TRABALHO - IDENTIFICAÇÃO DE SISTEMAS

Raphael Timbó Silva

Professor: Daniel Castello

Rio de Janeiro
Janeiro de 2017

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

FIR	Finite Impulse Response, p. 1
FRF	Função de Resposta em Frequência, p. 2
SNR	Signal to Noise Ratio, p. 4

Capítulo 1

Introdução

O presente trabalho tem por objetivo apresentar os resultados e conclusões referentes ao projeto final da disciplina Identificação de Sistemas.

O trabalho consiste na análise de um sistema através do projeto de um filtro adaptativo FIR (Finite Impulse Response).

1.1 Sistema utilizado

O sistema utilizado é mostrado na fig. 1.1.

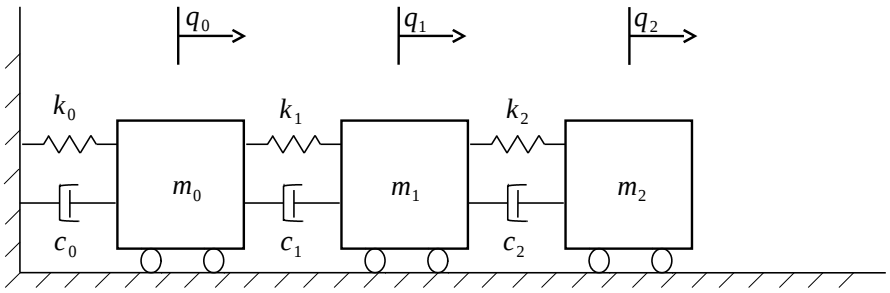


Figura 1.1: Sistema utilizado na análise.

Para este sistema temos que a energia cinética é:

$$T = \frac{1}{2}[m_0\dot{q_0}(t)^2 + m_1\dot{q_1}(t)^2 + m_2\dot{q_2}(t)^2] = \frac{1}{2}\dot{\mathbf{q}}^T(t)M\dot{\mathbf{q}}(t) \tag{1.1}$$

onde

$$\mathbf{q(t)} = [q_0(t)\ q_1(t)\ q_2(t)]^T$$

é o vetor de configuração e

$$M = \begin{bmatrix} m_0 & 0 & 0 \\ 0 & m_1 & 0 \\ 0 & 0 & m_2 \end{bmatrix}$$

é a matriz de massa do sistema.

A energia potencial tem a expressão:

$$\begin{aligned} V &= \frac{1}{2}[k_0 q_0(t)^2 + k_1(q_1(t) - q_0(t))^2 + k_2 q_2(t)^2] \\ &= \frac{1}{2}[(k_0 + k_1)q_0(t)^2 + (k_1 + k_2)q_1(t)^2 + (k_2)q_2(t)^2 - 2k_1 q_0(t)q_1(t) - 2k_2 q_2(t) \\ &= \frac{1}{2}\dot{\mathbf{q}}^T(t) K \dot{\mathbf{q}}(t) \end{aligned} \tag{1.2}$$

onde

$$K = \begin{bmatrix} k_0 + k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}$$

é a matriz de rigidez do sistema.

Para o sistema utilizado temos que $m_i = 1\ kg$ e $k_i = 1600\ N/m$.

O amortecimento utilizado será o proporcional: $C = \alpha M + \beta K$. Iremos analisar o caso em que $\alpha = 10^{-3}$ e $\beta = 10^{-3}$.

1.2 Resposta do sistema

O sistema em questão possui a FRF (Função de Resposta em Frequência) apresentada na fig. 1.2

Para nossa análise iremos considerar uma força aplicada na massa 2 (m_2) e a medição nesta mesma massa, conforme ilustrado na fig. 1.3. A aplicação da força nessa massa corresponde à FRF que pode ser visualizada no canto inferior direito da fig. 1.2 (input=2 e output=2). A FRF em questão é também mostrada na fig. 1.4.

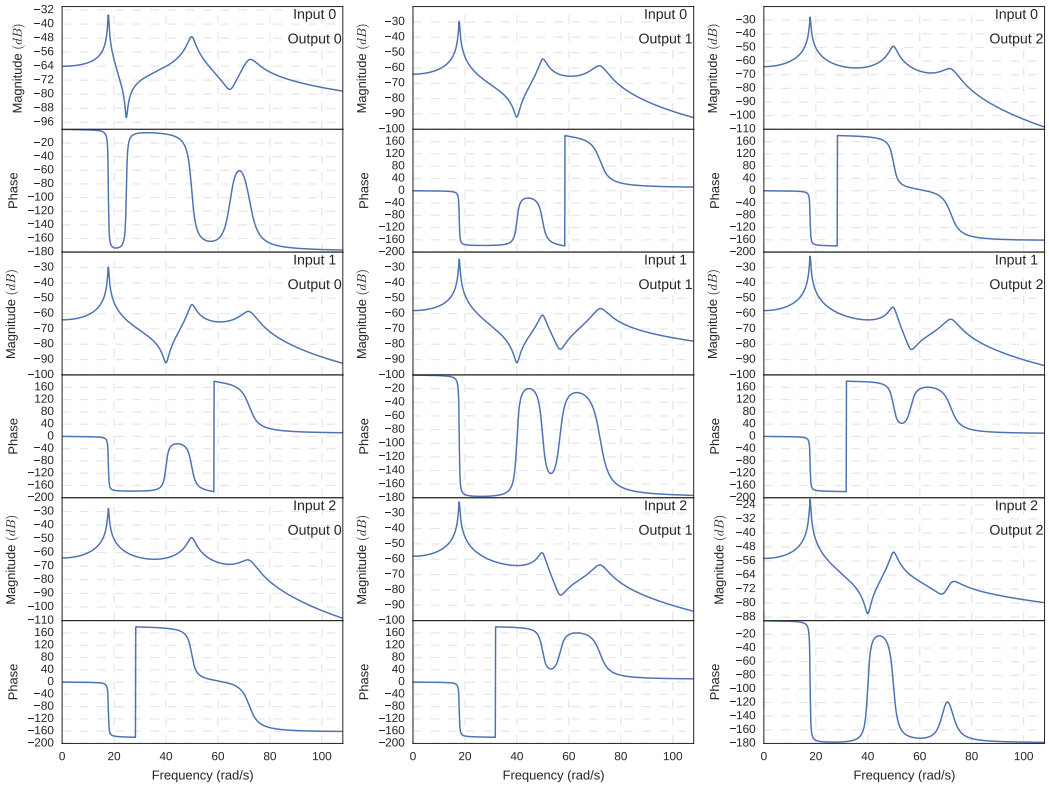


Figura 1.2: FRF para o sistema em análise.

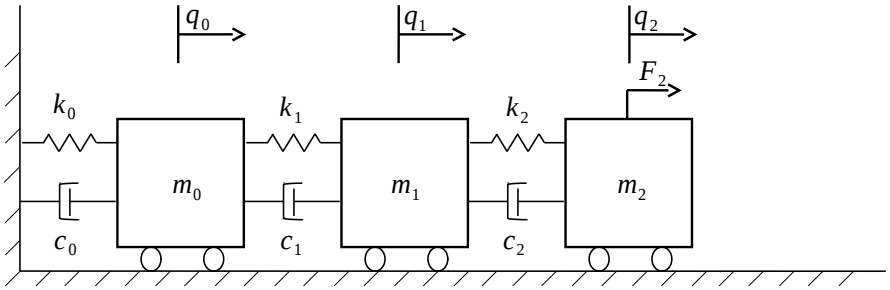


Figura 1.3: Aplicação de força e medição na massa m_2.

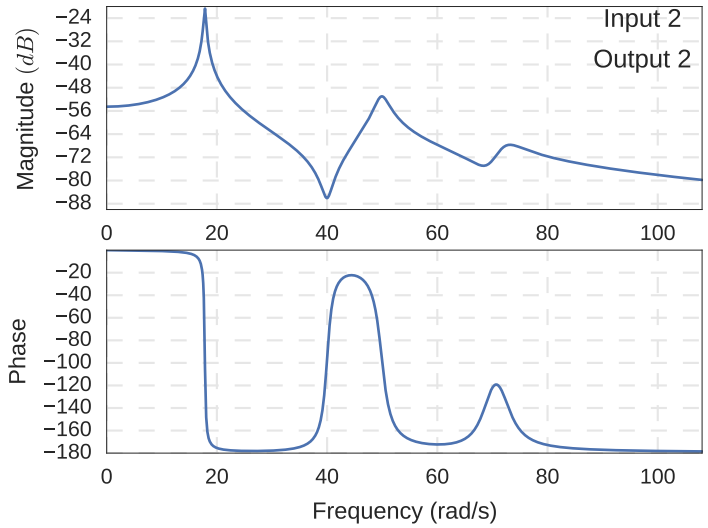


Figura 1.4: FRF para input em m_2 e medição em m_2.

Capítulo 2

Dados Pseudo-Experimentais

2.1 Resposta do sistema no tempo

Para a construção dos dados pseudo-experimentais foram observados os seguintes casos:

Forçamento:

- $F_0(t) = A_0 sin(2\pi f_0 t)$ (Considere $\frac{\omega_1}{2\pi} \leq f_0 \leq \frac{\omega_2}{2\pi}$)
- $F_1(t) = A_1 sin(2\pi f_1 t) + A_2 sin(2\pi f_2 t)$ (Escolha $\frac{0.8\omega_1}{2\pi} \leq f_j \leq \frac{1.2\omega_2}{2\pi}$ e $A_2 = 2A_1$; $j = 1, 2$)
- $F_2(t) =$ ruído branco

Número de amostras N:

- $N = 1000$
- $N = 5000$

Valores para a relação entre sinal e ruído - SNR (Signal to Noise Ratio):

- $SNR = 90$
- $SNR = 50$
- $SNR = 10$

A fig. 2.1 mostra a posição da frequência de excitação para a aplicação da força F_0, em que uma amplitude $A_0 = 1$ foi utilizada.

A fig. 2.2 mostra a resposta no tempo do sistema ao aplicarmos a força F_0 na frequência mostrada na fig. 2.1 para uma amostragem $N = 1000$. Podemos observar que, para $N = 1000$, temos uma excitação de aproximadamente 16 segundos e ainda

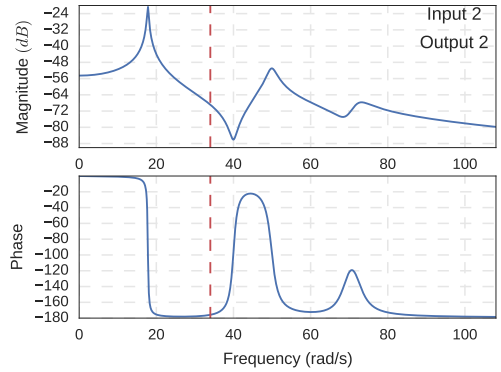


Figura 2.1: Frequência de excitação para a força F_0.

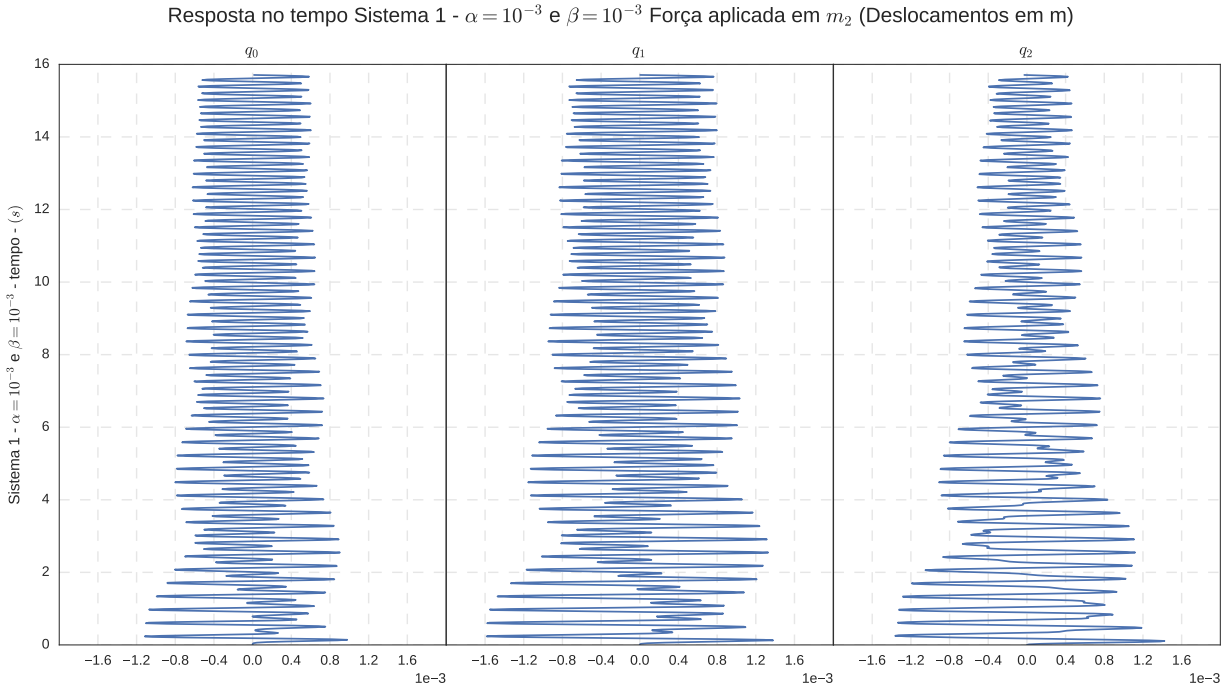


Figura 2.2: Resposta no tempo para a força F_0 com $N = 1000$.

temos algum transiente na resposta no tempo. Também é possível observar que essa parcela apresenta mais que uma frequência de oscilação.

A fig. 2.3 mostra a resposta no tempo para $N = 5000$. Neste caso, o tempo vai até aproximadamente 80 segundos e podemos observar que a parcela transiente é praticamente inexistente após os 20 segundos de excitação. Após esse tempo, é esperado que o sistema oscile apenas na frequência de excitação.

Para a força F_1, a fig. 2.4 mostra as frequências de excitação que foram aplicadas na massa m_2. Podemos notar que, nesse caso, as forças aplicadas estão próximas as frequências naturais do sistema.

A fig. 2.5 mostra a resposta no tempo para F_1 com $A_1 = 1$, $A_2 = 2$ e $N = 5000$. Como esperado, notamos um aumento na amplitude de 1×10^{-3} m para 1×10^{-2} m

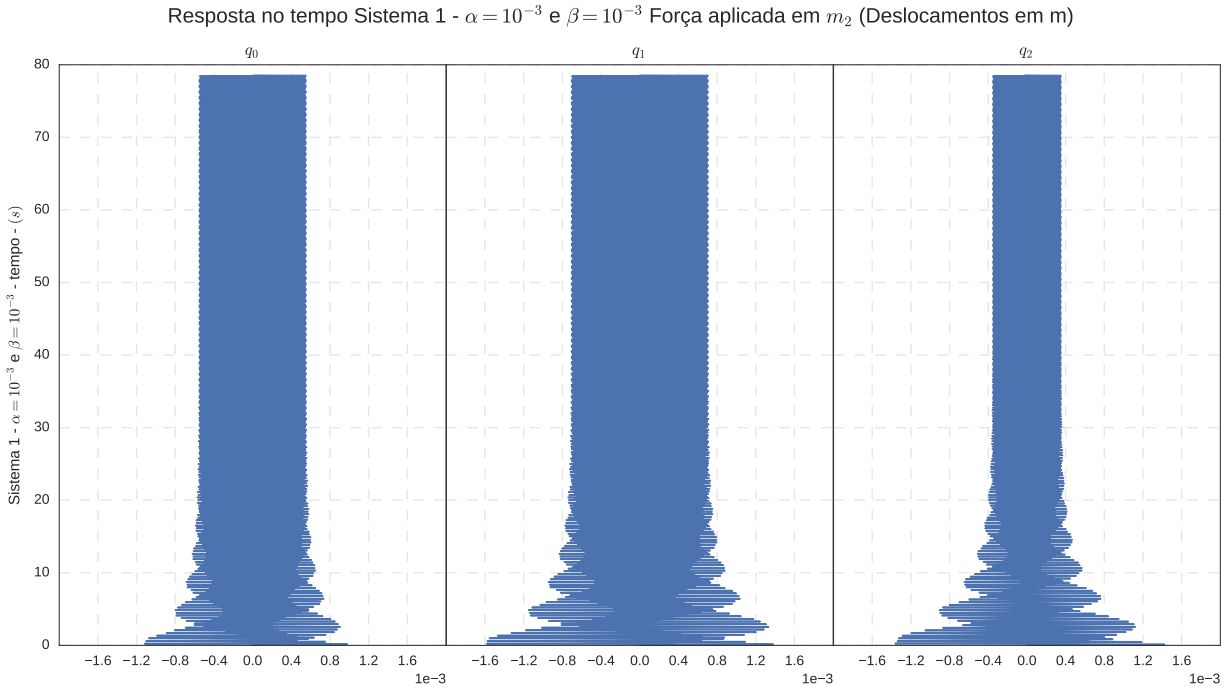


Figura 2.3: Resposta no tempo para a força F_0 com $N = 5000$.

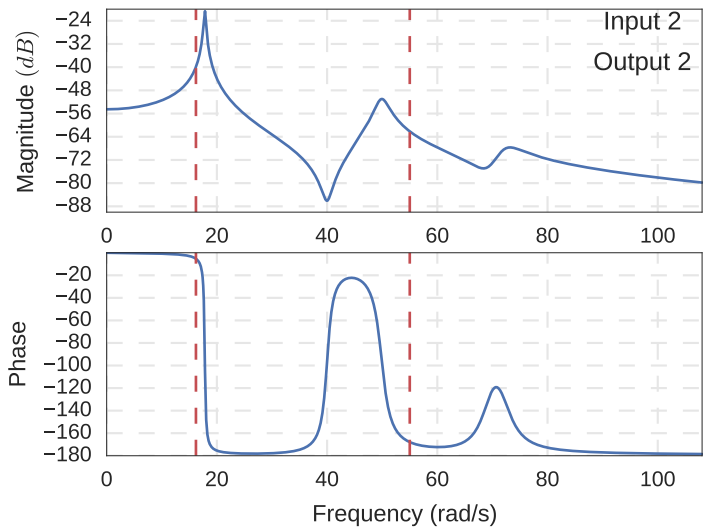


Figura 2.4: Frequência de excitação para a força F_1.

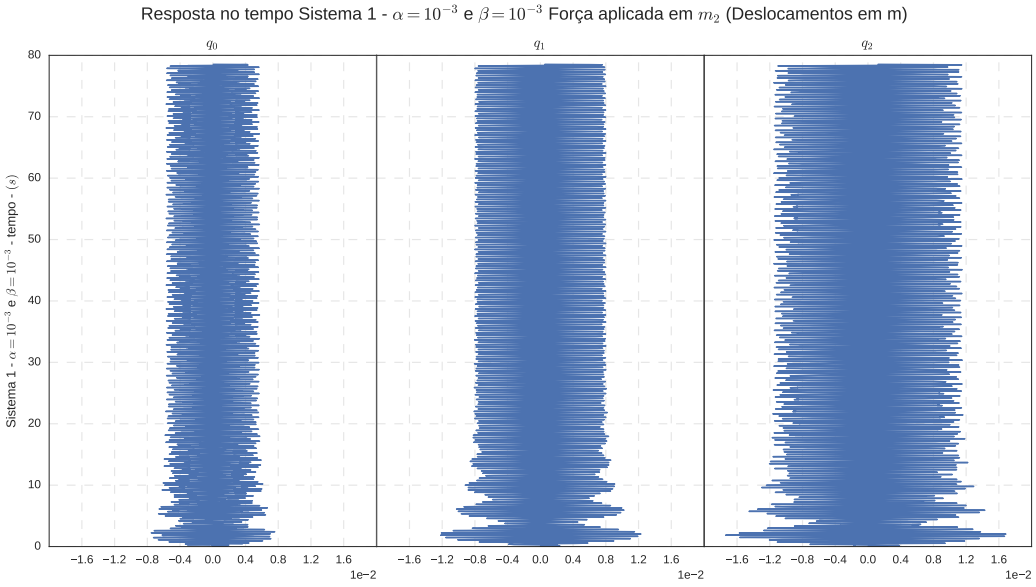


Figura 2.5: Resposta no tempo para a força F_1 com $N = 5000$.

quando comparado à força F_0.

O último caso de forçamento é mostrado na fig. 2.6 onde um ruído branco com variância 1 é aplicado ao sistema.

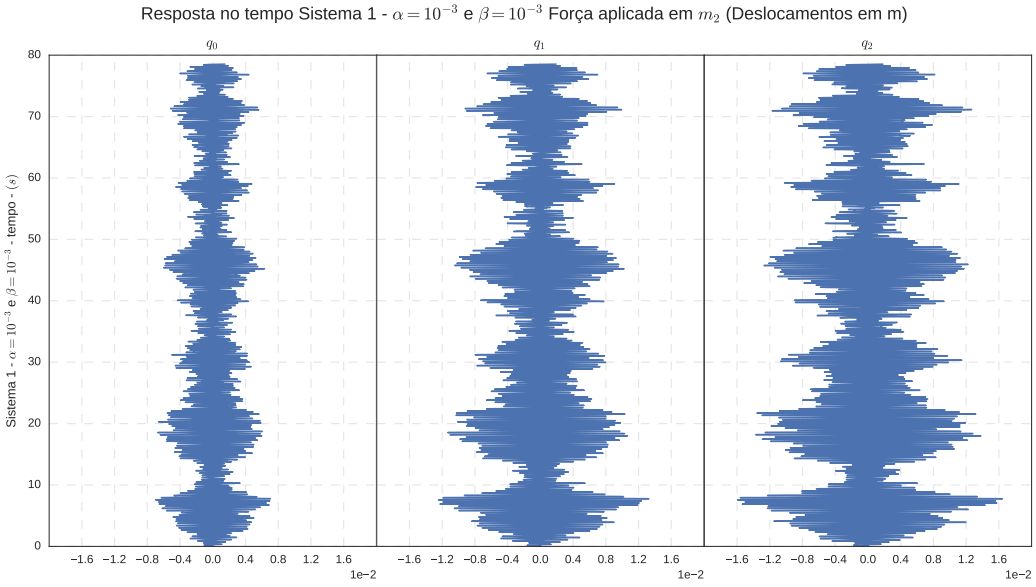


Figura 2.6: Resposta no tempo para a força F_2 com $N = 5000$.

2.2 Adição do ruído

Conforme mostrado no item 2.1, a análise será feita para três diferentes níveis de ruído ($SNR = 90, 50, 10$).

Temos então que o sinal utilizado para o projeto do filtro será:

$$y = y^{ideal} + n \tag{2.1}$$

onde n representa um ruído inserido no sinal.

Para calcularmos a amplitude do ruído inserido 'n' utilizaremos a eq. (2.2).

$$SNR = 20log_{10}\left(\frac{A_s}{A_n}\right) \rightarrow A_n = \frac{A_s}{10^{SNR/20}} \tag{2.2}$$

Abaixo (fig. 2.7 e fig. 2.8), são mostrados alguns resultados comparando o sinal puro e o sinal corrompido para um determinado nível de ruído.

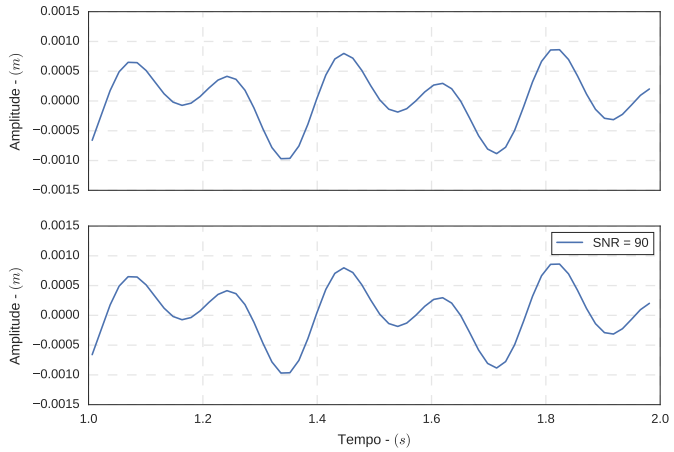


Figura 2.7: Sinal puro e sinal corrompido para F_0 e $SNR = 90$.

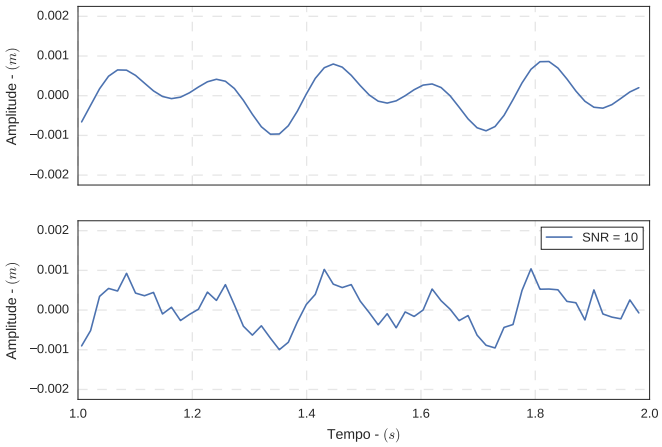


Figura 2.8: Sinal puro e sinal corrompido para F_0 e $SNR = 10$.

Capítulo 3

Projeto do Filtro Adaptativo

Para o algoritmo do filtro adaptativo CASTELLO e ROCHINHA [2] apresentam algumas opções que podem ser utilizadas. No caso do presente trabalho o algoritmo LMS (Least Mean Squares) foi escolhido.

3.1 Algoritmo LMS

Como mostrado por DINIZ [1], a configuração normalmente aplicada para identificação de sistemas com filtros adaptativos é mostrada na fig. 3.1. Nesta configuração temos que $x(k)$ é o sinal de entrada, o sinal de saída do sistema que desejamos identificar é $d(k)$ e a saída do filtro é $y(k)$. Estes sinais são comparados e o erro $e(k)$ é calculado.

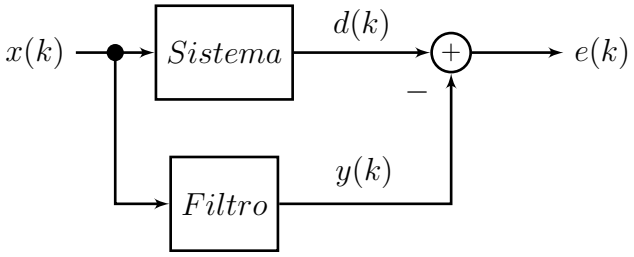


Figura 3.1: Configuração utilizada no algoritmo para filtros adaptativos.

Os valores de saída do filtro são calculados a partir de uma combinação linear dos seus coeficientes e do sinal de entrada, isto é:

$$y(k) = \sum_{i=0}^{N} w_i(k)x_i(k) = \mathbf{w}^T(k)\mathbf{x}(k) \tag{3.1}$$

onde $\mathbf{w}(k)$ representa os coeficientes do filtro.

Para aplicações onde o vetor do sinal de entrada é uma versão atrasada do mesmo sinal, isto é: $x_0(k) = x(k), x_1(k) = x(k-1), ..., x_N(k) = x(k-N)$, $y(k)$ é o resultado da aplicação de um filtro FIR ao sinal de entrada $x(k)$. Neste caso temos:

$$y(k) = \sum_{i=0}^{N} w_i(k)x(k-i) = \mathbf{w}^T(k)\mathbf{x}(k) \tag{3.2}$$

onde $\mathbf{x}(k) = [x(k)\ x(k-1)\ ...\ x(k-N)]^T$.

O MSE (Mean Square Error) pode ser calculado como:

$$\xi(k) = E[e^2(k)] = E[d^2(k) - 2d(k)y(k) + y^2(k)] \tag{3.3}$$

Podemos reescrever a eq. (3.3):

$$\xi = E[d^2(k)] - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \tag{3.4}$$

onde $p = E[d(k)\mathbf{x}(k)]$ é o vetor de correlação cruzada entre o sinal desejado e o sinal de entrada e $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^T(k)]$ é a matriz de correlação do sinal de entrada.

Se $\mathbf{p}$ e $\mathbf{R}$ são conhecidos, podemos encontrar a solução para $\mathbf{w}$ que minimiza ξ.

O gradiente do MSE relativo aos coeficientes é:

$$\mathbf{g}_\mathbf{w} = \frac{\partial\xi}{\partial\mathbf{w}} = \left[\frac{\partial\xi}{\partial w_0}\frac{\partial\xi}{\partial w_1}...\frac{\partial\xi}{\partial w_N}\right] = -2\mathbf{p} + 2\mathbf{R}\mathbf{w} \tag{3.5}$$

Igualando o gradiente a zero encontramos o vetor de coeficientes que minimiza ξ:

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p} \tag{3.6}$$

Se boas estimativas $\hat{\mathbf{p}}$ e $\hat{\mathbf{R}}$ estão disponíveis podemos buscar uma solução:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\hat{\mathbf{g}}_\mathbf{w}(k) \tag{3.7}$$

$$= \mathbf{w}(k) + 2\mu(\hat{\mathbf{p}}(k) - \hat{\mathbf{R}}(k)\mathbf{w}(k)) \tag{3.8}$$

para $k = 0, 1, 2, ...$, onde $\hat{\mathbf{g}}_\mathbf{w}(k)$ representa uma estimativa do gradiente da função objetivo com respeito aos coeficientes do filtro.

Uma possível solução é estimar o gradiente com estimativas instantâneas de $\mathbf{R}$ e de $\mathbf{p}$.

$$\hat{\mathbf{R}}(k) = \mathbf{x}(k)\mathbf{x}^T(k) \tag{3.9}$$

$$\hat{\mathbf{p}}(k) = d(k)\mathbf{x}(k) \tag{3.10}$$

Temos então que a estimativa para o gradiente é:

$$\hat{\mathbf{g}}_{\mathbf{w}}(k) = -2e(k)\mathbf{x}(k) \tag{3.11}$$

Chegamos então ao algoritmo LMS em que a equação de atualização é:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k) \tag{3.12}$$

onde o fator de convergência μ deve ser escolhido em um range que garanta a convergência.

3.2 Implementação do filtro

O filtro será implementado em Python como explicado abaixo.

Uma classe `class LMSFilter()` será criada para o filtro. Essa classe contém uma função de inicialização que possui como argumentos de entrada o número de coeficientes do filtro e o fator de convergência μ. Na criação do filtro os coeficientes são igualados a zero como mostrado no código abaixo.

```
class LMSFilter(object):
  def __init__(self, Nc, mu):
    """
    Iniciar filtro com Nc coeficientes.
    """
    self.Nc = Nc
    self.mu = mu
    # valores iniciais para o filtro w = [0, 0, ..., 0]
    self.w = np.zeros(Nc)
```

O filtro possui uma função `predict(self, x)` que calcula a saída prevista para o filtro baseado no sinal de entrada `x` ($x(k)$). O cálculo é feito conforme a eq. (3.1):

$$y(k) = \sum_{i=0}^{N} w_i(k)x_i(k) = \mathbf{w}^T(k)\mathbf{x}(k) \tag{3.1 revisitada}$$

```
def predict(self, x):
  y = self.w @ x
  return y
```

A atualização do filtro é feita pela função `update(self, d, x)` considerando a eq. (3.12):

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k) \tag{3.12 revisitada}$$

```
def update(self, d, x):
  """
  Atualizar filtro baseado no sinal de entrada x
  e no valor desejado d.
  """
  y = self.w @ x
  e = d - y
  self.w += 2*self.mu * e * x
```

Capítulo 4

Resultados e Discussões

4.1 Resultados para F_0

4.1.1 Adaptação do filtro

Os resultados da adaptação do filtro para uma força $F_0(t) = A_0 sin(2\pi f_0 t)$ com $N = 1000$ e $SNR = 90$ são mostrados na fig. 4.1. O número de coeficientes do filtro (N_c) e o fator de convergência (μ) utilizados são apresentados na figura.

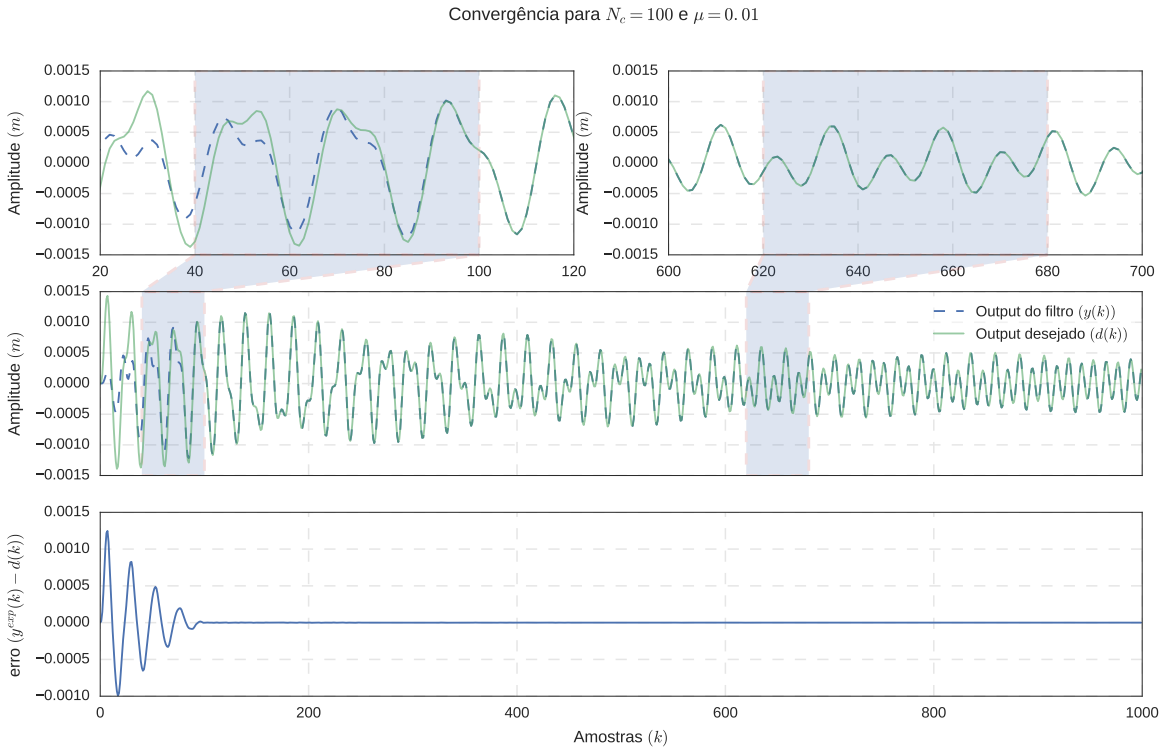


Figura 4.1: Evolução do filtro para $F = F_0$, $N = 1000$ e $SNR = 90$.

Para o caso do seno puro podemos o filtro tem uma convergência rápida e com menos de 100 iterações o erro já é próximo de zero.

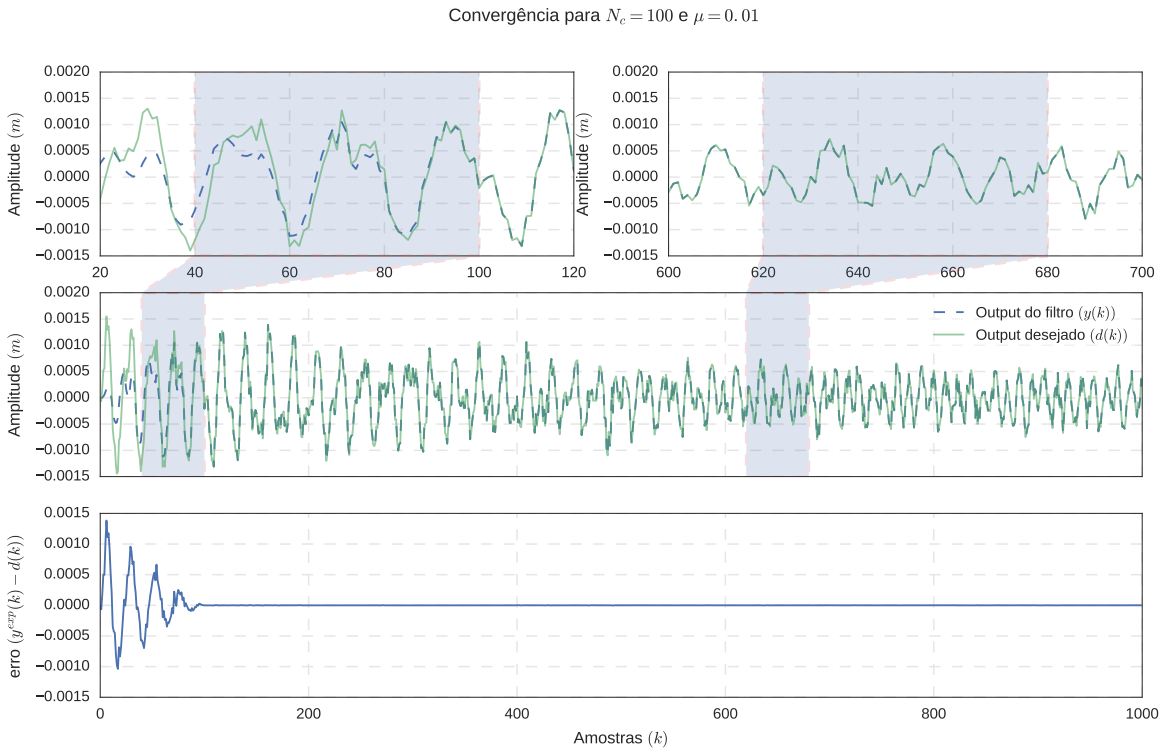


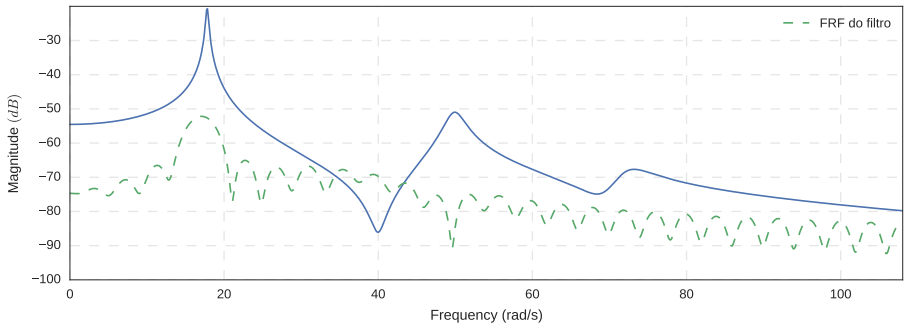
Figura 4.2: Evolução do filtro para $F = F_0$, $N = 1000$ e $SNR = 10$.

Na fig. 4.2 são apresentados os resultados para $N = 1000$ e $SNR = 10$. Podemos observar que, mesmo com um nível de ruído mais elevado, o algoritmo apresentou uma convergência rápida.

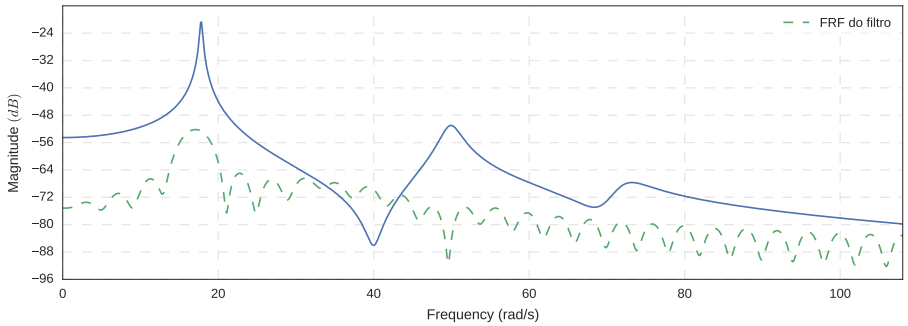
4.1.2 FRF do filtro

Na fig. 4.3a é apresentada a FRF do filtro obtido com $SNR = 90$ considerando o último vetor de coeficientes, e na fig. 4.3b é apresentada a FRF considerando o vetor obtido através do valor esperado dos coeficientes tomando por base as observações da segunda metade do vetor de dados. A fig. 4.4 apresenta os resultados para $SNR = 10$.

Podemos observar que FRF do filtro obtido não apresenta bons resultados, aproximando-se do valor esperado apenas na frequência de excitação ($34\,\mathrm{rad\,s^{-1}}$). Podemos notar um aumento na amplitude próxima à primeira frequência natural do sistema, mas o valor não se aproxima do esperado. Outro ponto importante é que, para este caso, a FRF do filtro não se mostrou sensível ao nível de ruído. A utilização do último vetor de coeficientes w ou do valor esperado para a segunda metade do vetor de dados também não teve impacto significativo na resposta.

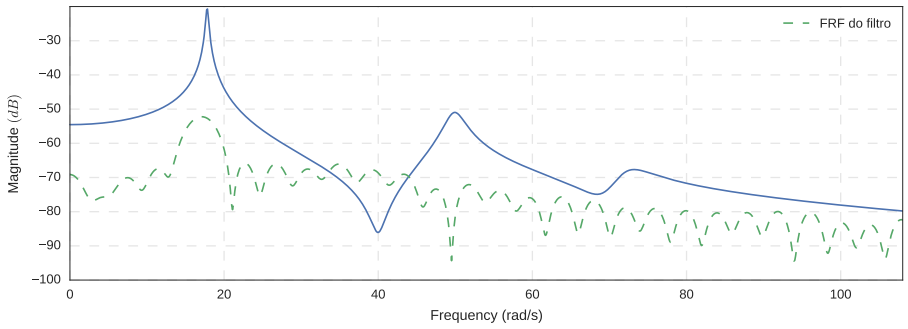


(a) último vetor w

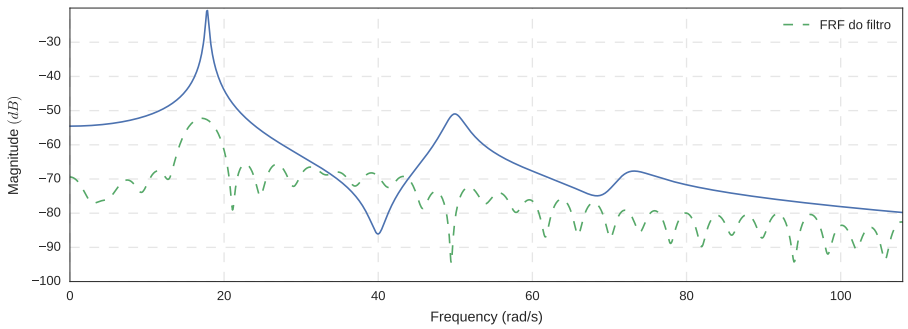


(b) valor médio de w

Figura 4.3: FRF do filtro obtido para $F = F_0$, $N = 1000$ e $SNR = 90$.



(a) último vetor w



(b) valor médio de w

Figura 4.4: FRF do filtro obtido para $F = F_0$, $N = 1000$ e $SNR = 10$.

4.1.3 Predição

Para analisarmos a predição do filtro, foi escolhida uma força arbitrária F_3 conforme eq. (4.1)

$$F_3 = B_1 sin(\omega_1 t) + B_2 sin(\omega_2 t) + B_3 sin(\omega_3 t) + \nu \tag{4.1}$$

onde $B_1 = 1\,\mathrm{N}$, $B_2 = 2\,\mathrm{N}$, $B_3 = 3\,\mathrm{N}$, $\omega_1 = 14\,\mathrm{rad\,s^{-1}}$, $\omega_2 = 40\,\mathrm{rad\,s^{-1}}$, $\omega_3 = 65\,\mathrm{rad\,s^{-1}}$ e ν é um ruído branco de variância 1.

A fig. 4.5 mostra a predição para o filtro obtido com $F = F_0$, $N = 1000$ e $SNR = 90$. Podemos observar que a predição obtida com o filtro não é boa e o erro $(e(n) = y^{exp}(n) - y(n))$ é da ordem do valor previsto pelo filtro.

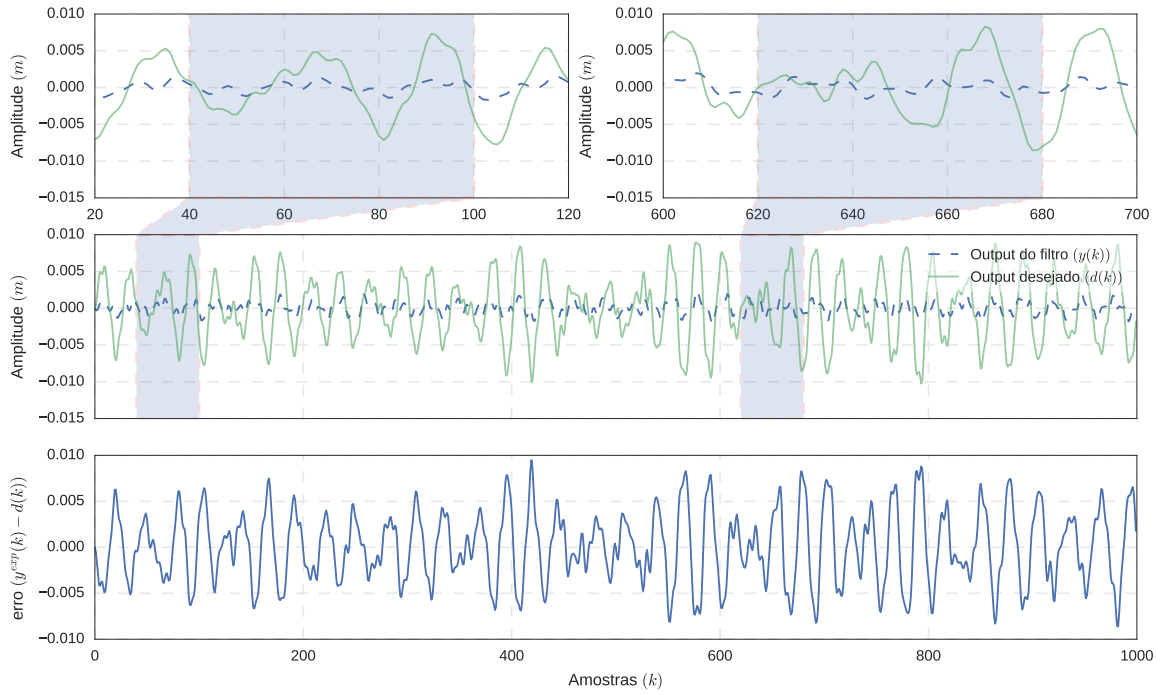


Figura 4.5: Predição do filtro obtido com $F = F_0$, $N = 1000$ e $SNR = 90$.

Os resultados obtidos para os diferentes valores de N e SNR não apresentam diferença para o caso em que $F = F_0$ e por isso serão omitidos.

4.2 Resultados para F_1

4.2.1 Adaptação do filtro

Os resultados da adaptação do filtro para uma força $F_1(t) = A_1 sin(2\pi f_1 t) + A_2 sin(2\pi f_2 t)$ com $N = 1000$ e $SNR = 90$ são mostrados na fig. 4.6. Para este caso a convergência se mostrou próxima ao que foi obtido na análise de F_0.

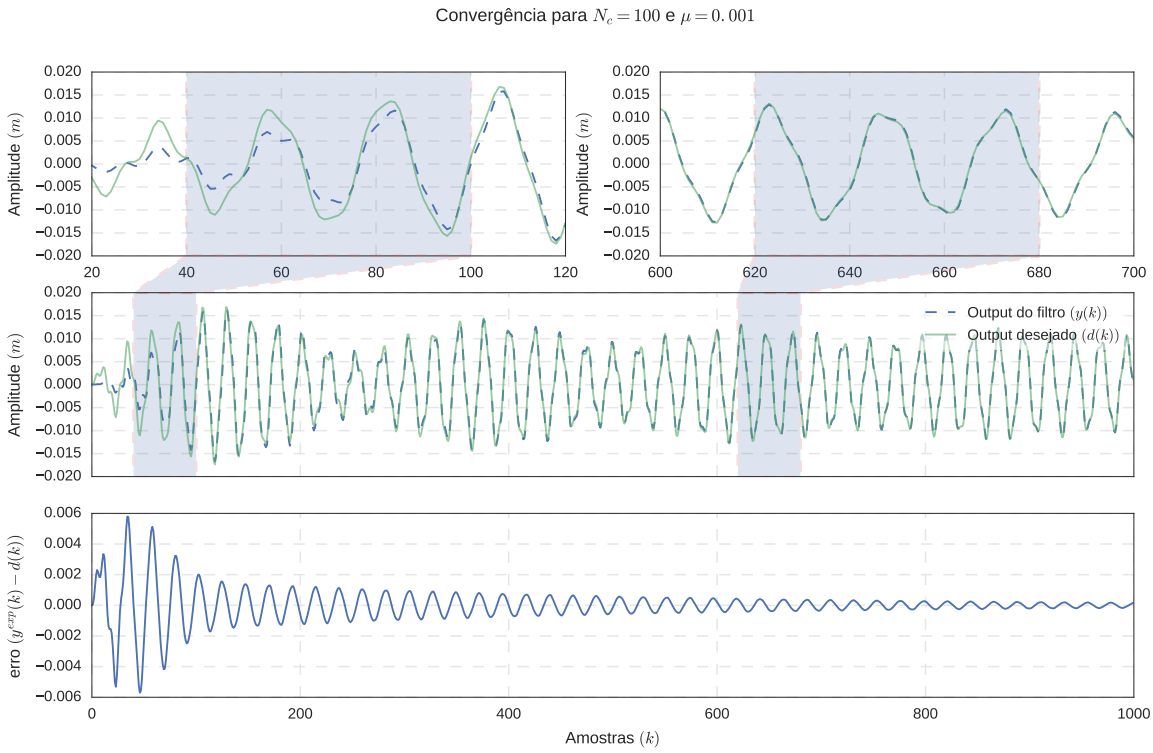


Figura 4.6: Evolução do filtro para $F = F_1$, $N = 1000$ e $SNR = 90$.

Na fig. 4.7 são apresentados os resultados para $N = 1000$ e $SNR = 10$. Para este caso a convergência não é boa e ao fim das iterações o erro do filtro ainda é alto.

Podemos tentar melhorar a adaptação aumentando o valor do fator de convergência de $\mu = 0.001$ para $\mu = 0.003$ o que por sua vez deve aumentar a velocidade de convergência do algoritmo. A fig. 4.8 mostra que, quando utilizamos $\mu = 0.003$, o valor de saída do filtro tende a variar mais rapidamente tentando acompanhar o valor desejado, no entanto, as variações bruscas nos valores dos coeficientes do filtro não permitem a convergência e consequente minimização do erro após decorridas várias iterações.

Conforme DINIZ [1], o valor de μ deve ser escolhido no intervalo mostrado na eq. (4.2)

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{4.2}$$

onde λ_{max} corresponde ao maior autovalor da matriz $\mathbf{R}$ definida na eq. (3.4). Podemos concluir que o valor $\mu = 0.003$ se encontra fora desse intervalo, já que a convergência não foi atingida e os resultados se apresentaram piores do que os obtidos com $\mu = 0.001$.

Para que possamos obter melhores resultados iremos utiliza um fator maior que $\mu = 0.001$ com o objetivo de aumentar a velocidade de adaptação do filtro e menor

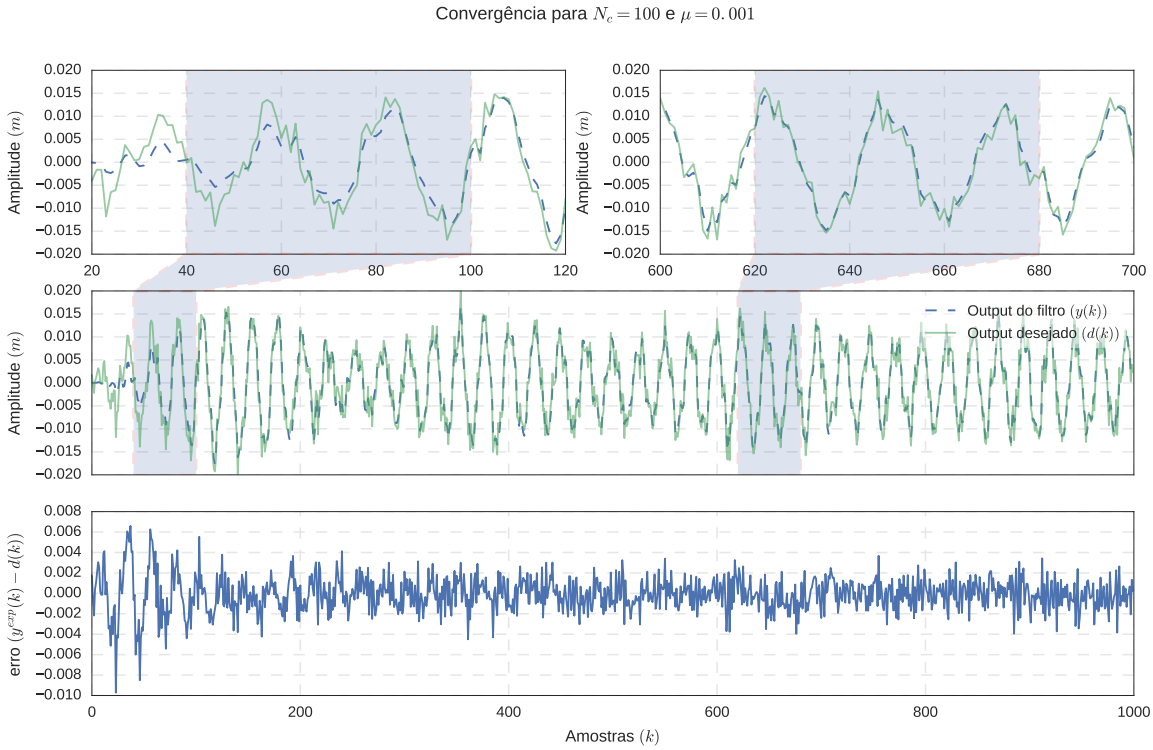


Figura 4.7: Evolução do filtro para $F = F_1$, $N = 1000$ e $SNR = 10$.

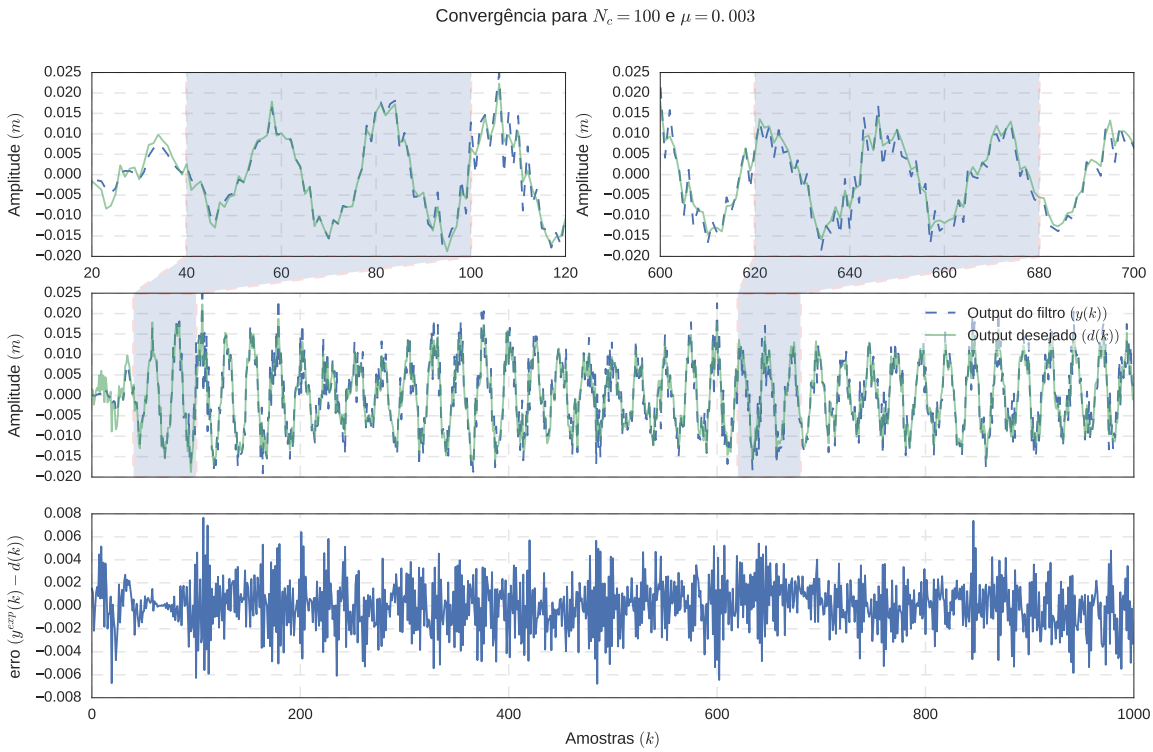


Figura 4.8: Evolução do filtro para $F = F_1$, $N = 1000$ e $SNR = 10$.

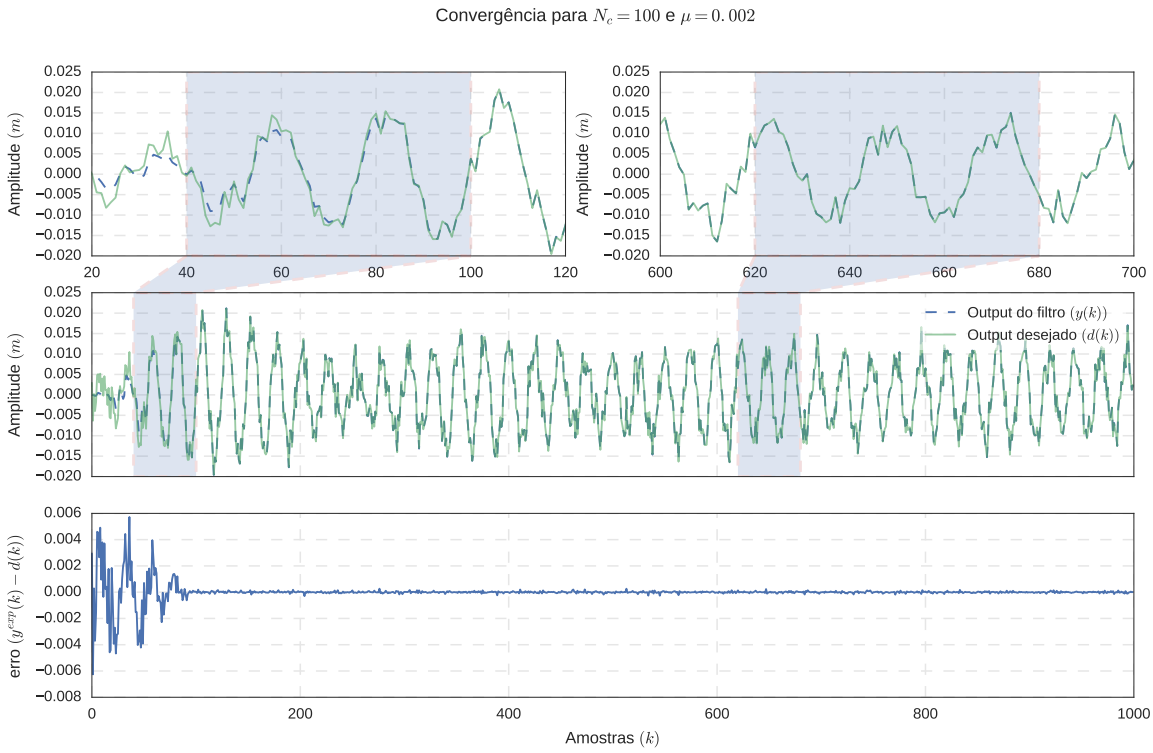


Figura 4.9: Evolução do filtro para $F = F_1$, $N = 1000$ e $SNR = 10$.

que $\mu = 0.003$ para que a convergência possa ser obtida. A fig. 4.9 mostra que para $\mu = 0.002$ o filtro atinge a convergência rapidamente, apresentando valores de erro próximos de 0 após 100 iterações.

4.2.2 FRF do filtro

Assim como visto em 4.1, os resultados para as FRFs se mostraram muito parecidos utilizando-se o último vetor de coeficientes ou o valor esperado dos coeficientes tomando por base as observações da segunda metade do vetor de dados. Sendo assim, iremos apresentar a FRF apenas para o último vetor de coeficientes do filtro.

A fig. 4.10 mostra os resultados obtidos com $N = 1000$ e $SNR = 90$. Podemos notar uma melhora dos resultados quando comparamos com a FRF obtida para $F = F_0$, mas os valores ainda estão distantes da FRF real do sistema.

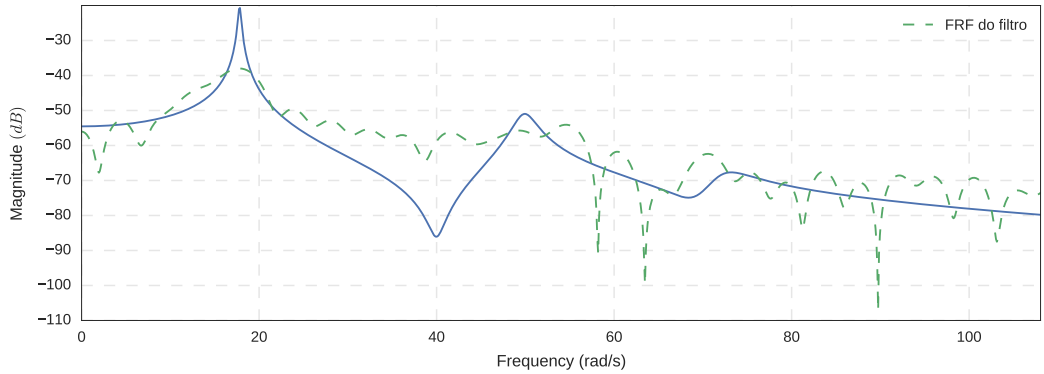


Figura 4.10: FRF do filtro obtido para $F = F_1$, $N = 1000$ e $SNR = 10$.

4.2.3 Predição

Para a predição iremos utilizar a mesma força utilizada em $F = F_0$ e descrita na 4.1. Os resultados da predição são apresentados na fig. 4.11 e mostram uma melhora, mas podemos ver que este filtro também não foi capaz de prever de maneira precisa o comportamento do sistema para uma força diferente da utilizada no processo de adaptação.

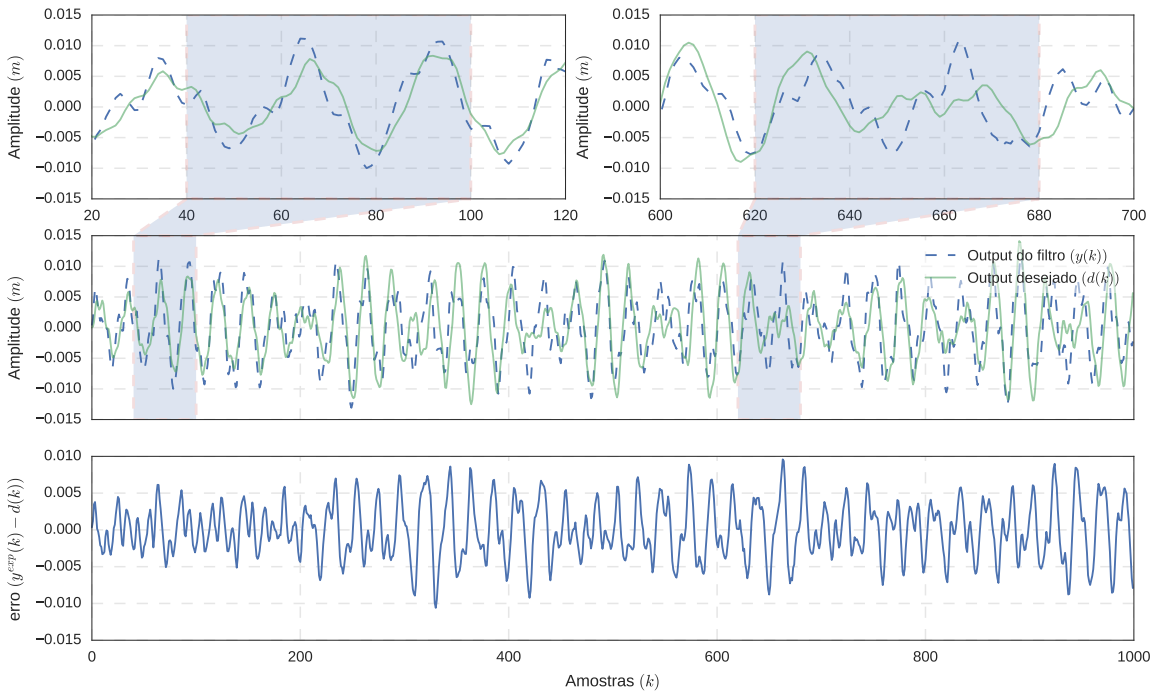


Figura 4.11: Predição do filtro obtido com $F = F_1$, $N = 1000$ e $SNR = 10$.

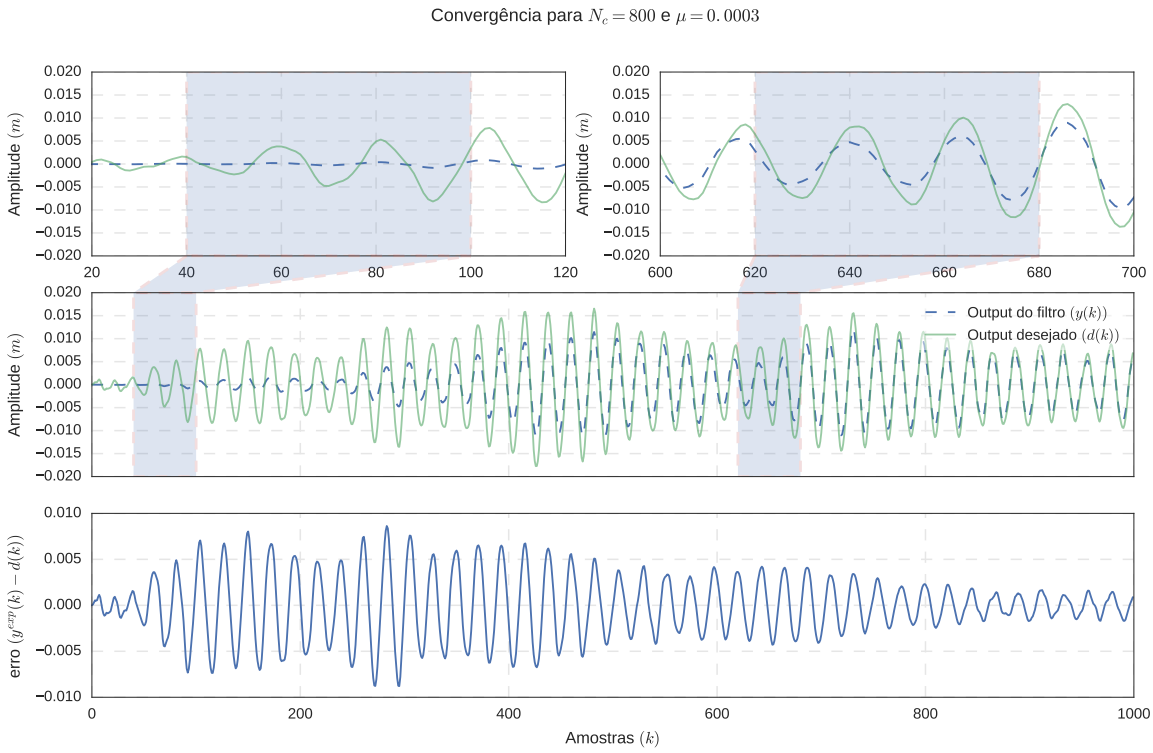


Figura 4.12: Evolução do filtro para $F = F_2$, $N = 1000$ e $SNR = 90$.

4.3 Resultados para F_2

4.3.1 Adaptação do filtro

Os resultados da adaptação do filtro para uma força F_2 (ruído branco) com $N = 1000$ são mostrados na fig. 4.12, fig. 4.13 e fig. 4.14 para $SNR = 90$, 50 e 10 respectivamente. Neste caso, mesmo com o filtro possuindo 8 vezes mais coeficientes ($Nc = 800$), não foi possível obter a convergência.

A convergência só pode ser obtida para uma amostragem com $N = 5000$ e um filtro com 1800 coeficientes. Os resultados da adaptação para $N = 5000$ são mostrados na fig. 4.15, fig. 4.16 e fig. 4.17 para $SNR = 90$, 50 e 10 respectivamente. Podemos observar que, mesmo para um nível de ruído elevado ($SNR = 10$) o filtro apresentou uma boa convergência após aproximadamente 1600 iterações.

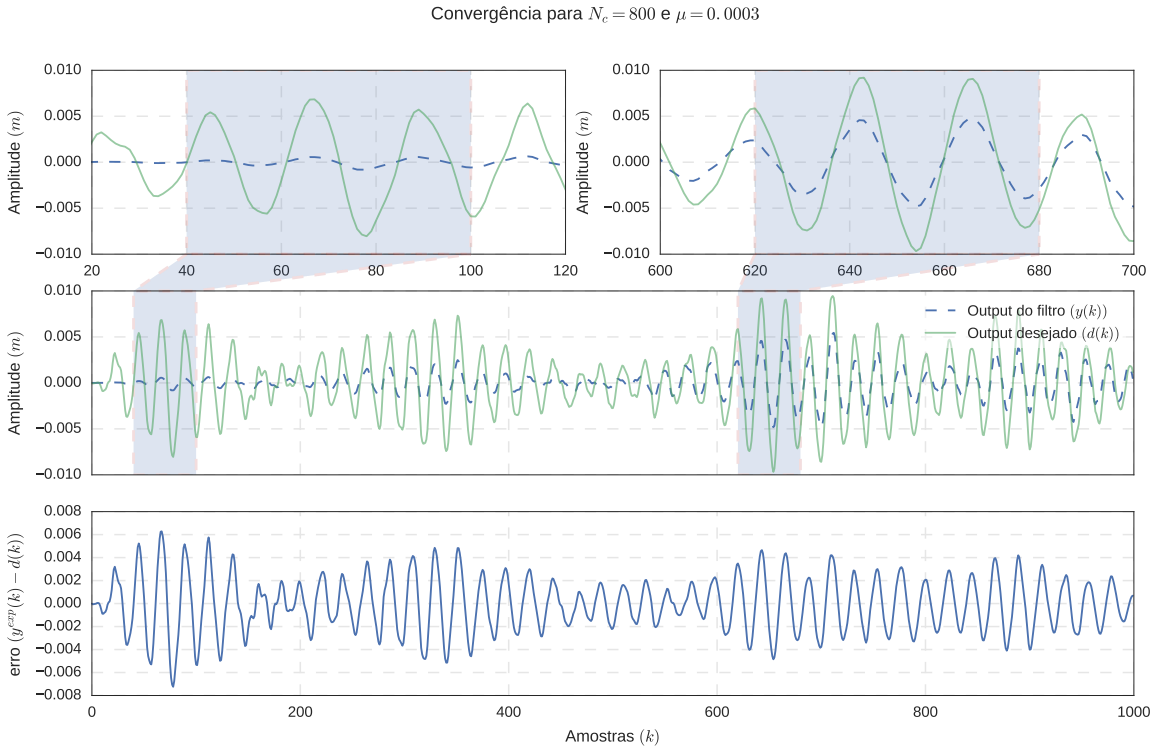


Figura 4.13: Evolução do filtro para $F = F_2$, $N = 1000$ e $SNR = 50$.

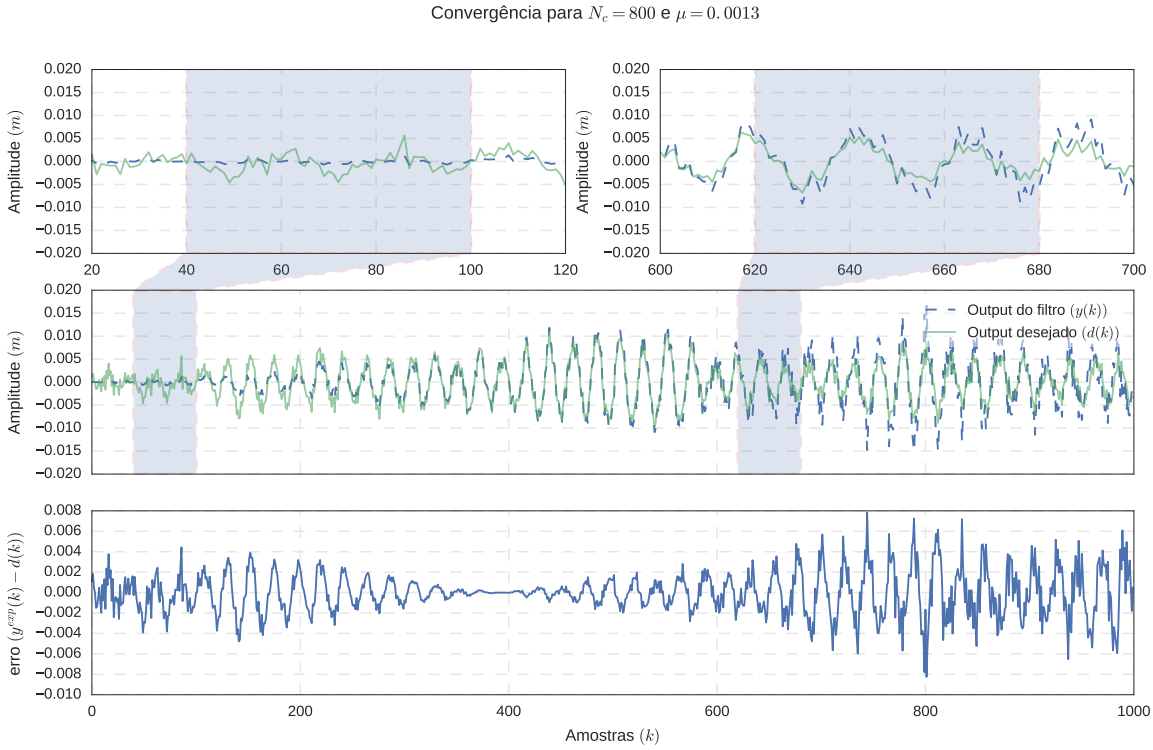


Figura 4.14: Evolução do filtro para $F = F_2$, $N = 1000$ e $SNR = 10$.

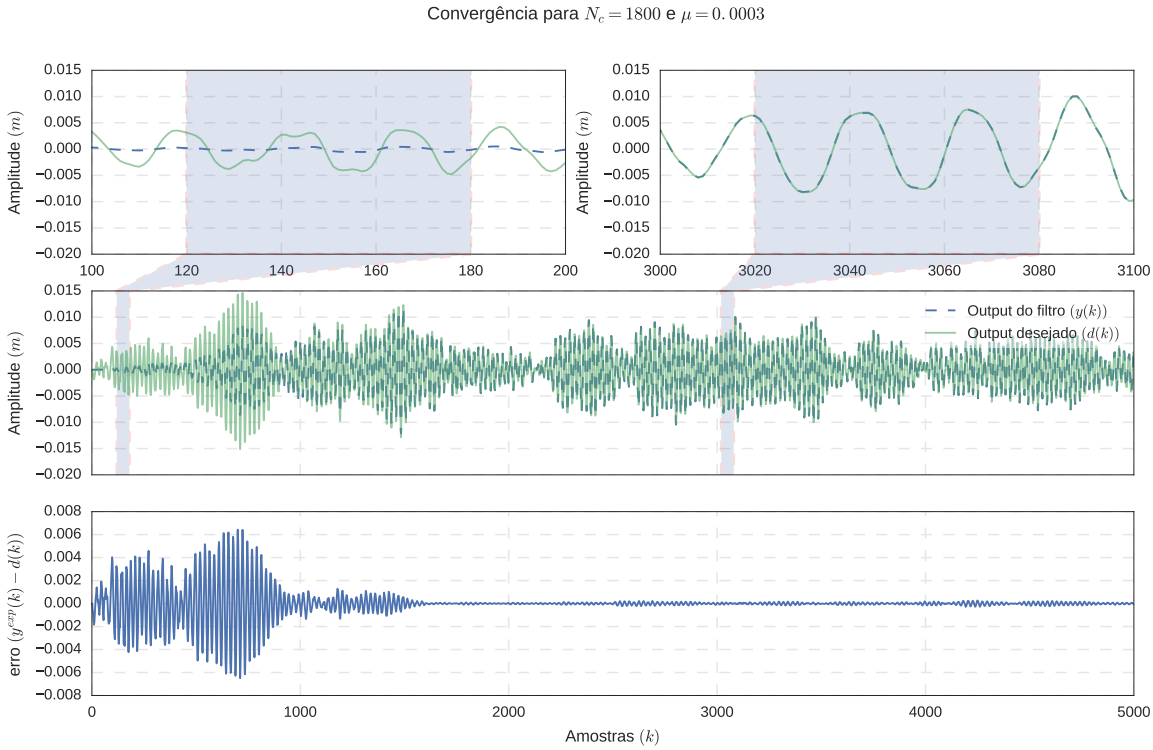


Figura 4.15: Evolução do filtro para $F = F_2$, $N = 5000$ e $SNR = 90$.

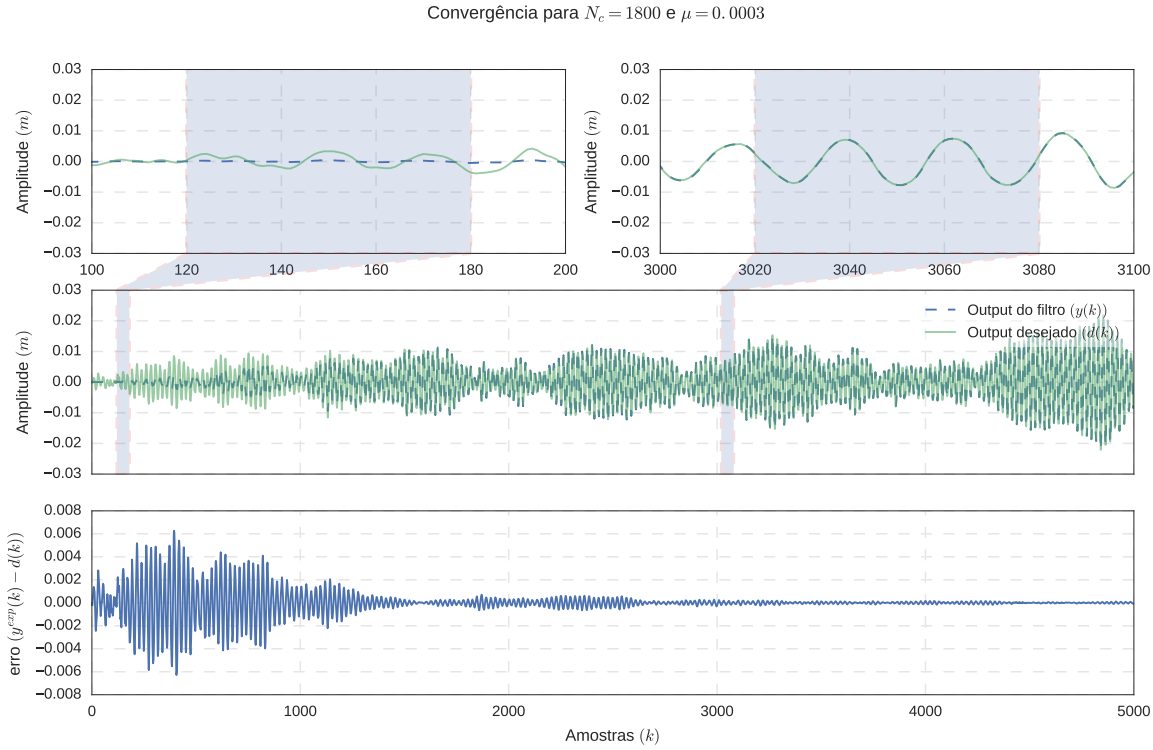


Figura 4.16: Evolução do filtro para $F = F_2$, $N = 5000$ e $SNR = 50$.

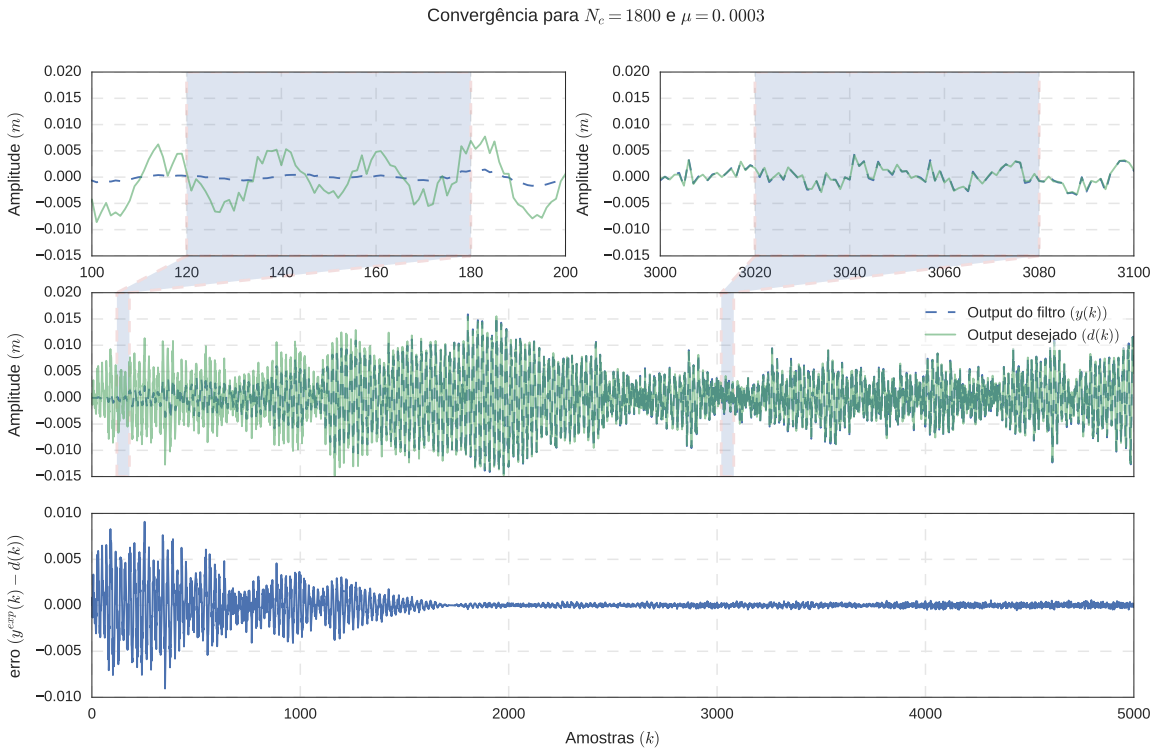


Figura 4.17: Evolução do filtro para $F = F_2$, $N = 5000$ e $SNR = 10$.

4.3.2 FRF do filtro

Para a análise das FRFs iremos utilizar os filtros obtidos com $N = 5000$, já que apenas estes apresentaram convergência nos seus coeficientes. Além disso, assim como na análise de F_1, apenas as FRFs do último vetor de coeficientes do filtro serão apresentadas.

A fig. 4.18 mostra que para $SNR = 90$ a FRF do filtro apresenta excelentes resultados quando comparada à FRF real do sistema. Os três picos referente às frequências naturas foram capturados, além disso a queda na amplitude devido o fenômeno de anti-ressonância também é claramente observado. Outro ponto im-

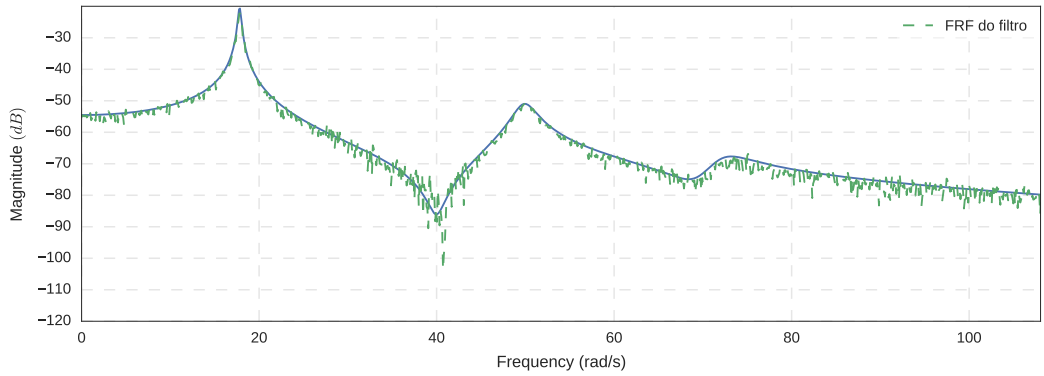


Figura 4.18: FRF do filtro obtido para $F = F_2$, $N = 5000$ e $SNR = 90$.

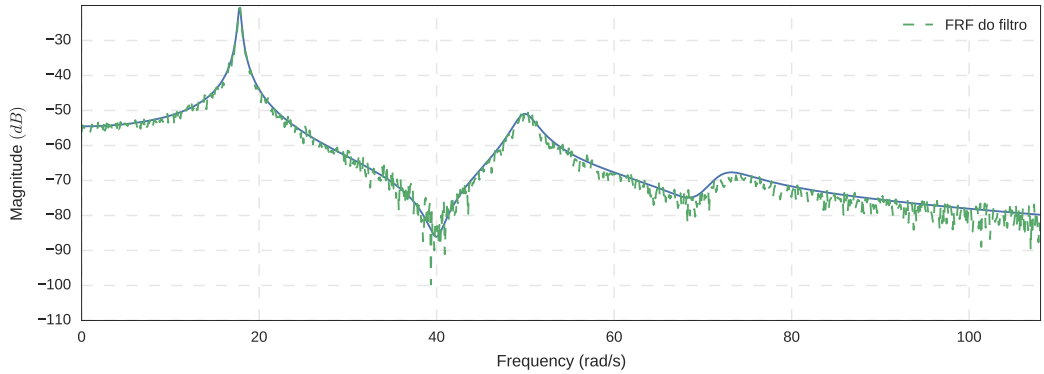


Figura 4.19: FRF do filtro obtido para $F = F_2$, $N = 5000$ e $SNR = 50$.

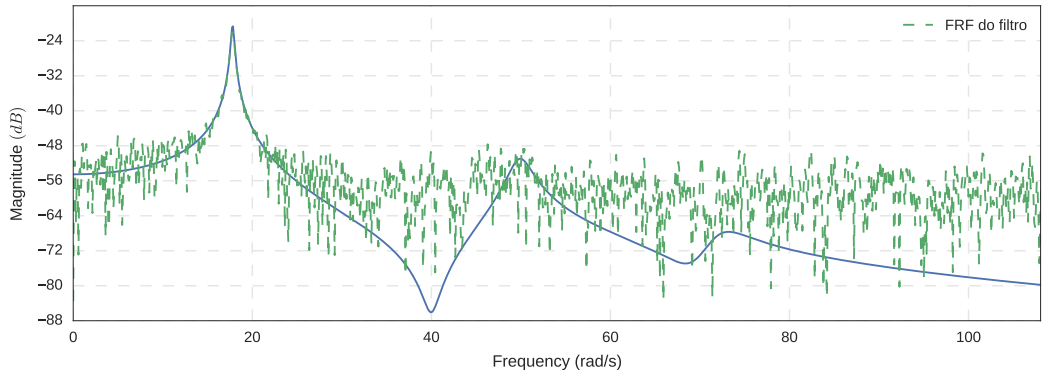


Figura 4.20: FRF do filtro obtido para $F = F_2$, $N = 5000$ e $SNR = 10$.

portante é que, mesmo nas frequências mais elevadas e distantes das frequências naturais, a FRF do filtro reproduziu com exatidão os valores de amplitude esperados.

A fig. 4.19 mostra que, para $SNR = 50$, os resultados ainda são satisfatórios. No entanto, ao aumentarmos o nível de ruído no sinal, podemos notar que para frequências mais elevadas a FRF do filtro apresenta uma maior variação, diferindo um pouco dos valores esperados quando comparamos com a FRF do sistema.

A fig. 4.20 apresenta a FRF para nível de ruído com $SNR = 10$. Neste caso ainda foi possível obter um bom resultado para a primeira frequência natural, mas se afastando desse pico vemos que o filtro não descreve bem o sistema. O fenômeno de anti-ressonância não é observado e mesmo o pico referente à segunda frequência natural não pode ser visto claramente.

4.3.3 Predição

Para a predição iremos utilizar a mesma força utilizada em $F = F_0$ e descrita na 4.1. Os resultados da predição são apresentados na fig. 4.21 e na fig. 4.22, para $SNR = 90$ e 10 respectivamente. Os resultados mostram que, para $SNR = 10$, o

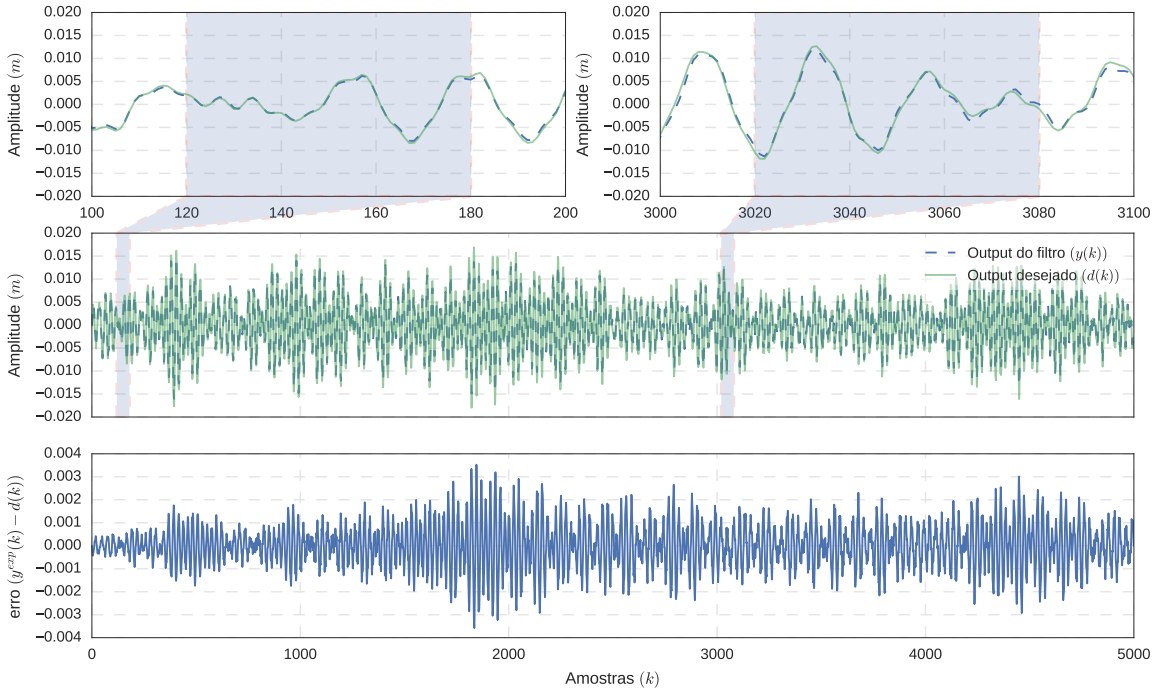


Figura 4.21: Predição do filtro obtido com $F = F_2$, $N = 5000$ e $SNR = 90$.

filtro obtido é capaz de prever com grande exatidão a resposta do sistema.

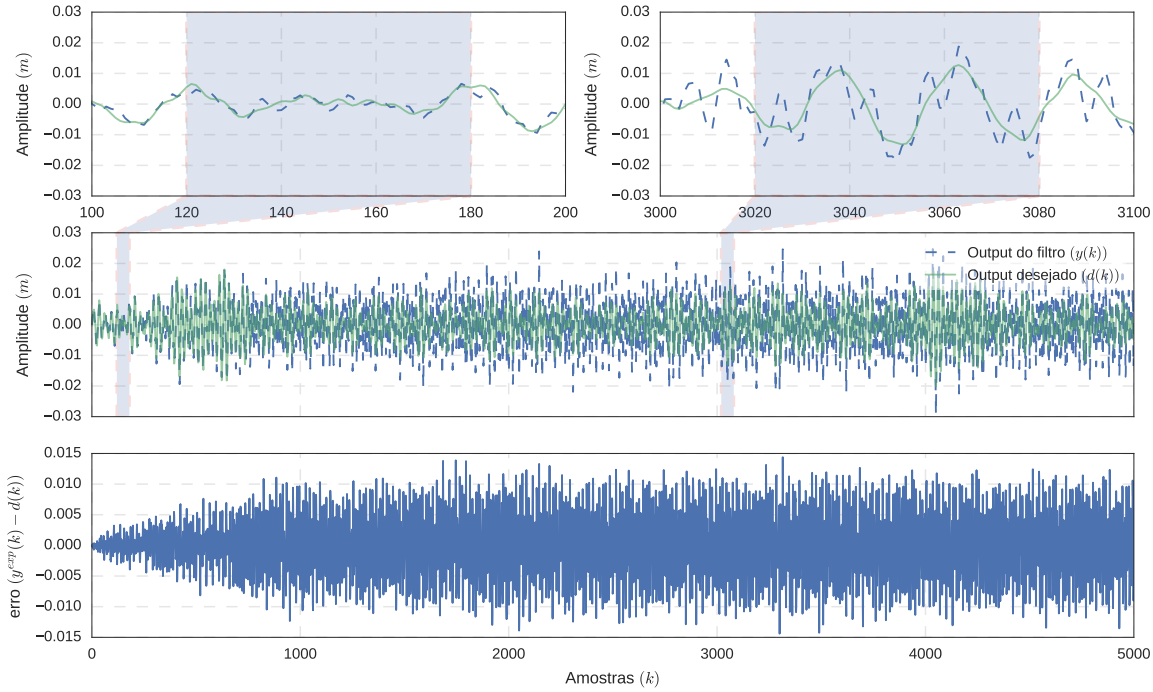


Figura 4.22: Predição do filtro obtido com $F = F_2$, $N = 5000$ e $SNR = 10$.

Capítulo 5

Conclusões

Através da construção de um filtro adaptativo para um sistema com três graus de liberdade foi possível avaliar a influência de diversos fatores no projeto do filtro.

Um filtro projetado com um sinal de entrada que possua uma frequência de excitação específica não apresenta boas predições para forçamentos diferentes daquele utilizado no processo adaptativo. Sendo assim, caso se deseje obter melhores predições para um determinado intervalo de frequência, recomenda-se que a frequência da força de entrada seja variada durante o processo de adaptação do filtro.

O ruído branco se mostrou como o melhor sinal de entrada para a identificação do sistema em um determinado intervalo de frequência. O filtro construído com esse sinal de entrada e com SNR alto foi capaz de detectar os três picos referentes às frequências naturais do sistema e também o fenômeno de anti-ressonância. Além disso o filtro foi capaz de prever com grande exatidão o comportamento do sistema para uma força diferente daquela utilizada no processo de adaptação. Para um SNR mais baixo o filtro ainda foi capaz de identificar a primeira frequência natural do sistema, mas apresentou resultados ruins para frequências afastadas do primeiro pico e também para a predição a uma força diferente da utilizada no processo de adaptação.

Com relação ao número de amostras utilizados, foi possível perceber que para o caso de uma excitação em frequência específica o filtro apresenta uma convergência rápida, sendo necessárias menos de 200 iterações para que o erro se aproximasse de zero. Já para o caso do ruído branco, foi necessária a utilização de um número maior de amostras e também de coeficientes para que a convergência fosse atingida.

Com relação ao fator de convergência, na seção 4.2.1 foi possível observar com a sua escolha tem influência no processo de adaptação do filtro. No caso analisado foi possível perceber que um valor baixo diminui a velocidade de adaptação do filtro, mas um valor muito alto pode resultar na não convergência do algoritmo.

Referências Bibliográficas

[1] DINIZ, P. S. *Adaptive filtering.* Springer, 1997.

[2] CASTELLO, D. A., ROCHINHA, F. A. "An experimental assessment of transverse adaptive fir filters as applied to vibrating structures identification", *Shock and Vibration*, v. 12, n. 3, pp. 197–216, 2005.

Apêndice A

Código utilizado

Foi omitido o código utilizado para gerar o sistema utilizado (código do trabalho 1) e para plotar os gráficos apresentados no trabalho.

```
def F0(t):
        A0 = 1
        w0 = 33.84
        return A0*np.sin(w0*t)

def F1(t):
        A1, A2 = (1, 2)
        w1, w2 = (0.9*18, 1.1*50)
        return A1*np.sin(w1*t) + A2*np.sin(w2*t)

def F2(t):
        var = 1
        norm = stats.norm(0, np.sqrt(var))
        return norm.rvs(len(t))

def F4(t):
        A1, A2, A3 = (1, 2, 3)
        w1, w2, w3 = (0.7*20, 0.8*50, 1.3*50)
        var = 1
        norm = stats.norm(0, np.sqrt(var))
        return A1*np.sin(w1*t) + A2*np.sin(w2*t) + A3*np.sin(w3*t) + norm.rvs(len(t

m0, m1, m2 = (1, 1, 1)
k0, k1, k2 = (1600, 1600, 1600)
M1 = np.array([[m0, 0, 0],
```

```
                [0, m1, 0],
                [0, 0, m2]])
K1 = np.array([[k0+k1, -k1,   0],
                [-k1, k1+k2, -k2],
                [0,     -k2,  k2]])
alpha, beta = 1e-3, 1e-3
C1 = alpha*M1 + beta*K1
sys = vt.VibSystem(M1, alpha*M1 + beta*K1, K1,
r'Sistema 1 - $\alpha = 10^{-3}$ e $\beta = 10^{-3}$')

def noise(sig, snr):
        """
        Returns a corrupted signal based on a
        signal-to-noise ratio (SNR).
        """
        a_s = np.sqrt((sig * sig).mean())
        a_n = a_s/10**(snr/20)
        var = a_n**2
        norm = stats.norm(0, np.sqrt(var))

        return sig + norm.rvs(len(sig))


class LMSFilter(object):
        def __init__(self, Nc, mu):
                """
                Iniciar filtro com Nc coeficientes.
                """
                self.Nc = Nc
                self.mu = mu
                # valores iniciais para o filtro w = [0, 0, ..., 0]
                self.w = np.zeros(Nc)

        def predict(self, x):
                y = self.w @ x
                return y

        def update(self, d, x):
                """
```

```
            Atualizar filtro baseado no sinal de entrada x
            e no valor desejado d.
            """
            y = self.w @ x
            e = d - y
            self.w += 2 * self.mu * e * x

class SysId(object):
    def __init__(self, name, sys, Nc, mu, F, N, snr,
    inp=2, out=0):
    self.name = name
    self.names = name.split('_')
    self.sys = sys
    self.Nc = Nc
    self.mu = mu
    self.F = F
    self.N = N
    self.snr = snr
    self.inp = inp
    self.out = out

    # criar filtro
    self.filt = vt.LMSFilter(Nc, mu)

    # criar array de tempo
    self.dt = 1/(8*(50/(2*np.pi)))
    self.t = np.linspace(0, N*self.dt, N)

    # criar forçamento (input)
    F_ = np.zeros((len(self.t), sys.n))
    F_[:, inp] = F(self.t)
    self.F_ = F_
    self.x = F_[:, inp]

    # resposta do sistema (d)
    _, _, self.sys_time_resp = sys.time_response(self.F_, self.t)
    self.d = self.sys_time_resp[:, out] # selecionar output

    # resposta do sistema com ruído (d_noise)
```

```
d_n = np.copy(self.sys_time_resp.T)
for i, sig in enumerate(d_n):
d_n[i, :] = noise(sig, snr)
self.sys_time_resp_noise = d_n.T
self.d_noise = self.sys_time_resp_noise[:, out] # sel. output

# atualizar filtro
ws_hist = np.zeros((N, Nc))
ys_hist = np.zeros(N)
# adicionar Nc zeros ao início do input
self.x_shift = np.concatenate([np.zeros(Nc - 1), self.x])
for i in range(N):
        self.filt.update(self.d_noise[i], self.x_shift[i: i + Nc])
        ys_hist[i] = self.filt.predict(self.x_shift[i: i + Nc])
        ws_hist[i] = self.filt.w

self.w = self.filt.w
self.ws = ws_hist
self.ys = ys_hist
self.e = ((self.d_noise - self.ys))

def y_last_w(self, sig):
        N = self.N
        Nc = self.Nc
        sig = np.concatenate([np.zeros(Nc - 1), sig])
        y_last_w = np.zeros(N)
        for i in range(N):
        y_last_w[i] += self.filt.predict(sig[i: i + Nc])

return y_last_w

def freq_resp(self, sig, worN):
        fs = 1/self.dt
        w, h = signal.freqz(sig, worN=worN)
        w *= fs

return w, h
```