UNIVERSIDADE FEDERAL DO MARANHÃO

ENGENHARIA DA COMPUTAÇÃO

PROCESSAMENTO DIGITAL DE SINAIS

FILTRO DE KALMAN

Relatório do projeto, referente a terceira avaliação da disciplina EECP0018 - Processamento Digital De Sinais (2019 .1 - T01), sob a Supervisão do **Prof. Me. Arnaldo Pinheiro De Azevedo Junior**

SÃO LUÍS/MA

2019

LUIS FELIPE FERREIRA SOARES

FILTRO DE KALMAN

Relatório do projeto, referente a terceira avaliação da disciplina EECP0018 - Processamento Digital De Sinais (2019 .1 - T01), sob a Supervisão do **Prof. Me. Arnaldo Pinheiro De Azevedo Junior**

SÃO LUÍS/MA

2019

RESUMO

A proposta do projeto foi o projeto e implementação do filtro de Kalman. Este se trata de um estimador ótimo, isto é, uma ferramenta que infere parâmetros desejados a partir de observações indiretas, incertas e/ou imprecisas de um sistema. É classificado com um filtro, pois determina a "melhor estimativa" do ruído de um sinal com o intuito de filtra-lo. Apesar disso, é frequentemente dito como a melhor solução para problemas de rastreamento e predição de trajetórias, pois não só exerce a função de filtro, mas também projeta as medidas na estimativa do estado do sistema. Alguns exemplos notáveis da aplicação do filtro de Kalman são: Determinação da órbita de planetas e estrelas, rastreamento (aeronaves, radares, mísseis) e robores capazes de se auto localizarem. Este trabalho se propõe a apresentar e implementar a sua versão linear, aplicando-a como filtro em medições reais e artificiais da posição e velocidade de um projétil em trajetória vertical.

SUMÁRIO

1 INTRODUÇÃO

Os filtros de Kalman são utilizados quando se tem incerteza na informação sobre a dinâmica de um sistema, porém o sistema é conhecido a ponto de que é possível prever o que ocorrerá a seguir. Mesmo que fatores externos interfiram no movimento previsto, o filtro de Kalman frequentemente será capaz de estimar o que de fato aconteceu. Assim, o filtro de Kalman é ideal para sistemas que estão constantemente mudando e quando há restrições de memória e velocidade, pois o algoritmo apenas depende do estado anterior e é muito rápido, sendo assim adequado em sistemas de tempo real e embarcados [1]. Os seus bons resultados na prática devido a sua performance e estrutura, a sua forma conveniente para processamento em tempo real, a simplicidade na formulação e implementação dado o conhecimento básico do seu funcionamento e a dispensa da necessidade de inverter as equações de medição tornam o filtro de Kalman uma escolha interessante em diversas aplicações [2].

O filtro de Kalman é dito um estimador ótimo [2, 3]. Ele é ótimo no sentido de que se o ruído é do tipo gaussiano, então o filtro minimiza o erro quadrático médio. Para observar este fato, considere que o sinal possa ser descrito da seguinte forma:

$$y_k = a_k x_k + n_k \tag{1.1}$$

onde y_k é uma medição no tempo, a_k um termo de ganho, x_k é o sinal real e n_k, o ruído. O objetivo é estimar x_k. A diferença entre o sinal medido e o real é dado por uma função de erro *f*. A forma de *f* depende da aplicação, mas se espera que esta seja positiva e cresça monotonicamente. Assim, uma função que satisfaz essas condições é

$$f(e_k) = (x_k - \hat{x}_k)^2 \, . \tag{1.2}$$

Em termos estatísticos, uma métrica mais útil é o valor esperado de (1.2):

$$E = E[f(e_k)] = E[e_k^2] \, , \tag{1.3}$$

que é o erro quadrático médio. Isto é equivalente a dizer que se busca $\hat{x}_k$ tal que maximiza a probabilidade de que y_k aconteça. Conforme suposição que o ruído n_k é do tipo gaussiano, com variância σ_k^2, tem-se que

$$P(y_k \mid \hat{x}_k) = K_k Exp\left[-\frac{1}{2} \frac{(y_k - a_k \hat{x}_k)^2}{\sigma_k^2} \right], \tag{1.4}$$

onde K_k é uma constante de normalização. A probabilidade máxima da função (1.4) em todo o domínio k é

$$P(y \mid \hat{x}) = \prod_k K_k exp\left[-\frac{1}{2} \frac{(y_k - a_k \hat{x}_k)^2}{\sigma_k^2} \right] \tag{1.5}$$

o que resulta em

$$\ln[P(y\,|\,\hat{x})] = -\frac{1}{2}\sum_k \frac{(y_k - a_k\hat{x}_k)^2}{\sigma_k^2} + C_K\,, \tag{1.6}$$

ou seja, o erro quadrático médio é a função a ser minimizada em $\hat{x}_k$ quando a variação n_k é gaussiana. O filtro ótimo que tem esta característica, dentre todos aqueles que minimizam o erro quadrático médio, é o filtro de Kalman [3]. Quando o ruído não é gaussiano, dada apenas a variância do ruído, o filtro de Kalman é o melhor estimador linear, porém estimadores não-lineares podem ser melhores [2].

2 DERIVAÇÃO DO FILTRO DE KALMAN

Considere que se deseja obter o valor de uma variável de um processo na forma:

$$x_{k+1} = Ax_k + w_k , \tag{2.1}$$

onde x_k é o vetor de estados, A é a matriz de transição e w_k é um ruído branco com variância conhecida. As observações são dadas na forma

$$z_k = Hx_k + v_k , \tag{2.2}$$

onde z_k é o vetor de medidas atuais de x no tempo k, H é uma matriz de relação entre todas as variáveis de estado consideradas e as variáveis de fato medidas e v_k é o vetor relacionado ao erro de medição. Como se deseja trabalhar com as distribuições gaussianas, é necessário conhecer as matrizes de covariância de w_k e v_k, que serão definidas, respectivamente, como:

$$R = E[v_k v_k^T] \tag{2.3}$$

e

$$Q = E[w_k w_k^T] . \tag{2.4}$$

O erro quadrático médio é dado por (1.3), isto é equivalente a

$$P_k = E[e_k e_k^T] = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] . \tag{2.5}$$

Assumindo uma estimativa $\hat{x}_k'$ a priori de $\hat{x}_k$, obtida através do conhecimento sobre o sistema, é possível determinar a equação para a estimativa de $\hat{x}_k$, combinando a estimativa a priori com os dados medidos, que é dada por:

$$\hat{x}_k = \hat{x}_k^{'} + K_k(z_k - H\hat{x}_k^{'}) = \hat{x}_k^{'} + K_k i_k , \tag{2.6}$$

onde K_k é o ganho de Kalman e i_k é denominado "termo de inovação". Combinando (2.6) e (2.2) com (2.5), tem-se

$$P_k = E[[(I - K_k H)(\hat{x}_k - \hat{x}_k^{'}) + K_k v_k][(I - K_k H)(\hat{x}_k - \hat{x}_k^{'}) + K_k v_k]^T] . \tag{2.7}$$

O termo $\hat{x}_k - \hat{x}_k'$ é o erro da estimativa a priori. Sabendo que este erro não está relacionado com o ruído v_k, utilizando (2.3) e as propriedades do valor esperado, reescreve-se (2.7) como

$$P_k = (I - K_k H)E[\hat{x}_k - \hat{x}_k^{'}](I - K_k H) + K_k E[v_k v_k^T]K_k^T = (I - K_k H)P_k^{'}(I - K_k H) + K_k R K_k^T . \tag{2.8}$$

A equação em (2.8) atualiza a covariância do erro utilizando a covariância a priori P_k'. A matriz de covariância tem a forma

$$P_k = \begin{bmatrix} E[e_{k-1}e_{k-1}^T] & E[e_k e_{k-1}^T] & E[e_{k+1}e_{k-1}^T] \\ E[e_{k-1}e_k^T] & E[e_k e_k^T] & E[e_{k+1}e_k^T] \\ E[e_{k-1}e_{k+1}^T] & E[e_k e_{k+1}^T] & E[e_{k+1}e_{k+1}^T] \end{bmatrix} . \tag{2.9}$$

A diagonal da matriz (2.9) contém os erros quadráticos médios de e_{k-1}, e_k e e_{k+1}. Assim, os erros quadráticos médios podem ser minimizados minimizando o traço da matriz P_k. O traço T de (2.8) é dado por

$$T[P_k] = T[P_k^{'}] - 2T[K_k H P_k^{'}] + T[K_k (H P_k^{'} H^T + R) K_k^T]. \tag{2.10}$$

Diferenciando (2.10) em relação a K_k, tem-se

$$\frac{dT[P_k]}{dK_k} = -2(H P_k^{'})^T + 2K_k (H P_k^{'} H^T + R). \tag{2.11}$$

Igualando (2.11) a zero e rearranjando os termos, obtém-se

$$(H P_k^{'})^T = K_k (H P_k^{'} H^T + R). \tag{2.12}$$

Resolvendo (2.12) em K_k, finalmente se deriva a equação para o ganho de Kalman:

$$K_k = (H P_k^{'})^T (H P_k^{'} H^T + R)^{-1} = (H P_k^{'})^T S_k^{-1}. \tag{2.13}$$

Aplicando (2.13) em (2.8), obtém-se

$$P_k = (I - K_k H) P_k^{'}. \tag{2.14}$$

Como as próximas iterações dependem as estimativas a priori $\hat{x}_k'$ e $\hat{P}_k'$, que variam com k, é necessário determina-las no passo atual, ou seja, realizar um projeção para determinar $\hat{x}_{k+1}'$ e $\hat{P}_{k+1}'$ [3]. Para tanto, combinam-se as equações (2.5), (2.6), (2.13) e (2.14), resultando em:

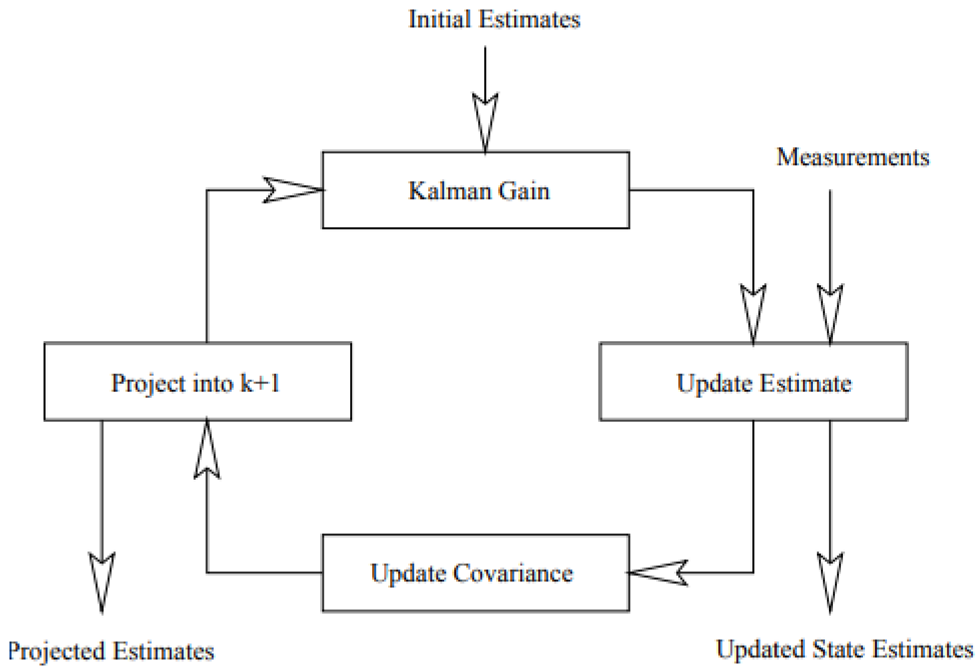
$$\hat{x}_{k+1}^{'} = A\hat{x}_k \tag{2.15}$$

e

$$\hat{P}_{k+1}^{'} = A P_k A^T + Q. \tag{2.16}$$

3 DESCRIÇÃO DA IMPLEMENTAÇÃO DO FILTRO DE KALMAN

Figura 1- Algoritmo do Filtro de Kalman



Fonte: Lacey [3]

As etapas da seção anterior podem ser descritas como um algoritmo recursivo, que funcionam conforme a **Figura 1**. Este consiste em etapas, sendo elas divididas em predição e atualização. Define-se as seguintes notações utilizadas no algoritmo [2]:

- $\hat{x}(k|k)$: Estimativa de x(k) dadas as medidas z(k), z(k-1)...
- $\hat{x}(k+1|k)$: Estimativa de x(k+1) dadas as medidas z(k), z(k-1)...
- $\hat{P}(k|k)$: Estimativa da covariância x(k) dadas as medidas z(k), z(k-1)...
- $\hat{P}(k+1|k)$: Estimativa da covariância x(k+1) dadas as medidas z(k), z(k-1)...

3.1 Estimação do Estado

A estimação do estado é feita da seguinte forma:

1. Predição do estado: $\hat{x}(k+1|k) = A\hat{x}(k|k) + Gu$
2. Predição da medição: $\hat{z}(k+1|k) = H\hat{x}(k+1|k)$
3. Medição de resíduo: $v(k+1) = z(k+1) - \hat{z}(k+1|k)$
4. Atualização da estimativa: $\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)v(k+1)$

3.2 Estimação da Covariância

A estimação do estado é feita da seguinte forma:

1. Predição da covariância: $P(k+1|k) = AP(k|k)A^T + Q$
2. Predição da medição: $S(k+1) = HP(k+1|k)H^T + R$
3. Ganho de Kalman: $K(k+1) = P(k+1|k)H^T S^{-1}(k+1)$
4. Atualização da estimativa: $P(k+1|k+1) = P(k+1|k) - K(k+1)S(k+1)K^T(k+1)$

4 IMPLEMENTAÇÃO DO FILTRO DE KALMAN EM PYTHON

O algoritmo da seção anterior foi implementado em “Python”, conforme mostra a **Figura 2**. Cada passo do algoritmo foi implementado como um método da classe Kalman. O método “Kalman.filter” devolve os valores filtrados $\{x(k+1|k+1), P(k+1|k+1)\}$.

Figura 2 - Implementação em Python do Filtro de Kalman

kalman_filter_class.py - C:\Users\LuisFelipe\AppData\Local\Programs...

File Edit Format Run Options Window Help

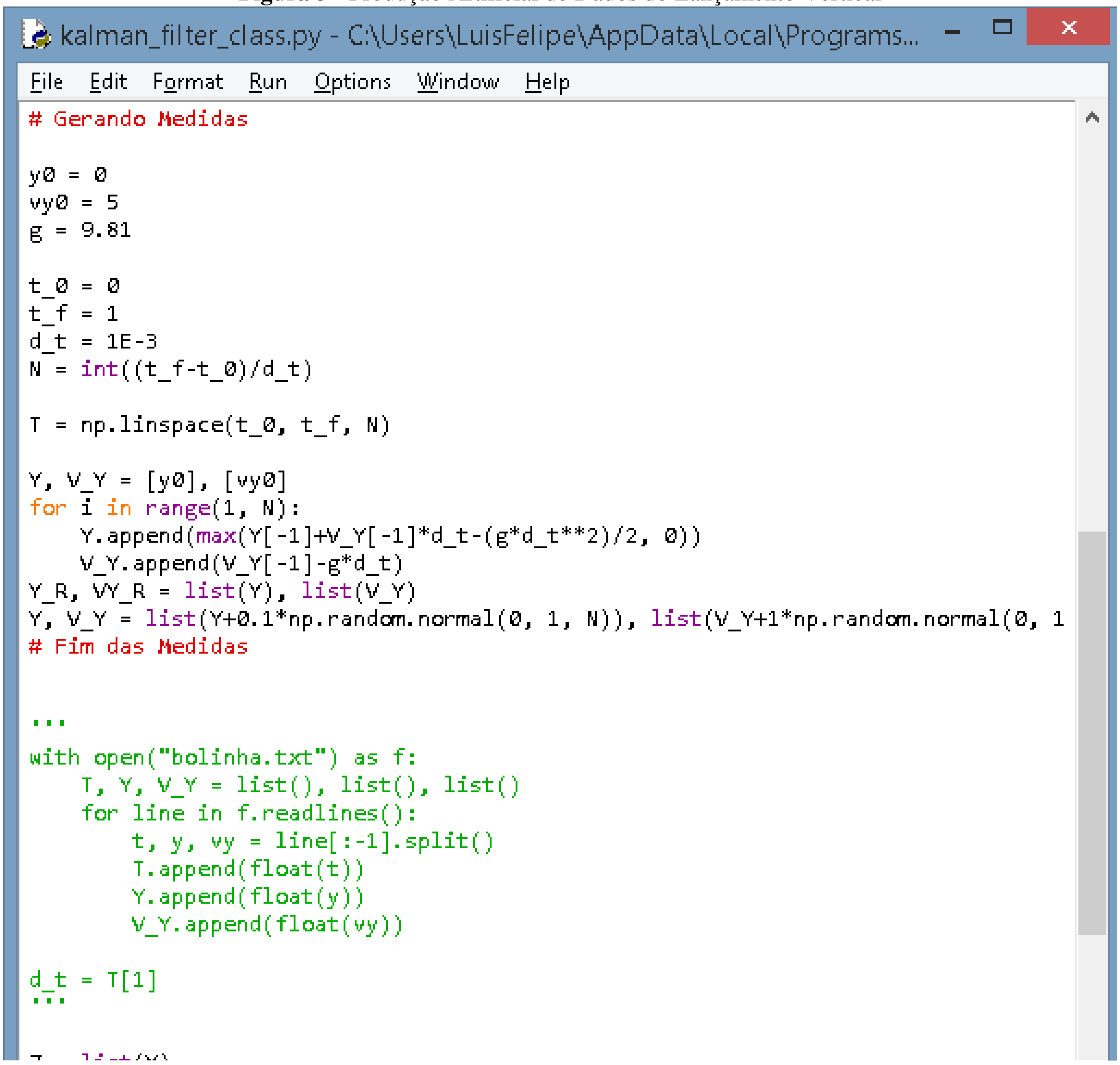
```
class Kalman:
    def __init__ (self, dt):
        self.F = np.matrix([[1, dt], [0, 1]])
        self.G = np.matrix([[0.5*dt**2], [dt]])
        self.u = -9.81
        self.H = np.matrix([1, 0])
        self.Q = np.matrix([[0.05, 0.05], [0.05, 0.05]])
        self.R = 20

    def state_prediction (self, x): # f[x(k|k)] = x(k+1|k)
        return self.F*x+self.G*self.u

    def state_measurement_prediction (self, x): # f[x(k+1|k)] = z(k+1|k)
        return self.H*self.state_prediction(x)

    def state_measurement_residual (self, x, z): #f[z(k+1), z(k+1|k)] = v(k+1)
        return z - self.state_measurement_prediction(x)

    def state_update (self, x, P, z): # f[x(k+1|k), K(k+1), v(k+1)] = x(k+1|k+1)
        K = self.filter_gain(P)
        return self.state_prediction(x)+K*self.state_measurement_residual(x, z)

    def covariance_prediction (self, P): #f[P(k|k)] = P(k+1|k)
        return self.F*P*self.F.T+self.Q

    def covariance_measurement_prediction (self, P): # f[P(k+1|k)] = S(k+1)
        return self.H*self.covariance_prediction(P)*self.H.T+self.R

    def filter_gain (self, P): # f[P(k+1|k), S(k+1)] = K(k+1)
        S = self.covariance_measurement_prediction(P)
        return self.covariance_prediction(P)*self.H.T*S.I

    def covariance_update (self, P): # f[K(k+1)] = P(k+1|k+1)
        K = self.filter_gain(P)
        I = np.eye(self.F.shape[1])
        return ((I-K*self.H)*P)*((I-K*self.H).T)+K*self.R*K.T

    def filter (self, x, P, z):
        return (self.state_update(x, P, z), self.covariance_update(P))
```

Ln: 43 Col: 0

Fonte: Elaborado pelo autor. Baseado no algoritmo de Kleeman [2].

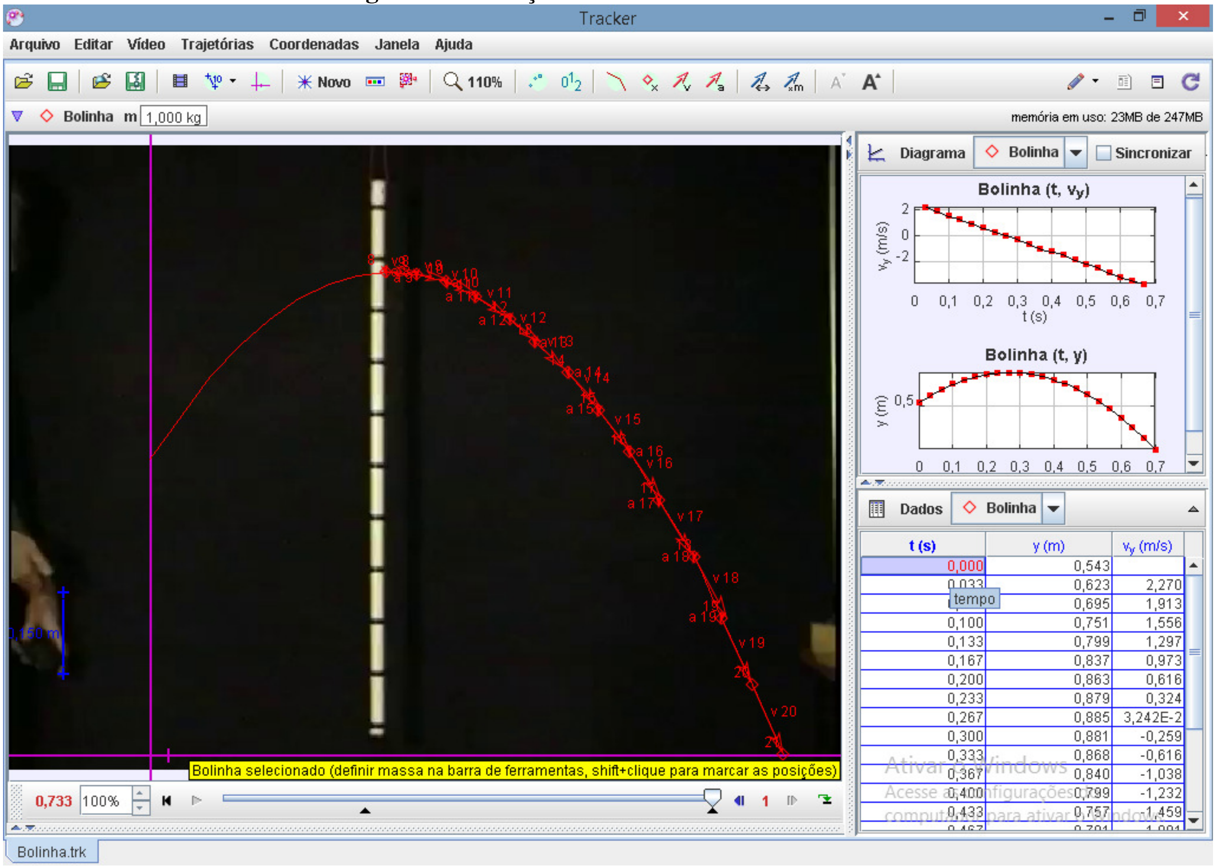
Os dados são gerados artificialmente utilizando um modelo real e adicionando ruído (amostras aleatórias de uma distribuição normal) a este sinal com a biblioteca “numpy”, como pode ser visto em **Figura 3**. Dados experimentais também foram obtidos a partir da ferramenta de análise de vídeos “Tracker”, conforme mostra a **Figura 4**. A aplicação do filtro sobre os dados pode ser vista na **Figura 5**.

Figura 3 - Produção Artificial de Dados do Lançamento Vertical



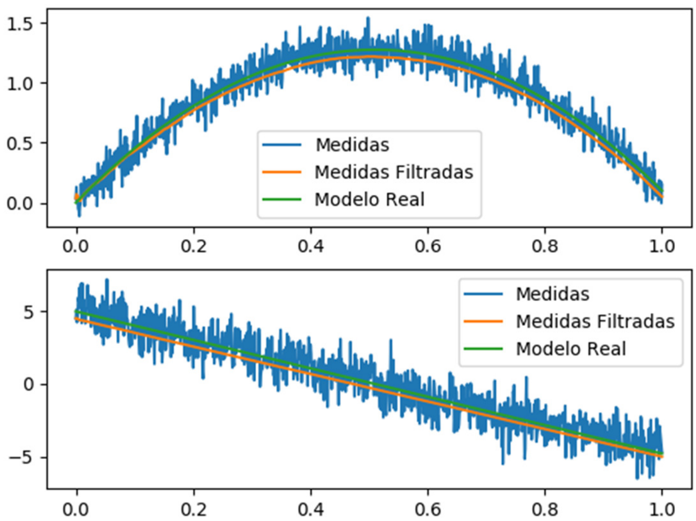
Fonte: Elaborado pelo autor.

Figura 4 - Obtenção de Dados através do Tracker



Fonte: Elaborado pelo autor.

Figura 5 - Comparação Entre Dados Medidos e Filtrados



Fonte: Elaborado pelo autor.

CONCLUSÃO

Inicialmente, introduziu-se o filtro de Kalman descrevendo a sua natureza e as suas aplicações mais recorrentes para ilustrar as possibilidades do seu uso e para justificar a sua apresentação como tema de projeto no contexto da disciplina de Processamento Digital de Sinais. Em seguida, se descreve as ideias por trás do filtro de Kalman, com o intuito de enriquecimento teórico sobre o funcionamento do filtro e também permitir que seja possível compreender os parâmetros presentes nos detalhes da implementação do algoritmo. Esta implementação foi descrita através de fluxograma, em passo-a-passos escritos e também com a implementação na linguagem de programação Python. Os dados provenientes como entrada foram tanto gerados artificialmente, quanto experimentalmente utilizando o software Tracker para extrair dados da trajetória de um lançamento oblíquo. Com isso, conclui-se que o trabalho foi satisfatório, já que além de se conhecer esta poderosa ferramenta que é o filtro de Kalman, também se implementou o algoritmo, levando a um conhecimento mais profundo das ideias e parâmetros envolvidos na construção do filtro, abrindo a possibilidade para implementações futuras do filtro em problemas mais interessantes.

REFERÊNCIAS

[1] BZARG. **How a Kalman filter works, in pictures**. 2015. Disponível em: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/. Acesso em 06 jul. 2019.

[2] Kleeman, L. **Understanding and Applying Kalman Filtering**. Monash University, 2017. Disponível em: https://ecse.monash.edu/centres/irrc/LKPubs/Kalman.pdf. Acesso em: 07 jul. 2019

[3] LACEY, T. **Tutorial: The kalman filter**. Georgia Institute of Technology, 2019. Disponível em: https://pdfs.semanticscholar.org/2a47/61df17525de463341320bf0458c98e04c654.pdf. Acesso em: 06 jul. 2019.

[4] SRINI, S. **The Kalman Filter: An algorithm for making sense of fused sensor insight**. 2018. Disponível em: https://towardsdatascience.com/kalman-filter-an-algorithm-for-making-sense-from-the-insights-of-various-sensors-fused-together-ddf67597f35e. Acesso em: 06 jul. 2019.

[5] TEOW, J. **Understanding Kalman Filters with Python**. 2018. Disponível em: https://medium.com/@jaems33/understanding-kalman-filters-with-python-2310e87b8f48. Acesso em 05 jul. 2019.