

# 【適応フィルタ (アフィン射影法) Adaptive Filter (Affine Projection Algorithm)】

[導出]

$$\arg \min_{\mathbf{w}_k} \|\mathbf{w}_k - \mathbf{w}_{k-1}\|^2 \quad \text{subject to } \mathbf{w}_k^H \mathbf{X}_k = \mathbf{d}_k$$

$\mathbf{X}_k$  は  $K \times L$  行列で,  $\mathbf{d}_k$  は  $1 \times L$  行列である.

Lagrange の未定数法を用いて, 評価関数は次のようになる.

$$J = (\mathbf{w}_k - \mathbf{w}_{k-1})^H (\mathbf{w}_k - \mathbf{w}_{k-1}) + \boldsymbol{\lambda}^H (\mathbf{d}_k - \mathbf{w}_k^H \mathbf{X}_k) + (\mathbf{d}_k - \mathbf{w}_k^H \mathbf{X}_k) \boldsymbol{\lambda}$$

$\boldsymbol{\lambda}$  は,  $L \times 1$  行列である. これを  $\mathbf{w}_k^*$  について微分し, 0 とおくと,

$$\mathbf{w}_k - \mathbf{w}_{k-1} = \mathbf{X}_k \boldsymbol{\lambda} \quad (*)$$

ここから,

$$\boldsymbol{\lambda} = (\mathbf{X}_k^H \mathbf{X}_k)^{-1} (\mathbf{d}_k - \mathbf{w}_{k-1}^H \mathbf{X}_k)^H$$

上式と (\*) より,

$$\mathbf{w}_k - \mathbf{w}_{k-1} = \mathbf{X}_k (\mathbf{X}_k^H \mathbf{X}_k)^{-1} (\mathbf{d}_k - \mathbf{w}_{k-1}^H \mathbf{X}_k)^H$$

誤差ベクトルを  $\mathbf{e}_k = \mathbf{d}_k - \mathbf{w}_{k-1}^H \mathbf{X}_k$  とすると, 更新式は次のようになる.

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mu \mathbf{X}_k (\alpha \mathbf{I} + \mathbf{X}_k^H \mathbf{X}_k)^{-1} \mathbf{e}_k^H \quad \text{※ } \mathbf{X}_k^H \mathbf{X}_k \text{ の対角成分に } \alpha \text{ を加え正則化}$$

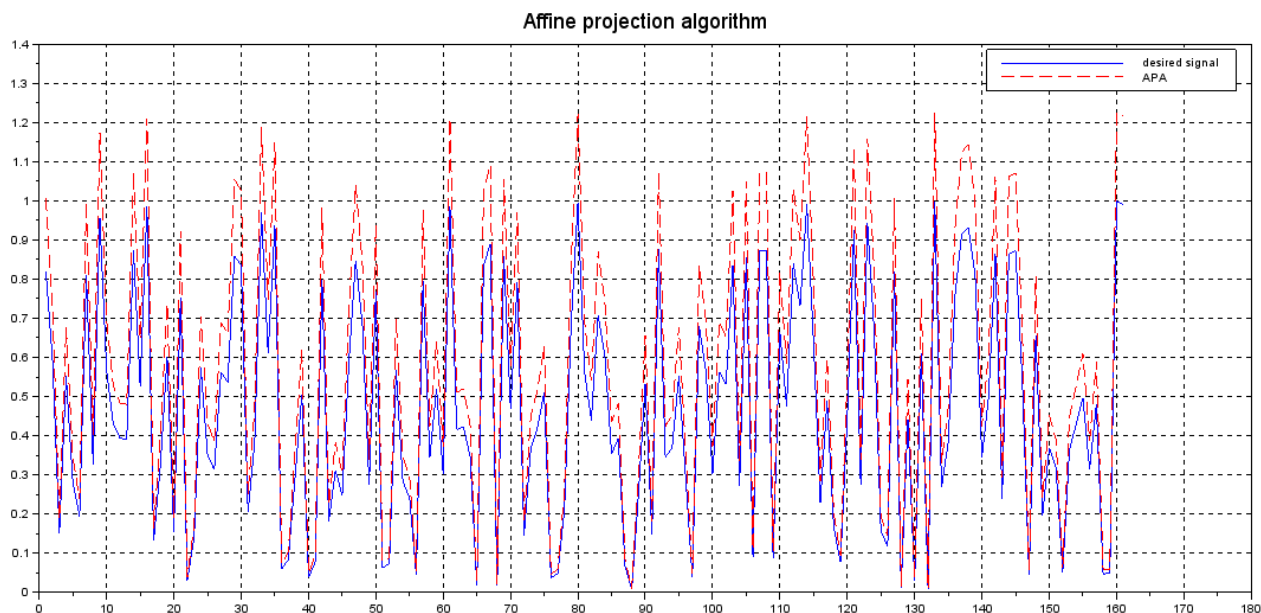


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```

////////////////////////////////////
//  適応フィルタ(アフィン射影法)
//  Adaptive Filter
//  Affine projection algorithm
//  M.Tsutsui
////////////////////////////////////

clear all;

```

```

funcprot(0)
function [y_opt]=APA(myu,arufa,RC);
//-----APA関数 -----//
//
// myu: ステップサイズ, arufa: 小さい正の定数(正則化)※逆行列の安定性確保
//
// RC: 更新回数
//
//-----APA関数 -----//

for i=1:1:RC-1;

    e=d-w_ini'*X;//誤差ベクトル

    w_ini=w_ini+myu*X*inv(arufa*eye(size_r2,size_r2)+X'*X)*e';//正規化

end

y_opt=w_ini'*X;//フィルタ出力

endfunction

d_size=80;//データサイズ

d=rand(0:0.5:d_size);//所望信号ベクトル

[size_r,size_r2]=size(d);//サイズ更新

w_ini=rand(size_r2,1);//適応フィルタ初期係数

X=w_ini*d;//入力ベクトル

plot(d);
plot(APA(0.5,3,9),'r--');
xgrid();
legend(['desired_signal','APA']);
title('Affine_projection_algorithm','fontsize',4);

```

## Source Code 2: Python

```

#-----
# Module Name:Adaptive Filter
#             Affine projection algorithm
# Author: m_tsutsui
#-----

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

def APA(myu,arufa,UC):
    """
    """
    """APA_function

    """
    """myu:step_size,arufa:regularisation_const

    """
    """UC:Update_count
    """
    """
    for i in arange(1,UC+1,1):

```

```

    global w_ini
    e=matrix(d)-matrix(w_ini).T*matrix(X) #error vector
    w_ini=w_ini+myu*matrix(X)*(arufa*np.eye(d_size,d_size)+matrix(X).T*matrix(X)).I*matrix(e).T

y_opt=matrix(w_ini).T*matrix(X) #filter out

return y_opt

if __name__ == '__main__':

    d_size=80 #data size

    w_ini=np.random.rand(d_size,1) #initial coefficient

    d=np.random.rand(1,d_size) #desired signal

    X=matrix(w_ini)*matrix(d) #input vector

    APA_out=APA(0.5,3,8)

#plot_command#####
    plt.figure(facecolor='w')
    plt.plot(d.T)
    plt.plot(APA_out.T,"r--")
    plt.grid()
    plt.legend(('desired_signal','APA'))
    plt.title('Affine_projection_algorithm',fontsize=20)
    plt.show()

```