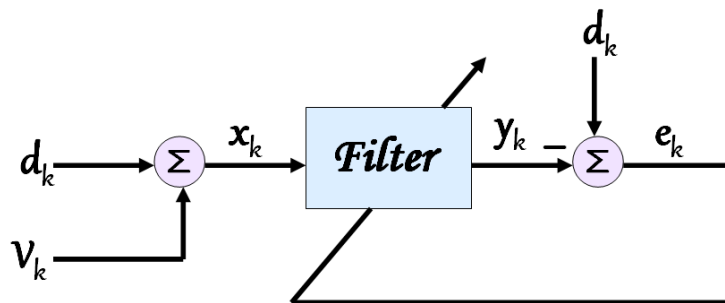


【ウィナーフィルタ Wiener Filter】



d_k : Desired signal v_k : Noise x_k : Observed value
 y_k : Filter output e_k : Estimation error

d_k 所望の信号, y_k 適応フィルター出力, 推定誤差 $e_k = d_k - y_k$ とし,
 フィルター入力 $\mathbf{x}_k = [x_k, x_{k-1}, \dots, x_{k-M+1}]^T$, フィルター係数 $\mathbf{w} = [w_1, w_2, \dots, w_{M-1}]^T$ とする.

評価関数は,

$$\begin{aligned}
 J(\mathbf{w}) &= \mathbb{E}[e_k^2] = \mathbb{E}[(d_k - \mathbf{w}^T \mathbf{x}_k)(d_k - \mathbf{w}^T \mathbf{x}_k)^T] = \mathbb{E}[d_k d_k^T] - 2\mathbf{w}^T \mathbb{E}[d_k \mathbf{x}_k] + \mathbf{w}^T \mathbb{E}[\mathbf{x}_k \mathbf{x}_k^T] \mathbf{w} \\
 &= \sigma_d^2 - 2\mathbf{w}^T \sigma_{dx} + \mathbf{w}^T \sigma_x \mathbf{w}
 \end{aligned}$$

但し, $\sigma_d^2 = \mathbb{E}[d_k d_k^T]$, $\sigma_{dx} = \mathbb{E}[d_k \mathbf{x}_k]$, $\sigma_x = \mathbb{E}[\mathbf{x}_k \mathbf{x}_k^T]$ である.

評価関数の偏微分を求め, 0 とおく.

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -2\sigma_{dx} + 2\sigma_x \mathbf{w} = 0$$

ここから, 以下の最適解を得る.

$$\mathbf{w}_{opt} = \sigma_x^{-1} \cdot \sigma_{dx}$$

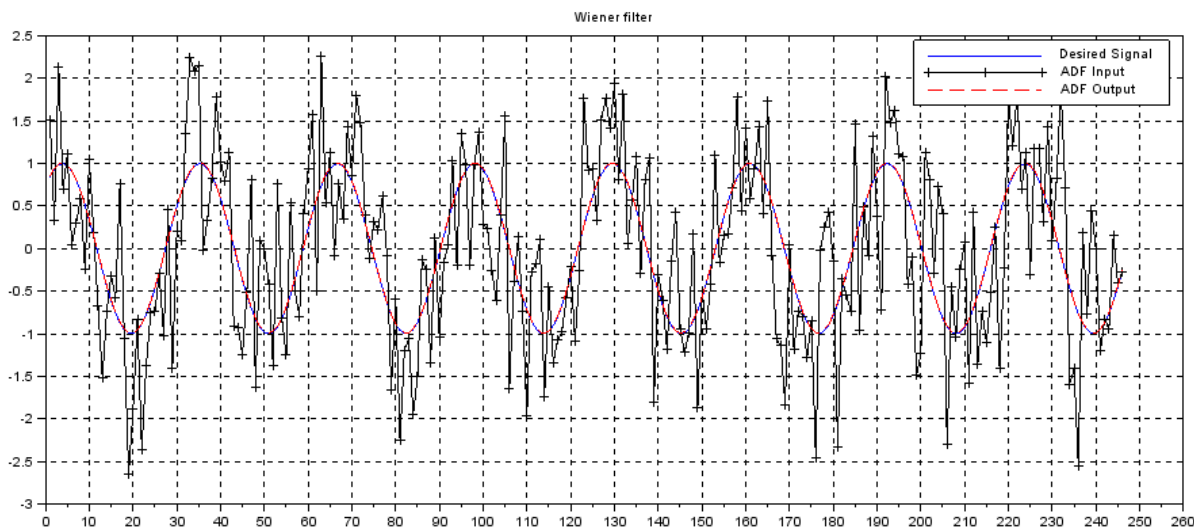


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```

////////////////////////////////////
//      ウィナーフィルタ
//      Wiener filter
//
//
//                      M.Tsutsui
////////////////////////////////////

clear all;

Data=50;//データ数
d=sin(1:0.2:Data)';//所望信号
Data_renew=size(d,1);
v=2*rand(Data_renew,1)-2*rand(Data_renew,1);//ノイズ
x=d+v;//適応フィルタ入力
[L1,L2]=size(d);
E_d=mean(d);
E_x=mean(x);
sigma_x=mvvacov(x);
sigma_d_x=1/L1*(x-E_x*ones(Data_renew,1))*(d-E_d*ones(Data_renew,1))';

w_re=inv(sigma_x)*sigma_d_x;

Out_opt=w_re'*x;//係数更新後フィルタ出力
plot(d);//所望信号
plot(x,'k+-');//適応フィルタ入力
plot(Out_opt,'r--');//適応フィルタ出力
xgrid();
hl=legend(['Desired_Signal';'ADF_Input';'ADF_Output']);
title('Wiener_filter');

```

Source Code 2: Python

```

#-----
# Module Name: Wiener filter
# Author: m_tsutsui
#-----

#Library_Import-----
from numpy import*
import math, numpy as np
import matplotlib.pyplot as plt
#-----

if __name__ == '__main__':

    Data=50 #Data Number
    sita=linspace(0,Data,100)

    #-----Desired Signal-----
    d=sin(sita) #Desired_Signal
    d_M=matrix(d).T #array->matrix
    [Data_renew,Data_renew_p]=d_M.shape

    L=np.size(d,0) #size_check
    ones_block=matrix(np.ones((Data_renew,1)))

    #-----Noise-----
    NG=2 #noise gain
    v=NG*(np.random.rand(Data_renew,1)-np.random.rand(Data_renew,1))
    v_M=matrix(v)#array->matrix

    x=(d_M+v_M) #ADF_In

```

```

#-----Expectation-----
E_d=np.mean(d)
E_x=np.mean(x)

#-----covariance-----
sigma_x=np.cov(x.T)#x_covariance
sigma_x_M=matrix(sigma_x)#array->matrix
sigma_d_x=1/L*(x-E_x*ones_block)*(d_M-E_d*ones_block).T

w_re=1/sigma_x*sigma_d_x #coefficient update

y_opt=w_re.T*x #ADF_Out

#plot_command-----
plt.figure(facecolor='w')
plt.plot(d)
plt.plot(array(x),'k')
plt.plot(array(y_opt),"r--")
plt.grid()
plt.legend(('Desired_Signal','ADF_Input','ADF_Output'))
plt.title('Wiener_filter')
plt.show()

```