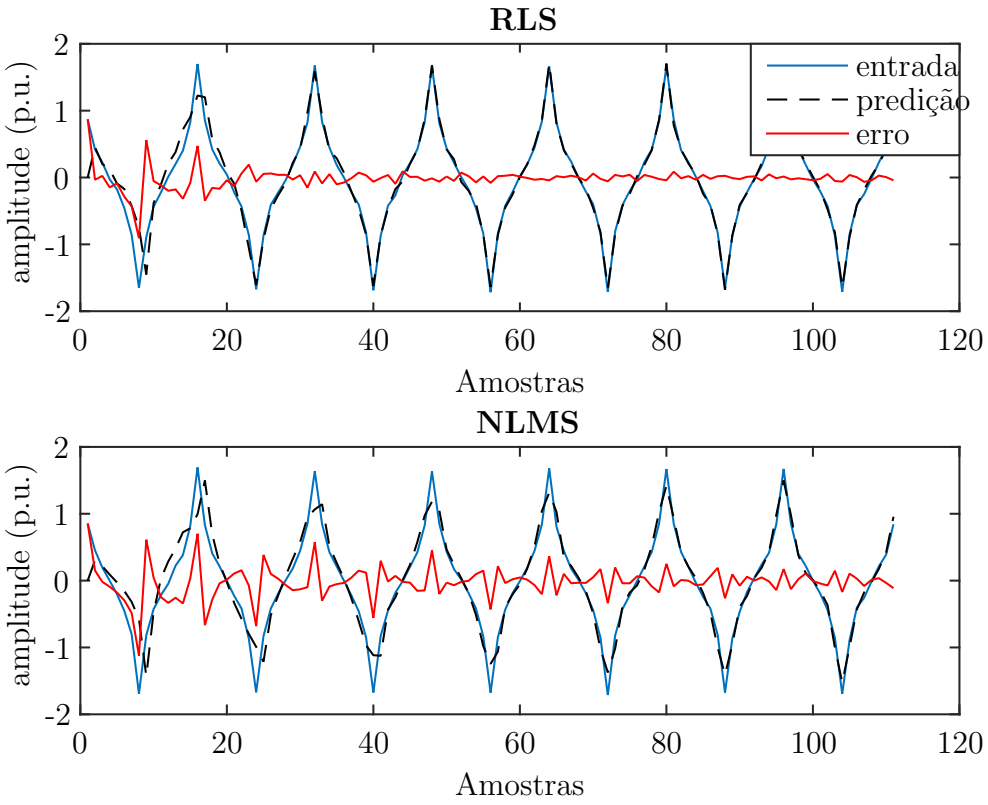


Figura 3.3: Convergência do RLS e NLMS na presença dos harmônicos 1, 3, 5 e 7; M=16

dividindo-o em partes por faixa de frequência, analisando posteriormente.

O teste realizado seguiu este procedimento: foram sorteados harmônicos entre 0 e 15 com uma V.A uniforme, desta frequência foram geradas outras duas, $f_1 = f_0 + U_{0,1}$ e $f_2 = f_0 + U_{0,1} + \delta$, sendo $U_{0,1}$ uma V.A uniforme com amplitude 0.1. Os resultados podem ser vistos na tabela 3.3.

	RLS	LMS
δ	erros(%)	erros (%)
0.05	36,8	51,0
0.1	25,8	50,3
0.2	7,3	43,5
0.4	1,5	25,8
0.6	0,5	14,8

Tabela 3.3: Tabela com os erros para classificação de frequências próximas

3.5 Filtragem de valores

Recuperando alguma equações do capítulo de referências, foi apresentado que o modelo estocástico do sinal Auto-Regressivo era da forma:

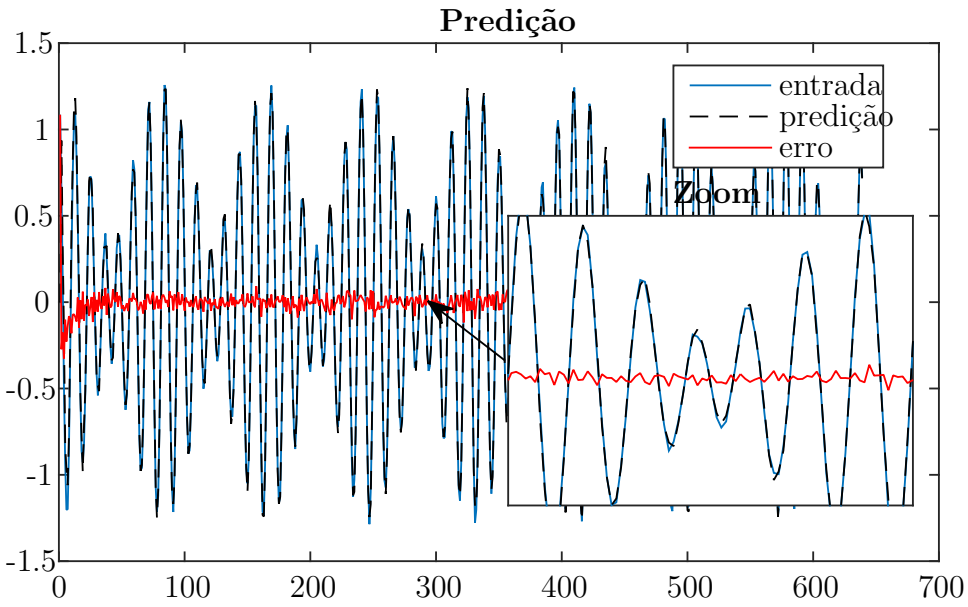


Figura 3.4: Efeito de batimento em frequências muito próximas

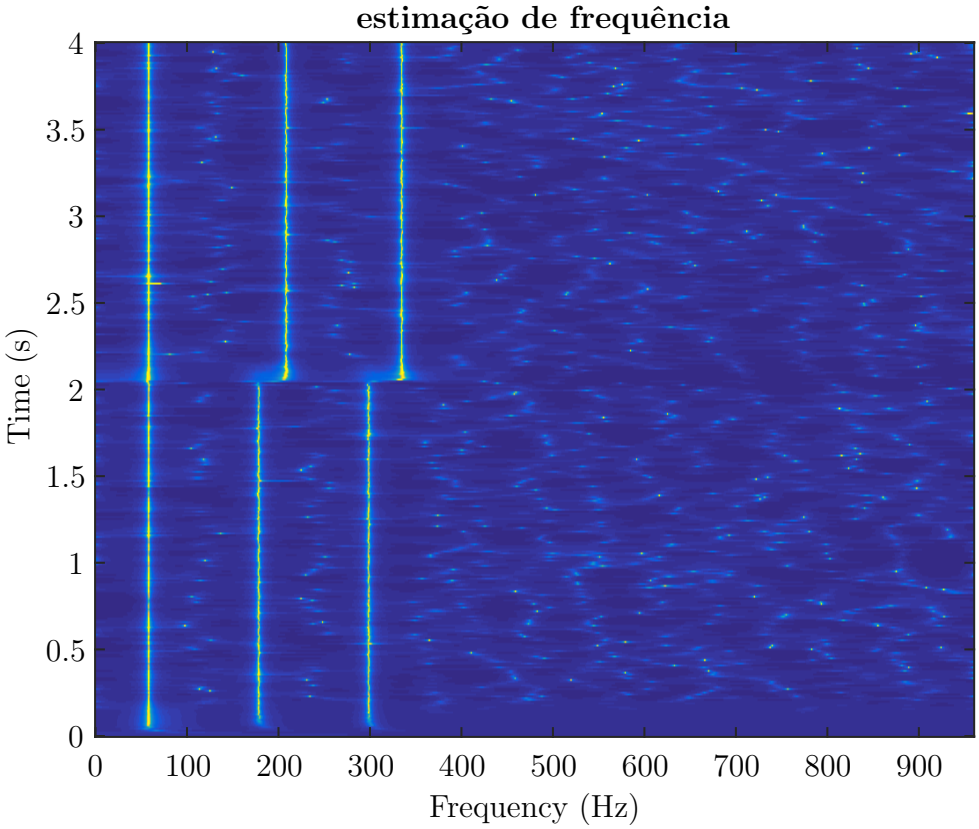


Figura 3.5: Imagem com estimação de amplitudes e frequências no tempo usando RLS

$$u[k] = \sum_{m=1}^{M} w_m u[k-m] + \xi[n] \tag{3.3}$$

Sendo ξ ruído branco. Desta maneira pode-se desenvolver a equação para obter a transformada Z de $u[n]$:

$$U(z) = \sum_{m=1}^{M} w_m U(z) z^{-m} + \Xi(z) \tag{3.4}$$

$$\frac{U(z)}{\Xi(z)} = \frac{1}{1 - \sum_{m=1}^{M} w_m z^{-m}} \tag{3.5}$$

A equação 3.5 nos mostra que o a porção $Q(z) = 1 - \sum_{m=1}^{M} w_m z^{-m}$ determina as características espectrais de $U(w)$ uma vez que ξ é ruído branco e tem densidade de potência constante para todo o espectro. Calculando $Q(w)^{-1}$ pode-se obter uma estimativa da energia naquele ponto. Assim se calcularmos $Q(w_{root})$ para todas as raízes, é possível observar quais tem maior energia e quais tem menor. Desta forma está permitido até mesmo descartar a análise mais profunda de certas componentes que serão acusadas na estimação de frequências. Este tipo de análise inclusive é usado em processamento de fala para identificar formantes de fonemas, e por si só já um método de estimação espectral paramétrico.

Podemos ver nas imagens 3.5 e 3.6 que a estimação feita desta forma para o RLS acusa muito mais frequências que não são reais de maneira aparentemente aleatória no decorrer do tempo, entretanto, ele converge muito mais rápido que o NLMS, que apresenta convergência suave e com pouco ruído. Em ambos os casos temos a presença do quinto e terceiro harmônicos que sofrem uma mudança brusca na frequência em t=2s.

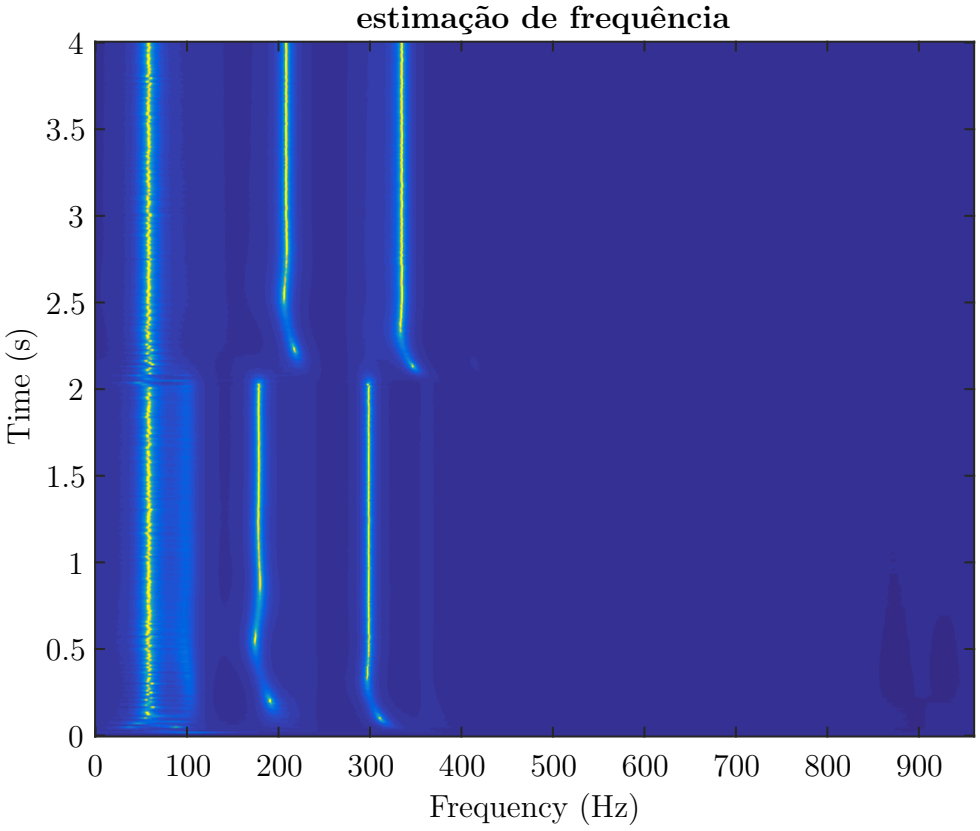


Figura 3.6: Imagem com estimação de amplitudes e frequências no tempo usando NLMS

3.6 teste de rastreio de frequências

Faremos o mesmo teste de rastreio de frequências realizado no artigo citado. Usamos M=64, e $fs = 64f_0$. Este teste consiste em um sinal com cinco componentes senoidais, três harmônicos da fundamental e dois inter-harmônicos na seguinte forma:

$$\begin{aligned} y(t) = A_1 sin(w_0 t + 30^o) + A_2 sin(3w_0 t + 70^o) + A_3 sin(5w_0 t + 40^o) + \\ +0.15 sin(w_1 t + 135^o) + 0.1 sin(w_2 t + 175^o) + \xi(t) \end{aligned} \tag{3.6}$$

Onde $A_1 = 1$, $A_2 = 1/3$, $A_3 = 1/5$, $w_0 = f_0 2\pi$ e $f_0 = 60\,Hz$. Para os inter-harmônicos $f_1 = 152\,Hz$ e $f_3 = 266\,Hz$. E $\xi(t)$=rand(t).

- Em $t = 1.0\,s$ as amplitudes dos harmônicos mudam para 0.8, 0.2 e 0.3 respectivamente.
- Em $t = 2.5\,s$, f_0 vai para 59.8 Hz.
- em $t = 4.0\,s$, f_0 vai para 60.3 Hz e os harmônicos voltam a 1.0, 1/3 e 1/5 respectivamente.
- seguimos simulando até 5.5 s.

Os resultados podem ser vistos na figura abaixo:

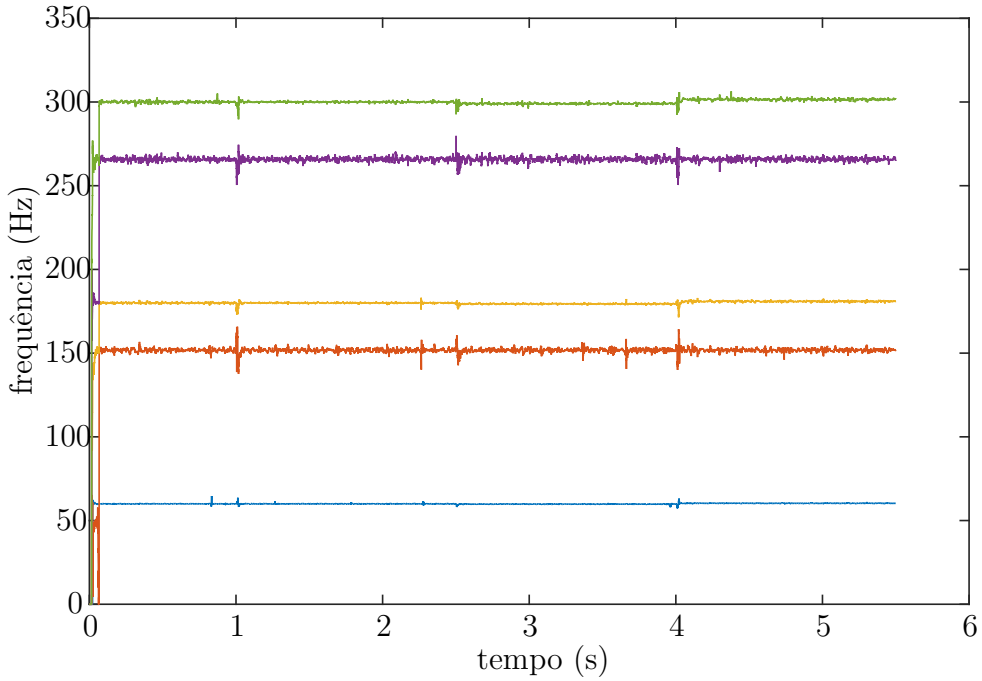


Figura 3.7: Rastreio com teste do artigo utilizando RLS

Para o cálculo das raízes foi utilizada a função roots do MATLAB. Esta função não nos permite usar valores antigos como pontos inicias de busca pelas novas raízes, e no caso do RLS já vimos que este tem uma convergência bastante turbulenta, por assim dizer. Isso tudo faz com que muitas vezes uma mesma raiz seja calculada em ordens diferentes. Assim é necessário plantear um pequeno classificador muito simples para que se possa fazer o rastreio. Foi utilizado um classificador por distância, no qual utiliza-se os valores antigos presentes no mesmo para determinar qual das novas amostras pertence à posição 'n' do classificador. A cada iteração analisamos as novas estimações $\hat{f}_i[k]$ (frequência i da iteração k) e as colocamos como uma nova evidência para a frequência mais próxima em valor absoluto no vetor de estimação global. E para reduzir o ruído de estimação, usamos uma soma convexa das estimativas antigas com a mais recente, da mesma maneira feita com o PLL.

$$\hat{fglobal}_n = \hat{fglobal}_{n-1}(1-\lambda) - [argmin_{\hat{f}_i[k]}(|\hat{fglobal}_{n-1} - f_i[k]|)]\lambda \tag{3.7}$$

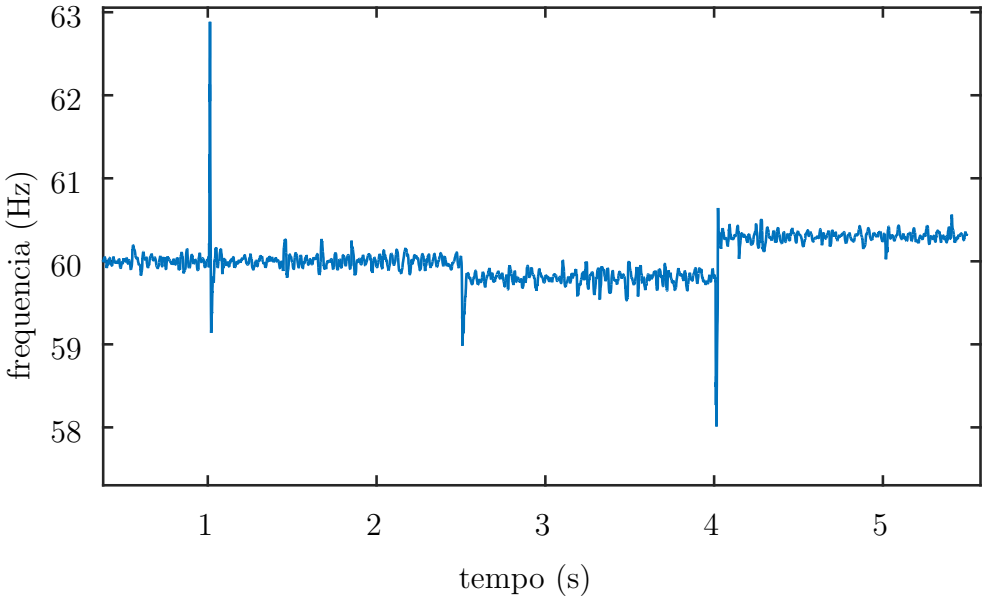


Figura 3.8: Rastreio com teste do artigo utilizando RLS, componente fundamental

3.7 complexidade computacional

A complexidade computacional do RLS é de cerca de $3M^2 + 4M$ somas e multiplicações por iteração[14], enquanto que a do NLMS é de cerca de $4M$ somas e multiplicações. Apesar de a ordem de complexidade do NLMS ser bem menor, entre as operações do RLS há uma série de multiplicações e somas de matrizes, que podem ser otimizadas com computação em paralelo para determinados dispositivos. Então é de se rever onde vai ser aplicado o algoritmo, dependendo do hardware usado, o custo do RLS pode não ser tão mais alto que o do NLMS.

3.8 Conclusões

O algoritmo apresentado tem grande capacidade de estimação para determinados contextos. Por exemplo, com os harmônicos relativamente separados e com amplitude parecida, praticamente não há erros na estimação tanto no NLMS quanto no RLS. Claro que este contexto não é o que se vai encontrar muitas das vezes, mas pode-se fazer ajustes de taxa de amostragem e eliminação de algumas componentes caso seja necessário, tudo depende da natureza do problema. Em situações mais críticas, com grande desnível dos harmônicos e proximidade dos mesmos, o RLS acaba levando grande vantagem. Escolher um método ou o outro depende do hardware disponível e também da proposta desenvolvida. Foi mostrado ainda que o NLMS faz uma estimação mais estável dos coeficientes e consequentemente das raízes e harmônicos, a qual pode ser aproveitado para uma implementação online.

Capítulo 4

Estrutura final

Agora utilizando todos os métodos estudados até aqui, será desenvolvida uma estrutura capaz de estimar as frequências presentes no sinal e suas relevâncias, e em seguida rastrear estas componentes por tempo indeterminado. Para tanto se fará uso da predição linear do capítulo anterior e posteriormente da estrutura PLL-Multitaxa.

4.1 PLL-M em partículas

Foi realizada, a cargo de comparação, uma simulação com sinal nas mesmas condições da presente no capítulo anterior.

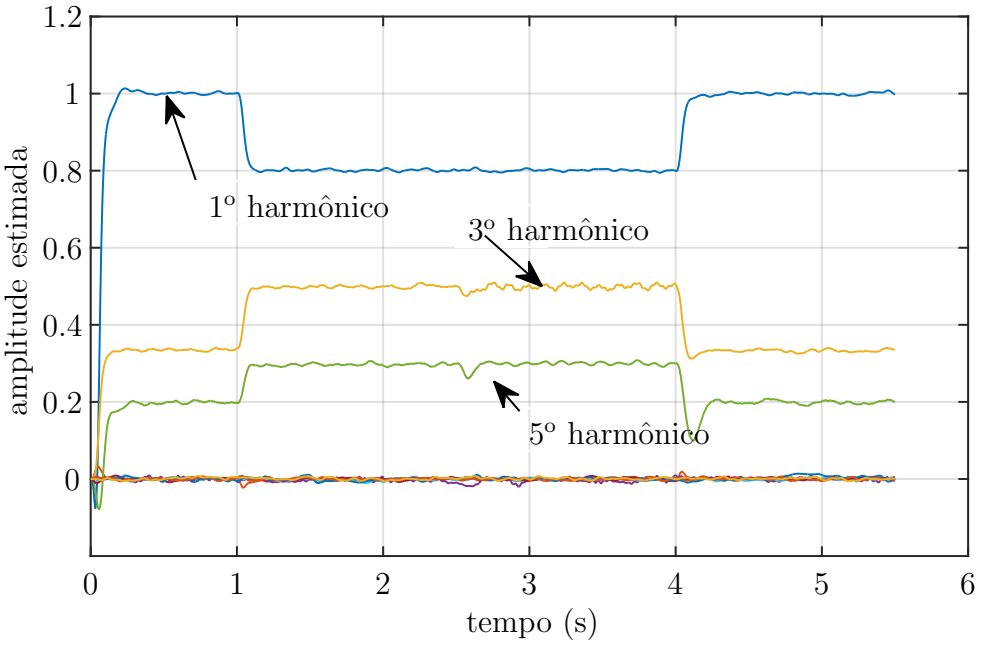


Figura 4.1: Estágio de estimação de frequências e ordem do sistema

4.1.1 Estimação de Frequências

O RLS será usado para calcular os coeficientes w_m de predição linear, faremos uma varredura da porção inicial do sinal aproveitando a grande velocidade de convergência. Para lidar com a desordem da estimação causada pelo RLS, utilizaremos

o classificador do capítulo anterior, com uma adição: vamos filtrá-las de acordo com a variância das últimas N amostras. Foi observado que as frequências corretas variam muito pouco dentro do classificador, diferente dos falsos positivos. Nos testes observamos diferença de pelo menos uma ordem de magnitude entre a variância de frequências corretas e falsas.

Passos para estimar as frequências:

1. Usar o RLS para estimar $\boldsymbol{w}$.

2. Calcular as raízes de $Q(z) = 1 - \sum_{m=1}^{M} w_m \, z^{-m}$

3. Calcular as frequências e módulos usando as equações [obs: colocar as equações da seção 2].

4. Usar o classificador e os estimadores já mencionados para encontrar as frequências, calcular a variância destas frequências e filtrá-las, ficando com os menores valores.

5. Enviar as frequências encontradas para inicializar o PLL.

4.1.2 PLL-Multitaxa

A estrutura usada será idêntica à vista no final do capítulo 3. Tendo como única alteração as constantes, que serão $\mu = \{100, 5000, 400\}$.

4.1.3 simulação e análise qualitativa

Realizaremos a mesma simulação do artigo [5], entretanto com algumas alterações para forçar mais a estabilidade do algoritmo, em $t = 2.5\,s$ aplicaremos um degrau de 1 Hz na frequência fundamental, ao invés de 0.2 Hz, e o ruído presente será de variância $\sigma = 0.05$ ao invés de 0.01. Usaremos M=64, e $fs = 128 f_0$, entretanto a frequência de amostragem será abaixada em 2 vezes para a estimação das frequências para diminuir o tempo de computação, uma vez que neste problema em geral não se investiga frequências acima do 15º harmônico. Usaremos coeficiente de esquecimento igual a 0.92 para o RLS. Escolheremos também rastrear as componentes julgadas mais energéticas pelo estimador. Para o classificador utilizaremos $\lambda = 0.1$ na classificação de frequências e umbral da variância igual a 5, ou seja, somente rastrearemos as componentes cuja variância das últimas 100 amostras for menor que 5.

Rodamos o RLS por meio segundo de sinal, onde devemos admitir que este mantém suas características estatísticas.

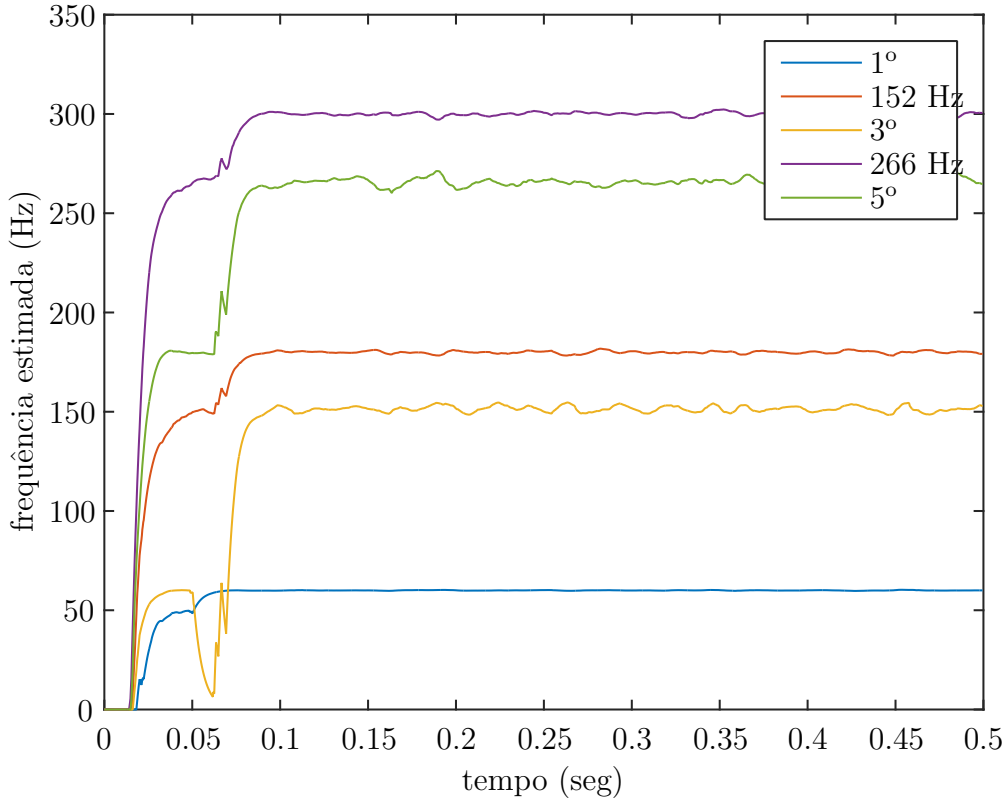


Figura 4.2: Estágio de estimação de frequências e ordem do sistema

Podemos observar na figura 4.2 o estágio de estimação de frequências e ordem do sistema, identificando as cinco componentes desejadas. Observa-se também que ele converge para estas cinco componentes em aproximadamente 100 ms. A partir deste momento já temos os valores corretos estabilizados e adequados para passarmos ao próximo estágio, de rastreio de frequência e amplitude.

No rastreio de amplitudes vemos que o PLL segue as componentes de maneira rápida, eficaz e estável, apesar de mudanças na frequência fundamental, na amplitude dos harmônicos e quando os dois ocorrem ao mesmo tempo. Entretanto observa-se um batimento nas estimações da frequência errante de 152 Hz e também no 5º e 3º harmônicos depois do degrau de 1 Hz en 2.5 s. Isto se deve à principalmente a interferências, uma vez que eles acabam ficando próximos, mas não leva a offset na estimação.

Percebe-se que a estimação fina de frequências do PLL-M que é muito mais precisa e estável que a do primeiro estágio, embora esta não seja global. Se por exemplo, uma destas componentes deixa de existir, some do sinal, o PLL não tem condições de acusar isto, ou se uma nova aparece no ligar da antiga, o PLL também não poderia identificá-la.

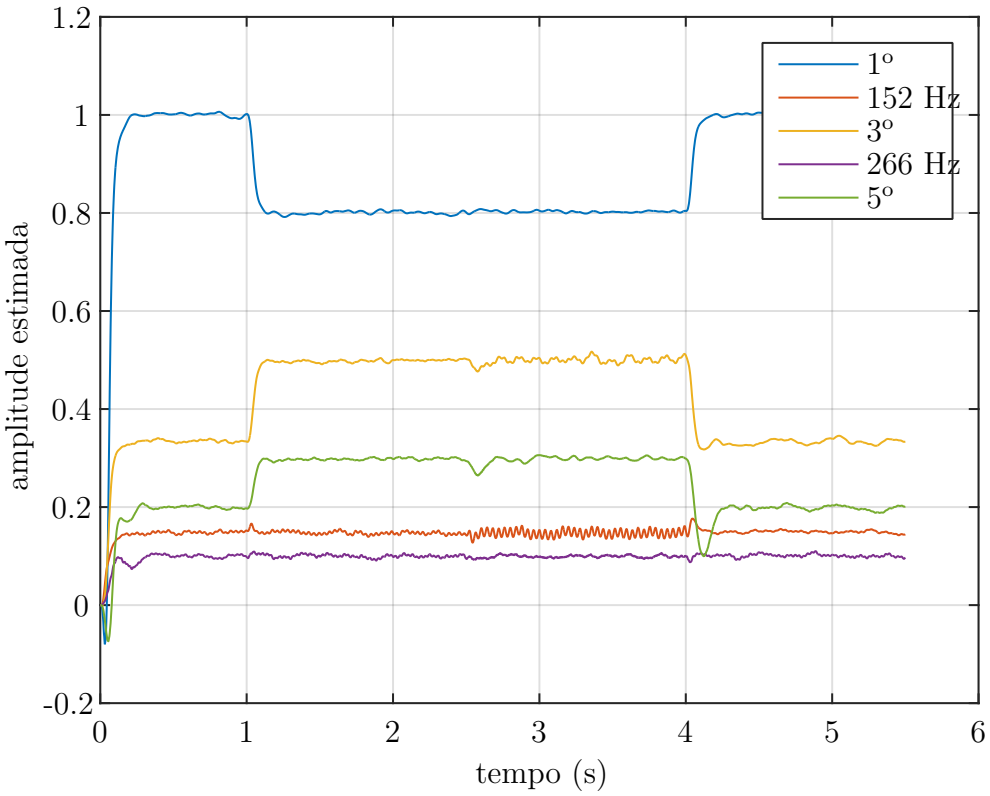


Figura 4.3: rastreio de amplitudes do PLL-M

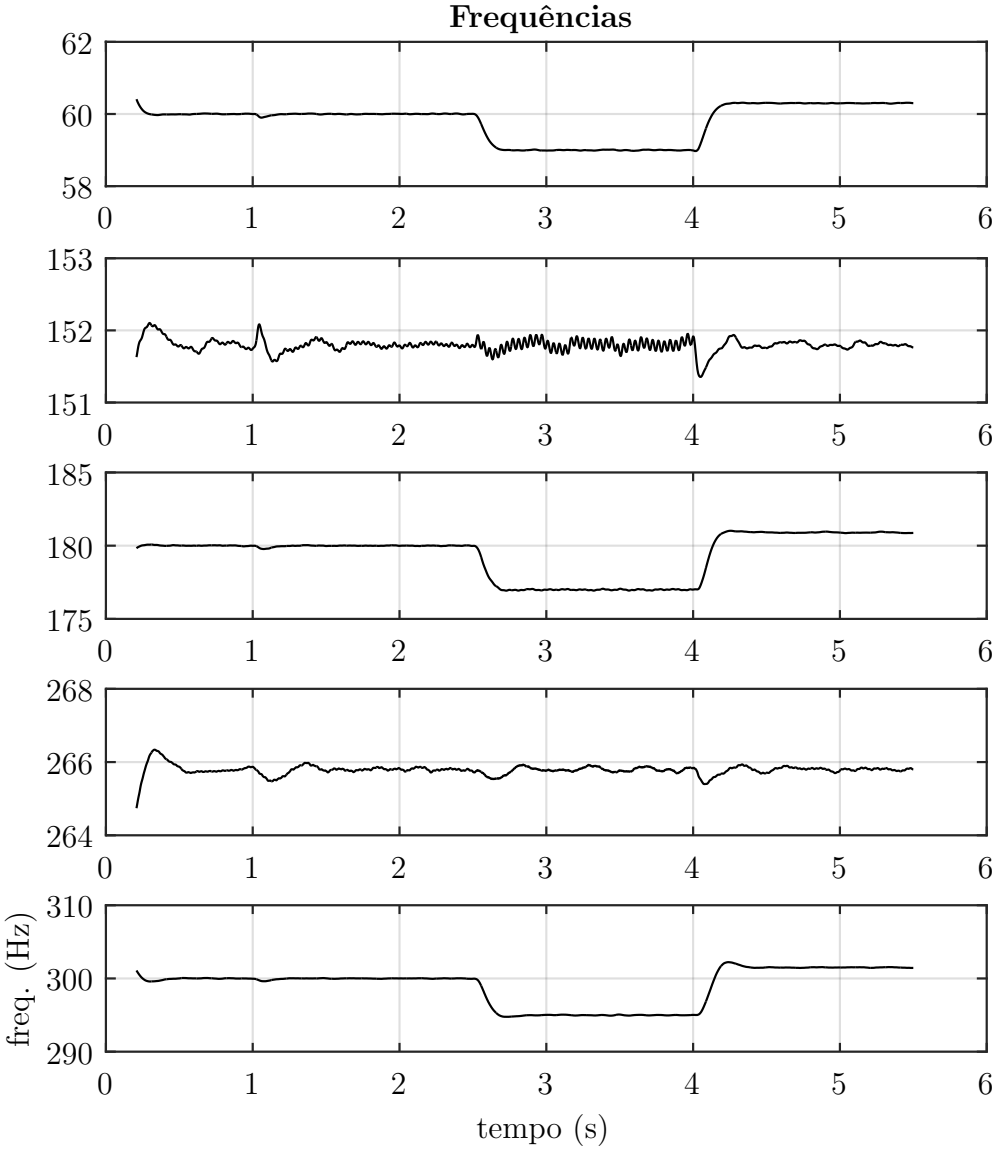


Figura 4.4: Rastreio de frequências do PLL-M

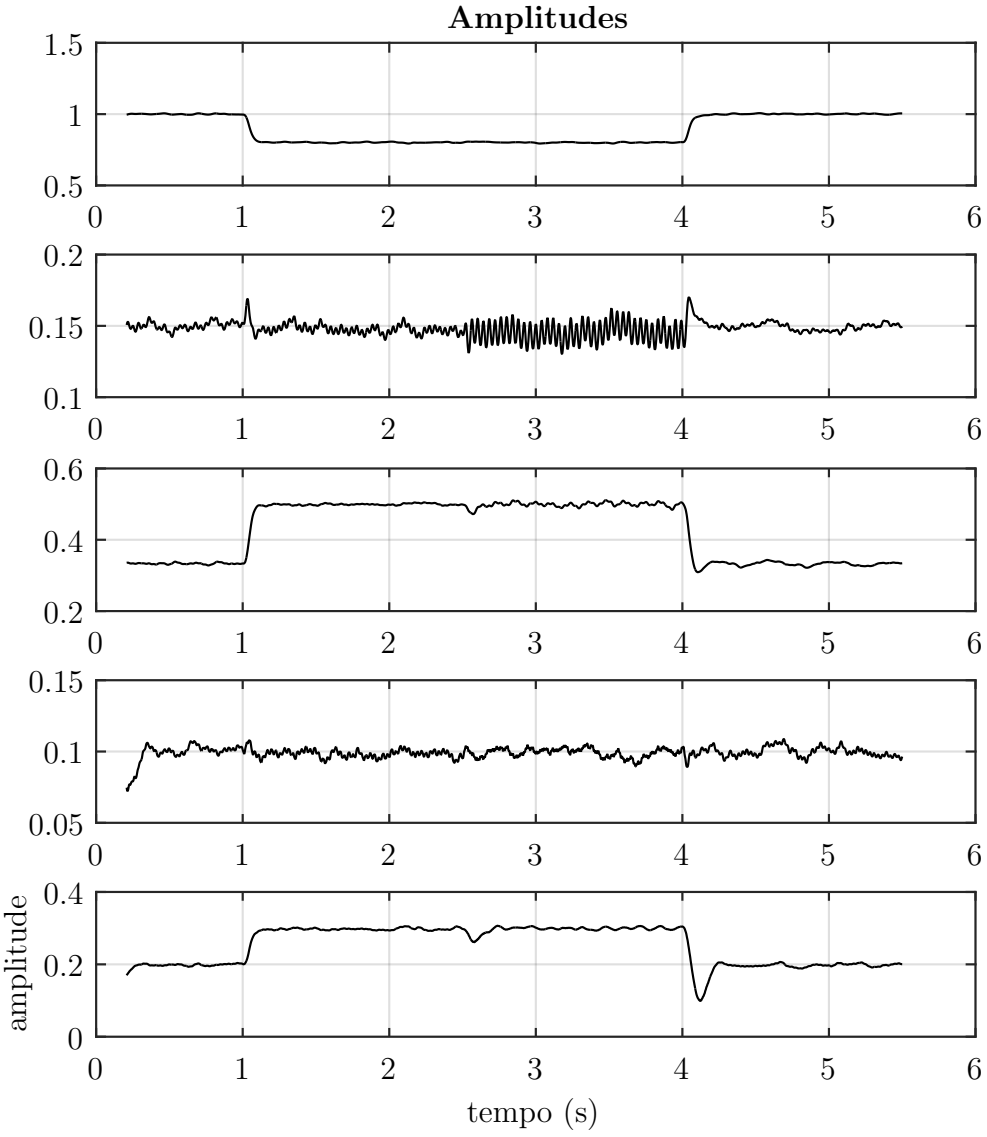


Figura 4.5: Rastreio de frequências do PLL-M

4.2 Conclusões

Neste trabalho apresentamos o que é e o problema que representa hoje a estimação espectral, bem como a fundamentação teórica matemática por trás do tema. Discorremos sobre métodos paramétricos e não paramétricos. Nos focamos principalmente em simular e apresentar alternativas para duas metodologias relativamente recentes [2][3] publicadas nos últimos 10 anos.

Apresentamos as virtudes e pontos fracos de cada um dos métodos, ao final do trabalho é proposto um método capaz de unir os pontos fortes de cada um. Um método que aproveita a robustez e simplicidade computacional no rastreio do PLL-M e o poder de identificação global dos métodos baseados em modelos AR. Entretanto, mesmo dentro da metodologia aplicada ao capítulo final, há muito o que se desenvolver. Um método mais robusto seria capaz de identificar o surgimento

de novas componentes e o desaparecimento de outras, fazendo uma administração inteligente dos recursos computacionais.

Apêndice A

Appendix Title

That's all, folks!

Bibliografia

[1] Saleh R Al-Araji, Zahir M Hussain e Mahmoud A Al-Qutayri. *Digital Phase Lock Loops*. Springer, 2006.

[2] Lupércio F Bessegato et al. "On-Line Process Control using Attributes with Misclassification Errors: An Economical Design for Short-Run Production". Em: *Communications in Statistics-Theory and Methods* 41.10 (2012), pp. 1813–1832.

[3] Janison Rodrigues de Carvalho et al. "A PLL-based multirate structure for time-varying power systems harmonic/interharmonic estimation". Em: *IEEE Transactions on Power Delivery* 24.4 (2009), pp. 1789–1800.

[4] Janison Rodrigues de Carvalho et al. "Estimação de harmônicos/interharmônicos: uma abordagem multitaxa". Em: (2008).

[5] Gary W Chang, Cheng-I Chen e Quan-Wei Liang. "A two-stage ADALINE for harmonics and interharmonics measurement". Em: *IEEE Transactions on Industrial Electronics* 56.6 (2009), pp. 2220–2228.

[6] Paulo SR Diniz. *Adaptive filtering*. Springer, 1997.

[7] Albert Einstein. "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]". Em: *Annalen der Physik* 322.10 (1905), pp. 891–921. DOI: http://dx.doi.org/10.1002/andp.19053221004.

[8] Simon S Haykin. *Adaptive filter theory*. Pearson Education India, 2005.

[9] Xuedong Huang et al. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001.

[10] Sanjit Kumar Mitra e Yonghong Kuo. *Digital signal processing: a computer-based approach*. Vol. 2. McGraw-Hill New York, 2006.

[11] Petre Stoica, Randolph L Moses et al. "Spectral analysis of signals". Em: (2005).

[12] Alireza K Ziarani e Adalbert Konrad. "A method of extraction of nonstationary sinusoids". Em: *Signal Processing* 84.8 (2004), pp. 1323–1346.