

【適応フィルタ (LMS Algorithm) Adaptive Filter (LMS Algorithm)】

所望の信号 d_k , 適応フィルタ出力 y_k , 推定誤差 $e_k = d_k - y_k$
 フィルタ入力 $\mathbf{x} = [x_0, x_1, \dots, x_n]^T$, フィルタ係数 $\mathbf{w} = [w_0, w_1, \dots, w_n]^T$ とする.

評価関数を次のように定義する.

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

なお, $\mathbb{E}[d_k \mathbf{x}_k] = \mathbf{p}, \mathbb{E}[\mathbf{x} \mathbf{x}^T] = \mathbf{R}$ とおいた,
 \mathbf{w} の二次形式であることを考慮し, 最急降下法の考えを用いると, 次のアルゴリズムが成立する.

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{1}{2} \mu \nabla J(\mathbf{w}_k)$$

ここで,

$$\nabla J(\mathbf{w}_k) = \left. \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}=\mathbf{w}_k} = 2(\mathbf{R} \mathbf{w}_k - \mathbf{p})$$

であるので, ステップサイズ μ を用いて, 次のように書ける.

$$\mathbf{w}_{k+1} = (\mathbf{I} - \mu \mathbf{R}) \mathbf{w}_k + \mu \mathbf{p}$$

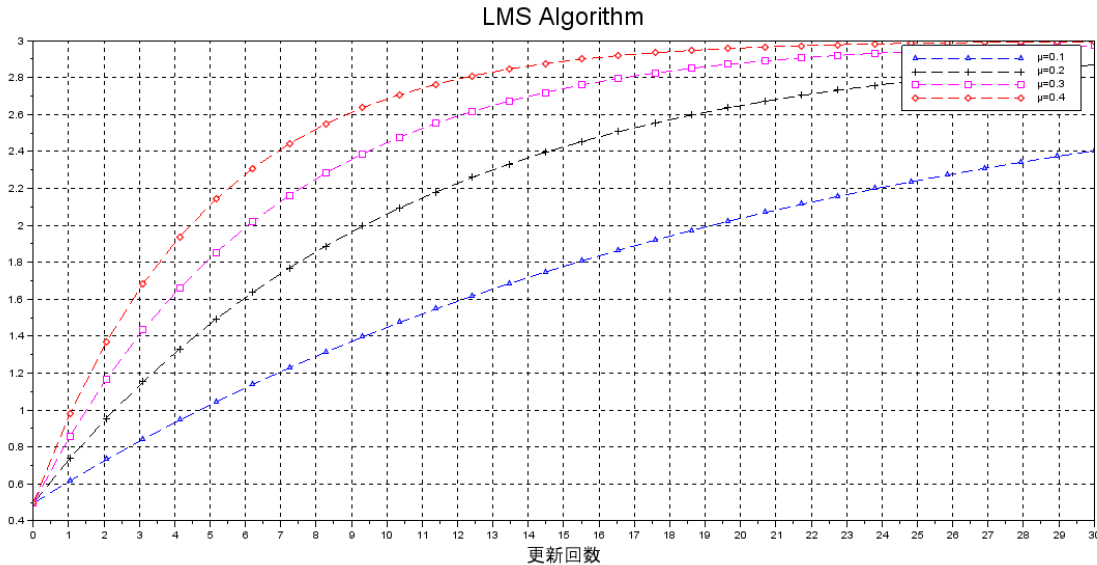


Figure 1: Scilab 実行結果
 収束変化 ($\mu=0.1\sim 0.4$) 収束値:3 更新回数:30

Source Code 1: Scilab

```

////////////////////////////////////
//  適応フィルタ(LMSアルゴリズム)
//  Adaptive Filter(LMS Algorithm)
//
//                                     M.Tsutsui
////////////////////////////////////

clear all;

```

```

funcprot(0)
function [y_opt]=lms_v(myu,update);//myu:ステップサイズ,update:更新回数

R=x*x'; //E[x*x']
p=d*x; //E[d*x];

buf=[];//バッファ用意(収束変化確認)
for n=1:update-1;
    w_renew=(eye(c_size,c_size)-myu*R)*w_renew+myu*p;//フィルター係数更新
    buf=[buf,w_renew];
end

y_opt=[];
for i=1:update-1;
    y_opt=[y_opt,buf(:,i)']*x;//係数更新後フィルタ出力
end

endfunction

c_size=2;//係数サイズ

d=3;//所望サンプル値

x=rand(c_size,1);//入力信号

w_renew=rand(c_size,1);//適応フィルタ初期係数

y_opt_ini=w_renew'*x;//初回出力

//__plot_____
x_a=linspace(0,30,30);//0から描画
plot(x_a,[y_opt_ini,lms_v(0.1,30)],'b^--');
plot(x_a,[y_opt_ini,lms_v(0.2,30)],'k+--');
plot(x_a,[y_opt_ini,lms_v(0.3,30)],'ms--');
plot(x_a,[y_opt_ini,lms_v(0.4,30)],'rd--');
xgrid();
xlabel("更新回数", "fontsize", 4);
legend([' $\mu=0.1$ '; ' $\mu=0.2$ '; ' $\mu=0.3$ '; ' $\mu=0.4$ '; ' $\mu=0.5$ ']);
title('LMS_Algorithm','fontsize',5);

```

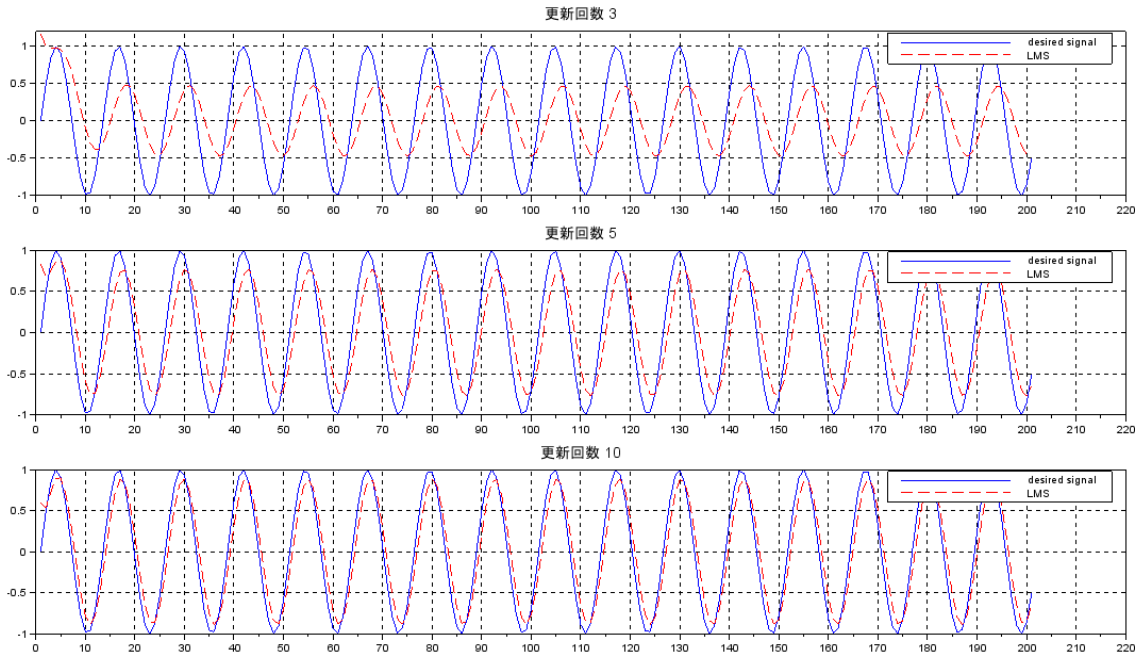


Figure 2: Scilab 実行結果

Source Code 2: Scilab

```

////////////////////////////////////
//  適応フィルタ(LMSアルゴリズム)
//  Adaptive Filter(LMS Algorithm)
//
//                                     M.Tsutsui
////////////////////////////////////

clear all;

c_size=3;//係数サイズ
d=sin(0:0.5:100)';//所望サンプル→所望信号

[size_r,size_r2]=size(d);//サイズ更新

x=0.1*rand(size_r,1);//入力信号

w=0.3*rand(size_r,1);//適応フィルタ係数

R=x*x'; //E[x*x']

funcprot(0)
function [y_opt]=lms_agm(myu,update);//myu:ステップサイズ,update:更新回数

p_buf=[];//p サンプルd変化
for i=1:1:size_r;
    p_buf=[p_buf,d(i,1)*x];
end

w_buf=[];//係数バッファ
for p_loop=1:1:size_r;
    for n=1:update-1;
        w=(eye(size_r,size_r)-myu*R)*w+myu*p_buf(:,p_loop);//フィルタ係数更新
    end
    w_buf=[w_buf,w];

```

```

end

y_opt=w_buf'*x;//適応フィルタ出力

endfunction

//__plot_____

subplot(3,1,1)
plot(d);
plot(lms_agm(0.1,5),'r--');
xgrid();
legend(['desired_signal','LMS']);
title('更新回数_3','fontsize',3);

subplot(3,1,2)
plot(d);
plot(lms_agm(0.1,10),'r--');
xgrid();
legend(['desired_signal','LMS']);
title('更新回数_5','fontsize',3);

subplot(3,1,3)
plot(d);
plot(lms_agm(0.1,15),'r--');
xgrid();
legend(['desired_signal','LMS']);
title('更新回数_10','fontsize',3);

```

Source Code 3: Python

```

#-----
# Module Name: Adaptive Filter LMS Algorithm
# Author m_tsutsui
#-----

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

def lms_off(myu,update,samp_n):
    """
    myu:step_size,update:update_count

    samp_n:desired_signal_sample_number

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

    for n in np.arange(1,update,1):
        w=(np.eye(d_size,d_size)-array(myu)*matrix(R))*matrix(w)+array(myu)*d[samp_n,0]*matrix(x)
        w_opt=w

    y_opt=matrix(w_opt).T*matrix(x)

    return y_opt #ADF 1 sample out

if __name__ == '__main__':

```

```

d_size=50 #data size

t=matrix(np.linspace(0,d_size,d_size)).T
d=array(sin(t)) #Desired Signal (sine wave)

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

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

R=matrix(x)*matrix(x).T #E[x ·x ']

ADF_out=[]
for j in np.arange(0,d_size,1):
    ADF_buf=float (lms_off(0.03,7,j))
    ADF_out.append(ADF_buf) #ADF out

#_plot_command_#####
plt.figure(facecolor='w')#Backgroundcolor_white
plt.plot(array(d))
plt.plot(array(ADF_out),"r--")
plt.grid()
plt.legend(('Desired_Signal','LMS'))
plt.title('LMS_Algorithm(off_line)')
plt.show()
#_end_#####

```