

【適応フィルタ (再帰最小二乗法) Adaptive Filter (Recursive Least Squares Method)】

■変数・パラメータ

\mathbf{x}_k : フィルタ入力, \mathbf{w}_k : フィルタ係数, d_k : 所望信号, $\mathbf{R}_k = \sum_{i=1}^k \mathbf{x}_i \mathbf{x}_i^H$: 自己相関行列

$\mathbf{P}_k = \mathbf{R}_k^{-1}$: 自己相関行列の逆行列, λ : 忘却係数 ($0 < \lambda \leq 1$), α : 小さい正の定数

■ RLS Algorithm

初期値

$$\mathbf{P}_0 = \alpha^{-1} \mathbf{I}$$

$$\mathbf{w}_0 = \mathbf{0}$$

更新

For

$$\mathbf{g}_k = \frac{\lambda^{-1} \mathbf{P}_{k-1} \mathbf{x}_k}{1 + \lambda^{-1} \mathbf{x}_k^H \mathbf{P}_{k-1} \mathbf{x}_k} \quad \text{:ゲインベクトル}$$

$$e_k = d_k - \mathbf{w}_{k-1}^H \mathbf{x}_k \quad \text{:推定誤差}$$

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mathbf{g}_k e_k^* \quad \text{:係数ベクトル}$$

$$\mathbf{P}_k = \lambda^{-1} (\mathbf{I} - \mathbf{g}_k \mathbf{x}_k^H) \mathbf{P}_{k-1} \quad \text{:自己相関行列の逆行列}$$

$$y_k = \mathbf{w}_k^H \mathbf{x}_k \quad \text{: フィルタ出力}$$

end

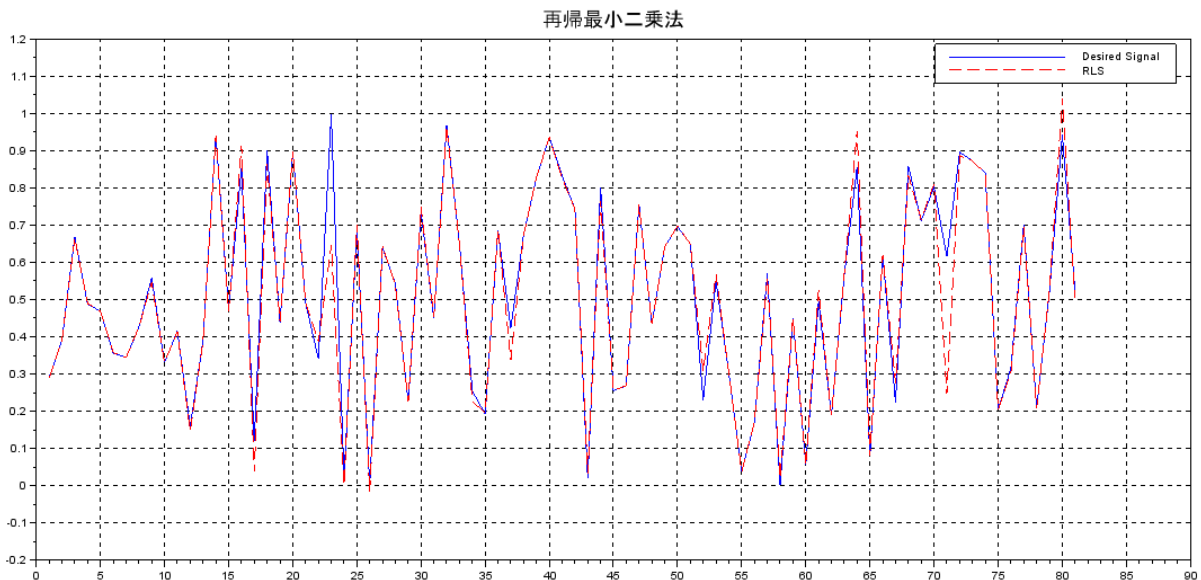


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```

////////////////////////////////////
//  適応フィルタ(再帰最小二乗法)
//  Adaptive Filter
//  Recursive least squares
//
//                                     M.Tsutsui
////////////////////////////////////

```

```

clear;

funcprot(0);
function[y_opt_buf]=RLS(arufa,lambda,update);//arufa:  $\alpha$  ,lambda:忘却係数 ,update:更新回数

    y_opt_buf=[];//ADF出力バッファ

    P_ini=1/arufa*eye(size_r2,size_r2);//P初期値

    for s_loop=1:1:size_r2;//__サンプル変化
        for i=1:1:update;//__係数更新ループ__

            gain=(1/lambda*(P_ini*x))/(1+1/lambda*x'*P_ini*x);//ゲインベクトル

            e=d(1,s_loop)-w_ini'*x;//誤差 サンプルループ

            w_ini=w_ini+gain*e;//係数更新

            P_ini=1/lambda*(eye(size_r2,size_r2)-gain*x'*P_ini);//自己相関行列更新

        end//_____係数更新ループ__

        y_opt=w_ini'*x;//ADF出力 1Sample
        y_opt_buf=[y_opt_buf,y_opt];//ADF出力ベクトル

    end//_____サンプル変化___

endfunction

d_size=80;//データサイズ

d=rand(0:1:d_size);//所望信号

[size_r,size_r2]=size(d);//サイズ更新

w_ini=zeros(size_r2,1);//適応フィルタ初期係数

x=rand(size_r2,1);//フィルタ入力

plot(d);
plot(RLS(0.01,0.4,5),'r--');
xgrid();
title('再帰最小二乗法','fontsize',4);
legend(['Desired_Signal','RLS']);

```

Source Code 2: Python

```

#-----
# Name:Adaptive Filter Recursive least squares
# Author: m_tsutsui
#-----

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

def RLS(arufa,lamda,update): #RLS Algorithm
    """
    """
    arufa:留, lamda:forgetting_factor

```

```

update:Update_Count

"""
y_opt_buf=[]

P_ini=1/arufa*np.eye(d_size,d_size)

for s_loop in np.arange(0,d_size,1):# d sample loop
    for i in np.arange(1,update+1,1):
        gain=(1/lamda*(matrix(P_ini)*matrix(x)))*(1+1/lamda*matrix(x).T*matrix(P_ini)*matrix(x)).I #gain vector
        e=d[0,s_loop]-matrix(w_ini).T*matrix(x) #error
        global w_ini
        w_ini=matrix(w_ini)+matrix(gain)*e
        P_ini=1/lamda*(np.eye(d_size,d_size)-matrix(gain)*matrix(x).T)*matrix(P_ini)

    y_opt=float(matrix(w_ini).T*matrix(x)) #ADF out (1sample)
    y_opt_buf.append(y_opt) #ADF out vector

return y_opt_buf

if __name__ == '__main__':

    d_size=50 #data size

    d=np.random.rand(1,d_size) #Desired Signal

    w_ini=np.zeros([d_size,1]) #initial coefficient

    x=np.random.rand(d_size,1) #filter input

#_plot_command_#####
plt.figure(facecolor='w')
plt.plot(array(d).T)
plt.plot(array(RLS(0.01,0.4,5)).T,"r--")
plt.grid()
plt.legend(('Desired_Signal','RLS'))
plt.title('Recursive_Least_Squares')
plt.show()
#_end_#####

```