Assessment metric

The aim of these filters is to remove echoes, i.e. make the output resemble the `loudspeaker` input as much as possible. Hence, we use the following metric to assess how well we have reduced the echo.

`norm(err)`

Inputs and Outputs

In task 1 (a), (b) and (c), we use `mike - loudspeaker` (i.e. pure echoes) as the *desired signal*. We try to estimate the filter impulse response θ that makes `loudspeaker` become `mike - loudspeaker`.

After we obtain θ, we take `mike - transpose(theta) * phi` as the echo-cancellation output.

Task 1 (a) constant echo amplitude

RLS

Initial conditions

$$P = \rho \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Noise-free environment

By trial and error, we find when $\lambda = 0.999$, $\rho = 1000$, the RLS filter has best echo-cancellation performance.

`norm(err)` $= 0.004143$

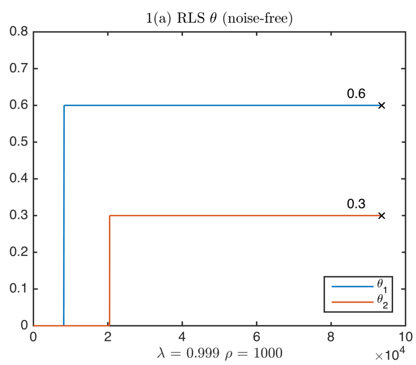

Figure 1: RLS θ trends

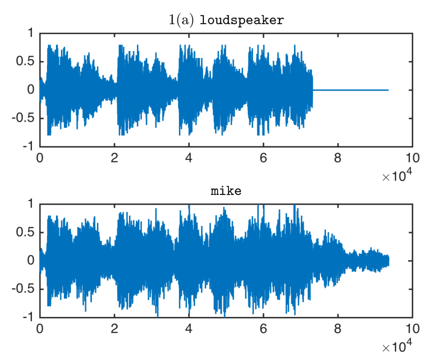

Figure 2: inputs

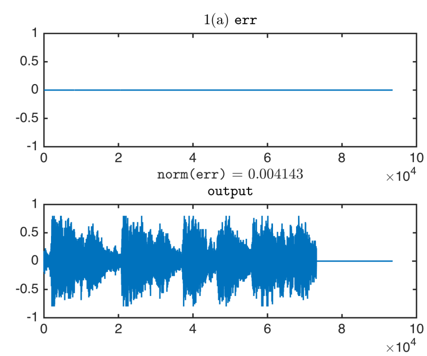

Figure 3: output and comparison

In Fig. 1, θ_1 and θ_2 eventually converge to 0.6 and 0.3. In Fig. 3, echoes are successfully suppressed and `err` is negligible comparing with `mike1`.

```
Elapsed time is 1.341335 seconds.
lambda = 0.999000
rho = 1000.000000
norm(err) = 0.004143
```

Improper choice of λ and ρ

In Fig. 4, smaller $\lambda = 0.998$ leads to severe overshoot. If we reduce λ further, θ will become zero.

In Fig. 5, smaller $\rho = 0.001$ results in longer transition time.

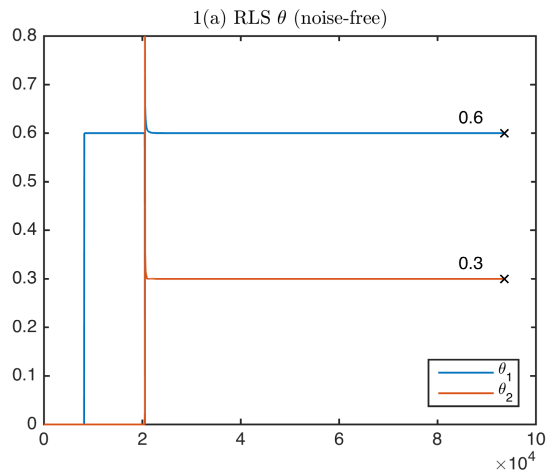

Figure 4: small λ

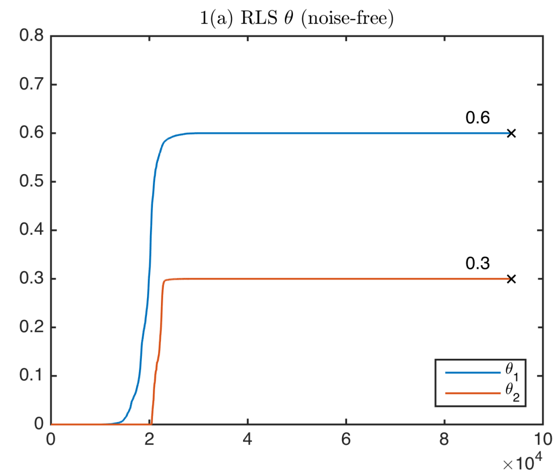

Figure 5: small ρ

Noisy environment

By trial and error, we find when $\lambda = 0.999$, $\rho = 0.1$, the RLS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 4.849423 > 0.004143$$

`norm(err)` increases due to the interference of the background noise.

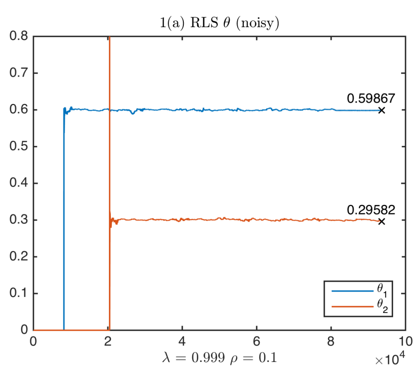

Figure 6: RLS θ trends

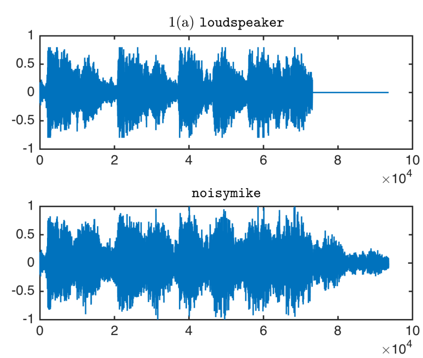

Figure 7: inputs

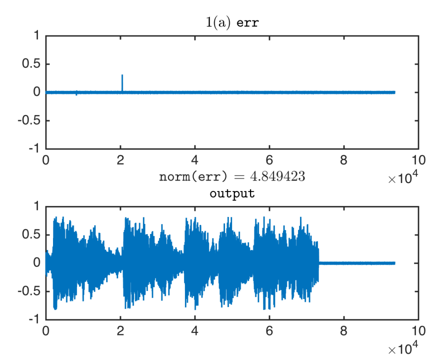

Figure 8: output and comparison

In Fig. 6, θ_1 and θ_2 eventually converge to 0.599 and 0.296. In Fig. 8, echoes are successfully suppressed and `err` is negligible comparing with `noisymike1`.

LMS

Noise-free environment

By trial and error, we find when `step_size` $= 2\mu = 10$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 0.158492$$

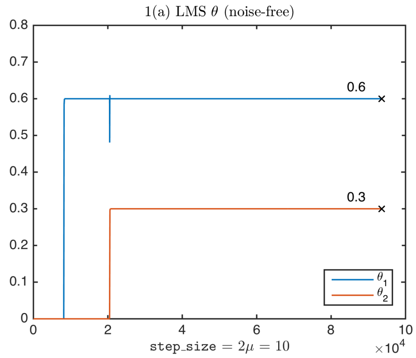

Figure 9: LMS θ trends

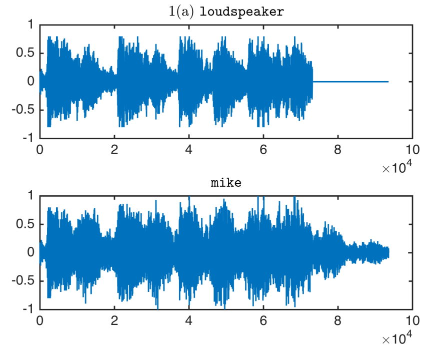

Figure 10: inputs

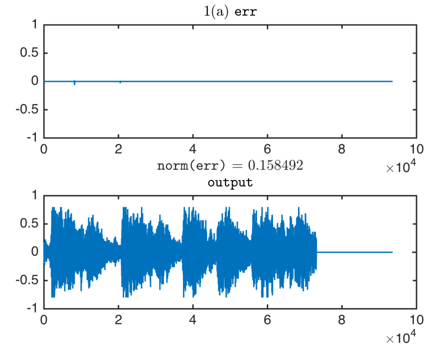

Figure 11: output and comparison

In Fig. 9, θ_1 and θ_2 eventually converge to 0.6 and 0.3. In Fig. 11, echoes are successfully suppressed and `err` is negligible comparing with `mike1`.

```
Elapsed time is 0.603833 seconds.
step_size = 10.000000
norm(err) = 0.158492
```

Improper choice of μ

In Fig. 12, bigger `step_size` $= 2\mu = 11$ leads to instability. If we increase `step_size` further, the LMS filter will diverge and become unstable.

In Fig. 13, smaller `step_size` $= 2\mu = 0.001$ results in longer transition time.

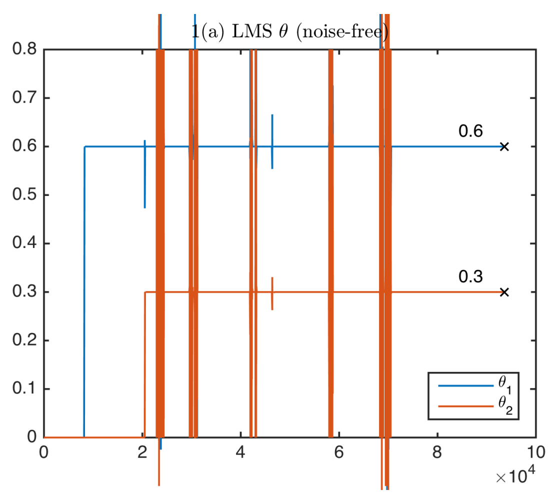

Figure 12: big μ

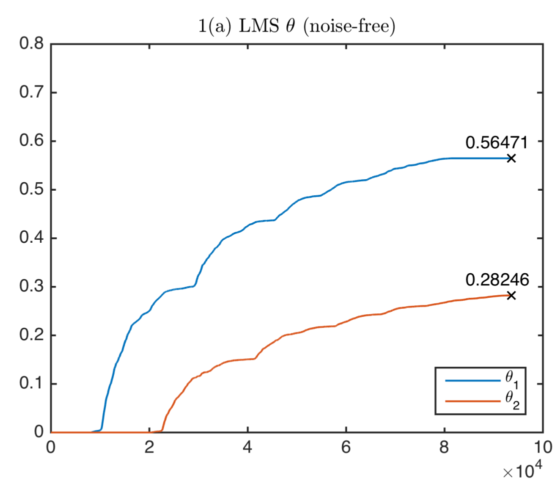

Figure 13: small μ

Noisy environment

By trial and error, we find when `step_size` $= 2\mu = 0.5$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 4.919796 > 0.158492$$

`norm(err)` increases due to the interference of the background noise.

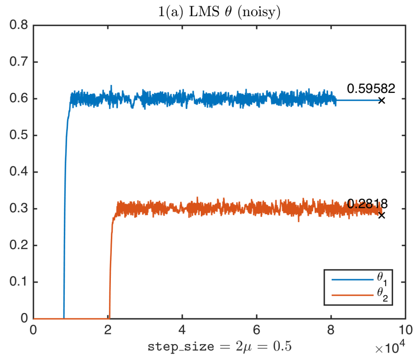

Figure 14: LMS θ trends

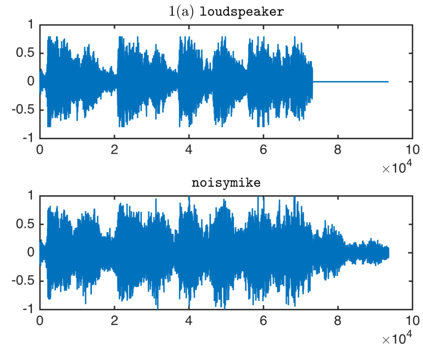

Figure 15: inputs

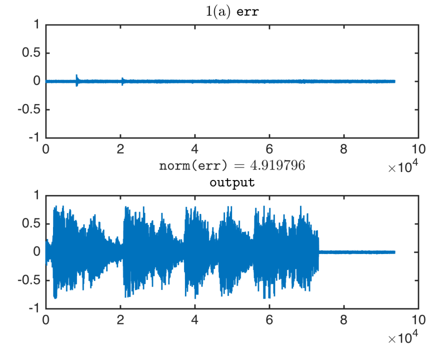

Figure 16: output and comparison

In Fig. 14, θ_1 and θ_2 eventually converge to 0.596 and 0.282. In Fig. 16, echoes are successfully suppressed and `err` is negligible comparing with `noisymike1`.

RLS & LMS comparison

θ_1 and θ_2 in both RLS and LMS converge to 0.3 and 0.6 rapidly. However, when the second echo arrives (at 2.5 s), θ_1 in LMS has an abnormal spike (shown in Fig. 9). Considering the echo amplitudes are constant, RLS is expected to perform better than LMS. In fact, `norm(err)` of RLS is smaller than LMS (0.004143 vs 0.158492).

We use `tic` and `toc` in MATLAB to measure time to run RLS or LMS algorithm. RLS is more time-consuming than LMS.

$$\frac{\text{RLS}}{\text{LMS}} = \frac{1.341335 \text{ seconds}}{0.603833 \text{ seconds}} = 2.22$$

Task 1 (b) time-varying echo amplitude

RLS

Noise-free environment

By trial and error, we find when $\lambda = 0.999$, $\rho = 1000$, the RLS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 2.125772$$

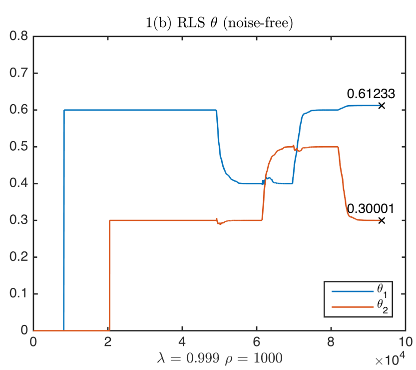

Figure 17: RLS θ trends

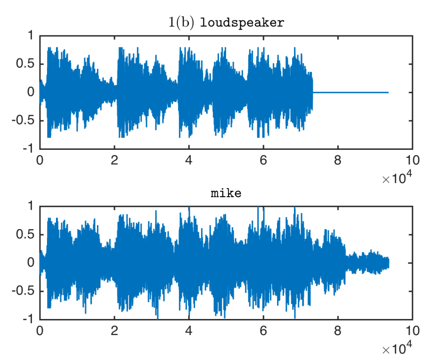

Figure 18: inputs

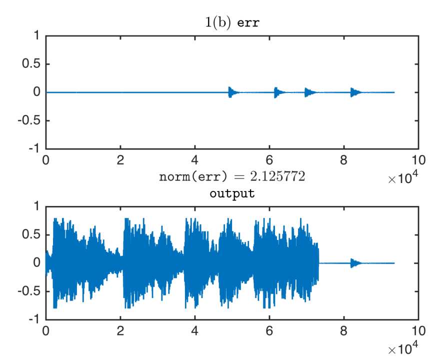

Figure 19: output and comparison

In Fig. 17, θ_1 and θ_2 eventually converge to 0.612 and 0.300. In Fig. 19, echoes are successfully suppressed and `err` is negligible comparing with `mike2`.

Noisy environment

By trial and error, we find when $\lambda = 0.999$, $\rho = 0.6$, the RLS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 5.294251 > 2.125772$$

`norm(err)` increases due to the interference of the background noise.

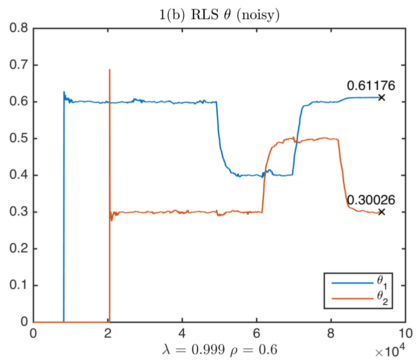

Figure 20: RLS θ trends

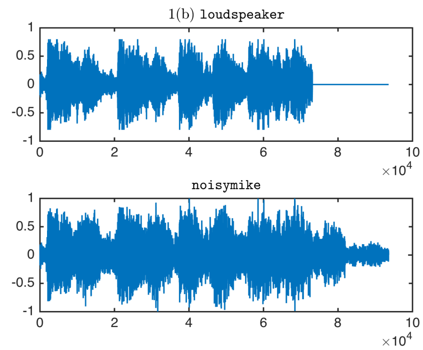

Figure 21: inputs

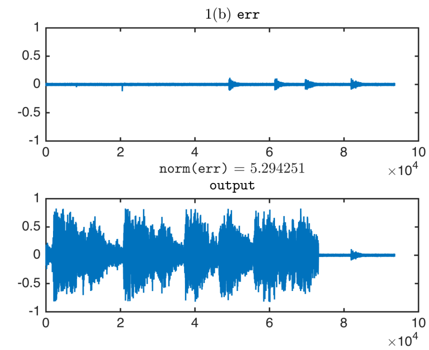

Figure 22: output and comparison

In Fig. 20, θ_1 and θ_2 eventually converge to 0.612 and 0.300. In Fig. 22, echoes are successfully suppressed and `err` is negligible comparing with `noisymike2`.

LMS

Noise-free environment

By trial and error, we find when `step_size` $= 2\mu = 10$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 0.210652$$

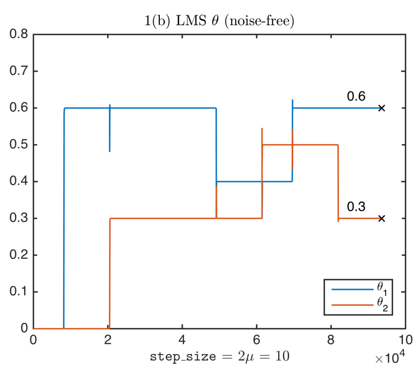

Figure 23: LMS θ trends

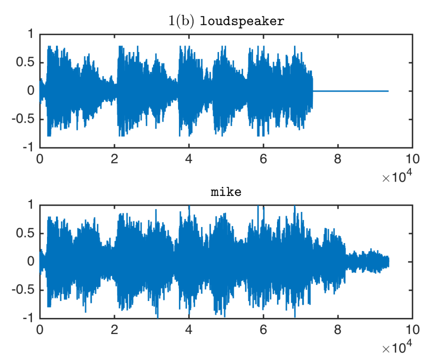

Figure 24: inputs

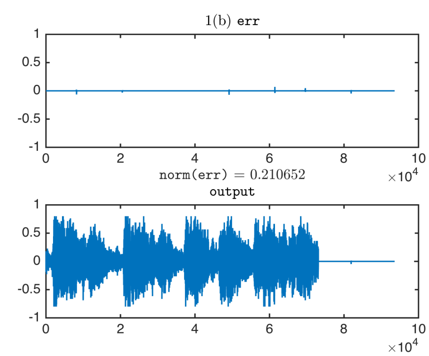

Figure 25: output and comparison

In Fig. 23, θ_1 and θ_2 eventually converge to 0.6 and 0.3. In Fig. 25, echoes are successfully suppressed and `err` is negligible comparing with `mike2`.

Noisy environment

By trial and error, we find when `step_size` $= 2\mu = 0.6$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 4.943508 > 0.210652$$

`norm(err)` increases due to the interference of the background noise.

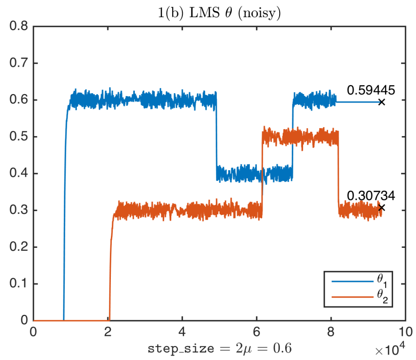

Figure 26: LMS θ trends

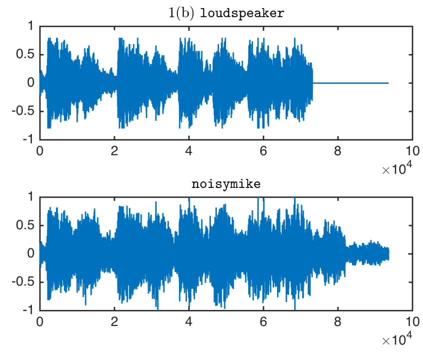

Figure 27: inputs

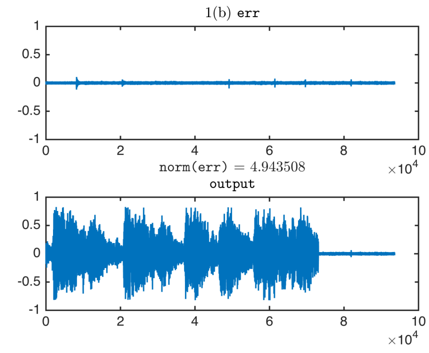

Figure 28: output and comparison

In Fig. 26, θ_1 and θ_2 eventually converge to 0.594 and 0.307. In Fig. 28, echoes are successfully suppressed and `err` is negligible comparing with `noisymike2`.

RLS & LMS comparison

As is shown in Fig. 17 and Fig. 23, θ in LMS converges faster than RLS because LMS focus more on recent data points. As a result, when the echo amplitude is time-varying, LMS can react more nimbly than RLS. In fact, `norm(err)` of LMS is smaller than RLS (0.210652 vs 2.125772).

Task 1 (c) unknown delay

RLS

$$0.001 \times 8192 \approx 8$$

New filter length

$$2 \times (8 + 1 + 8) = 34$$

We will only plot trends of θ_9 $(9 = 8+1)$ and θ_{26} $(26 = (8+1+8)+8+1)$, because other values of θ are approximate 0.

Noise-free environment

By trial and error, we find when $\lambda = 0.999$, $\rho = 100$, the RLS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 2.074119$$

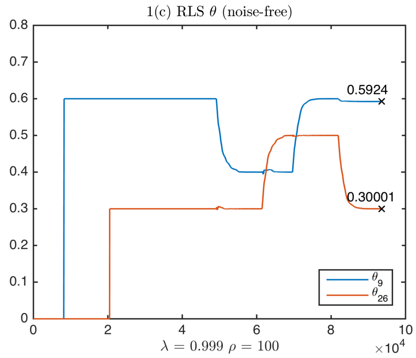

Figure 29: RLS θ trends

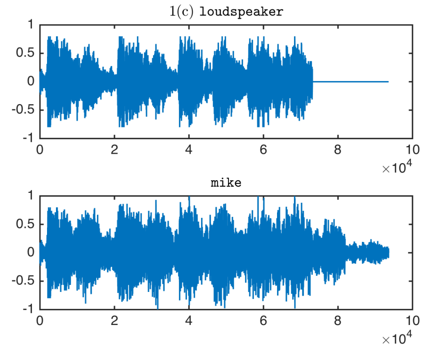

Figure 30: inputs

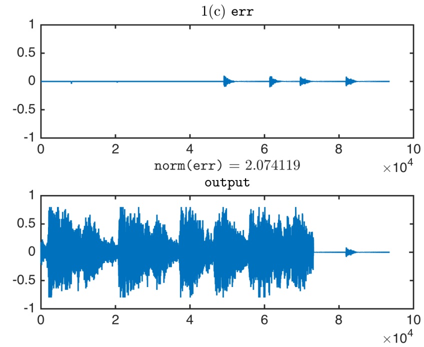

Figure 31: output and comparison

In Fig. 29, θ_9 and θ_{26} eventually converge to 0.592 and 0.300. In Fig. 31, echoes are successfully suppressed and `err` is negligible comparing with `mike2`.

Noisy environment

By trial and error, we find when $\lambda = 0.999$, $\rho = 0.001$, the RLS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 5.333936 > 2.074119$$

`norm(err)` increases due to the interference of the background noise.

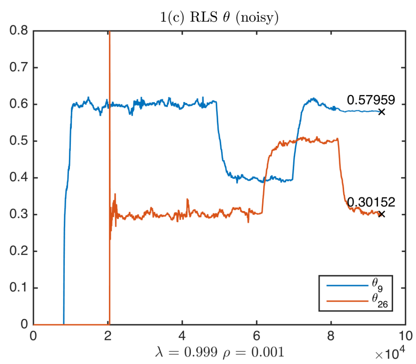

Figure 32: RLS θ trends

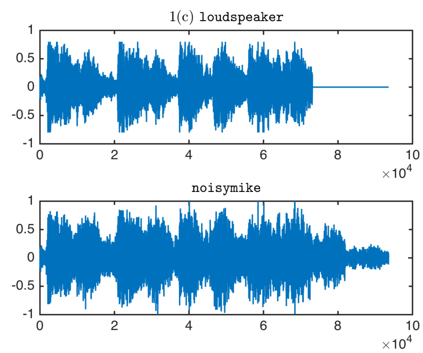

Figure 33: inputs

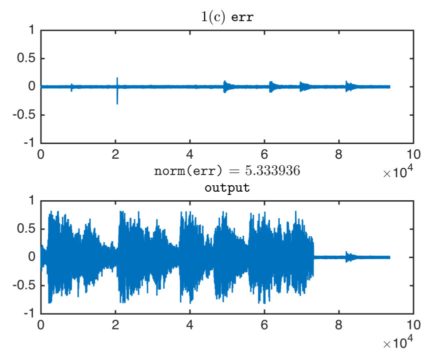

Figure 34: output and comparison

In Fig. 32, θ_9 and θ_{26} eventually converge to 0.580 and 0.302. In Fig. 34, echoes are successfully suppressed and `err` is negligible comparing with `noisymike2`.

LMS

Noise-free environment

By trial and error, we find when `step_size` $= 2\mu = 0.3$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 1.113590$$

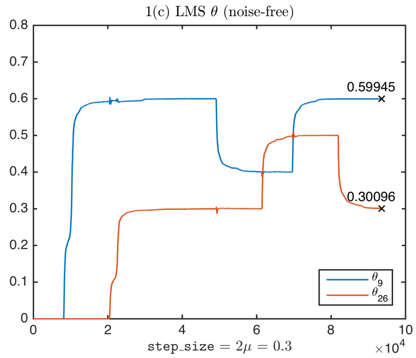

Figure 35: LMS θ trends

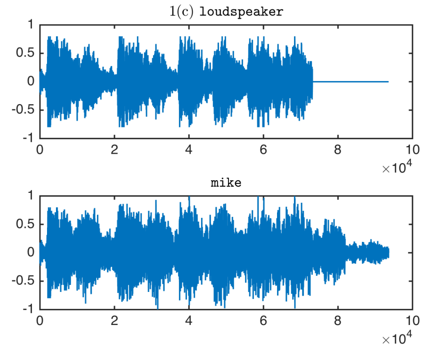

Figure 36: inputs

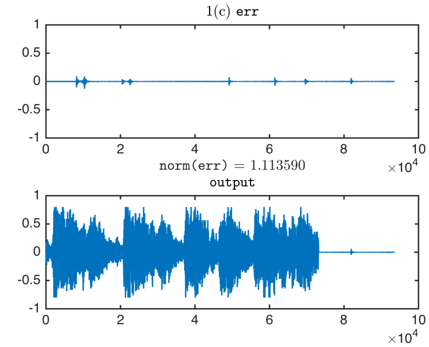

Figure 37: output and comparison

In Fig. 35, θ_9 and θ_{26} eventually converge to 0.599 and 0.301. In Fig. 37, echoes are successfully suppressed and `err` is negligible comparing with `mike2`.

Noisy environment

By trial and error, we find when `step_size` $= 2\mu = 0.1$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(err)} = 5.294403 > 1.113590$$

`norm(err)` increases due to the interference of the background noise.

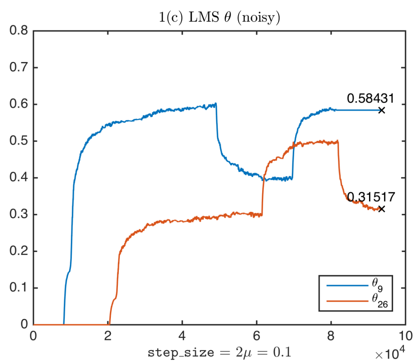

Figure 38: LMS θ trends

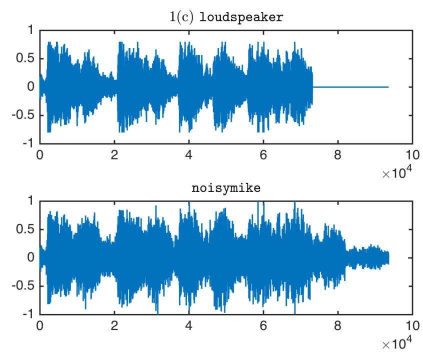

Figure 39: inputs

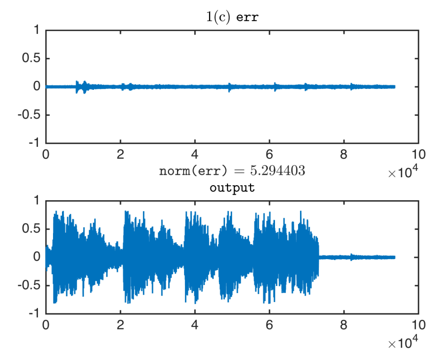

Figure 40: output and comparison

In Fig. 38, θ_9 and θ_{26} eventually converge to 0.584 and 0.315. In Fig. 40, echoes are successfully suppressed and `err` is negligible comparing with `noisymike2`.

Task 1 (d)

RLS

Noise-free environment

By trial and error, we find when $\lambda = 1$, $\rho = 3$, the RLS filter has best echo-cancellation performance.

$$\texttt{norm(output - loudspeaker)} = 3.084252$$

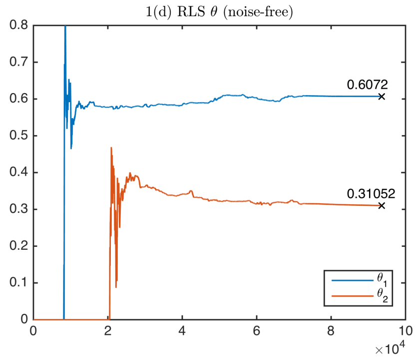

Figure 41: RLS θ trends

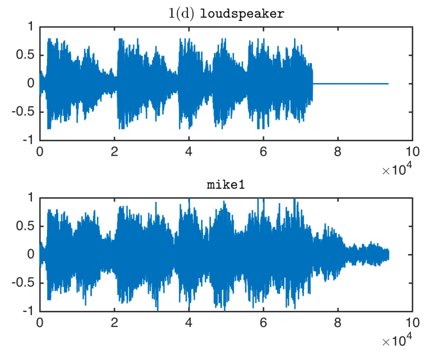

Figure 42: inputs

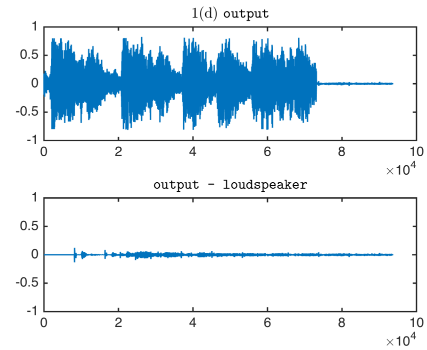

Figure 43: output and comparison

In Fig. 41, θ_1 and θ_2 eventually converge. In Fig. 43, echoes are successfully suppressed and `output - loudspeaker` is negligible comparing with `mike1`.

Noisy environment

By trial and error, we find when $\lambda = 1$, $\rho = 3$, the RLS filter has best echo-cancellation performance.

$$\texttt{norm(output - loudspeaker)} = 7.224809 > 3.084252$$

`norm(output - loudspeaker)` increases due to the interference of the background noise.

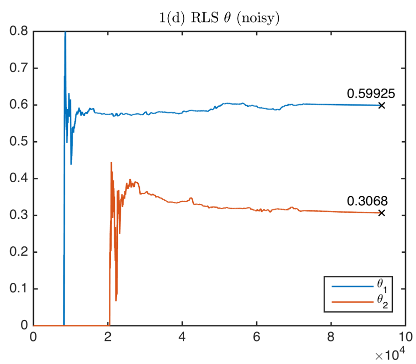

Figure 44: RLS θ trends

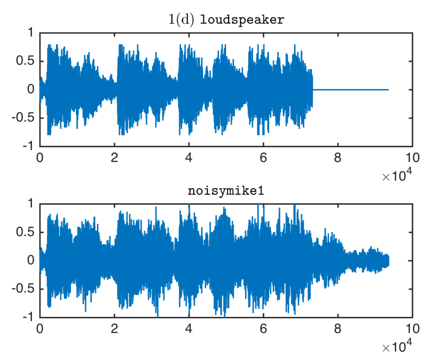

Figure 45: inputs

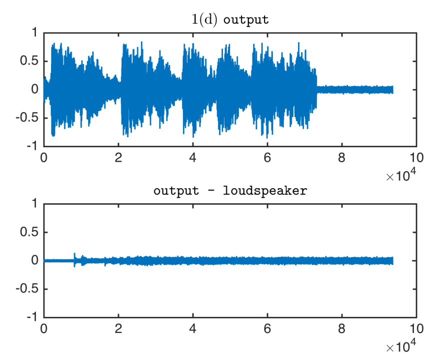

Figure 46: output and comparison

In Fig. 44, θ_1 and θ_2 eventually converge. In Fig. 46, echoes are successfully suppressed and `output - loudspeaker` is negligible comparing with `noisymike1`.

LMS

Noise-free environment

By trial and error, we find when `step_size` $= 2\mu = 0.015$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(output - loudspeaker)} = 8.113330$$

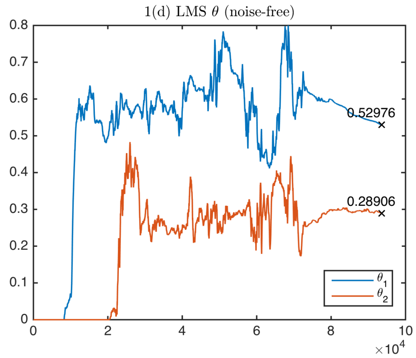

Figure 47: LMS θ trends

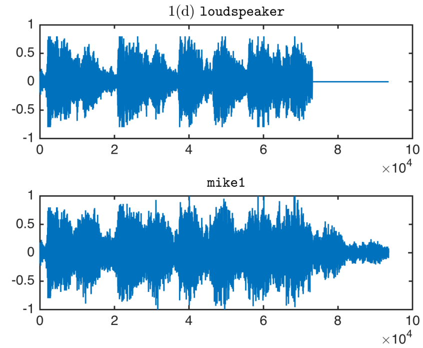

Figure 48: inputs

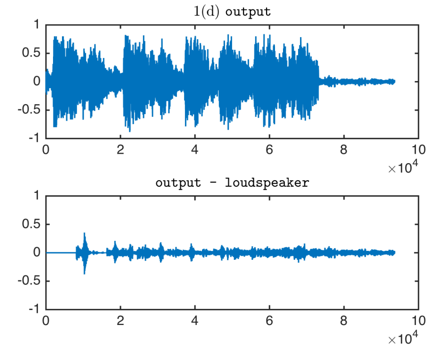

Figure 49: output and comparison

In Fig. 47, θ_1 and θ_2 eventually converge. In Fig. 49, echoes are successfully suppressed and `output - loudspeaker` is negligible comparing with `mike1`.

Noisy environment

By trial and error, we find when `step_size` $= 2\mu = 0.015$, the LMS filter has best echo-cancellation performance.

$$\texttt{norm(output - loudspeaker)} = 10.257979 > 8.113330$$

`norm(output - loudspeaker)` increases due to the interference of the background noise.

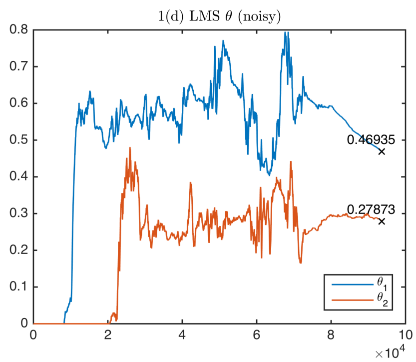

Figure 50: LMS θ trends

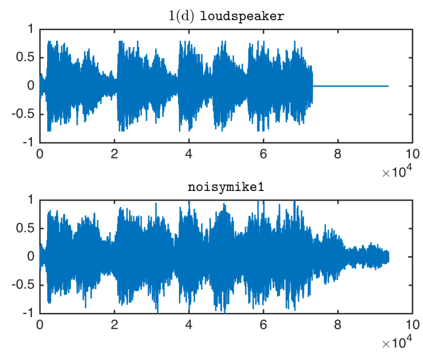

Figure 51: inputs

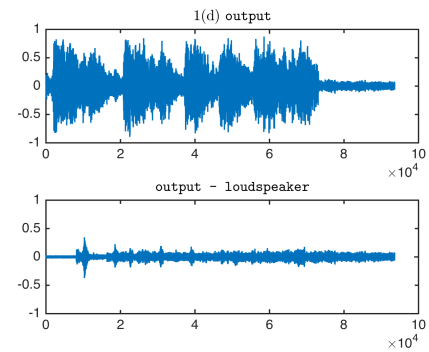

Figure 52: output and comparison

In Fig. 50, θ_1 and θ_2 eventually converge. In Fig. 52, echoes are successfully suppressed and `output - loudspeaker` is negligible comparing with `noisymike1`.

Task 2 Prelab

LMS

$$e(N) = y(N) - \phi(N)^T\hat{\theta}(N-1)$$
$$\hat{\theta}(N) = \hat{\theta}(N-1) + 2\mu\phi(N)e(N)$$

$$e(N) = y(N) - (\phi_1(N)\hat{\theta}_1(N-1) + \phi_2(N)\hat{\theta}_2(N-1))$$

$$\begin{bmatrix}\hat{\theta}_1(N)\\ \hat{\theta}_2(N)\end{bmatrix} = \begin{bmatrix}\hat{\theta}_1(N-1)\\ \hat{\theta}_2(N-1)\end{bmatrix} + 2\mu \begin{bmatrix}\phi_1(N)\\ \phi_2(N)\end{bmatrix} e(N)$$
$$= \begin{bmatrix}\hat{\theta}_1(N-1) + 2\mu\phi_1(N)e(N)\\ \hat{\theta}_2(N-1) + 2\mu\phi_2(N)e(N)\end{bmatrix}$$

Optimization

μ is a constant, 2μ is constant as well. Hence, we consider 2μ together as one floating point number to reduce multiplications. Additionally, $2\mu \cdot e(N)$ can be cached.

```
float err = out - (in1 * gain[0] + in2 * gain[1]);
float factor = 1e-18 * err;
gain[0] += in1 * factor;
gain[1] += in2 * factor;
```

RLS

$$\begin{aligned} e(N) &= y(N) - \phi(N)^T\hat{\theta}(N-1)\\ P(N) &= \frac{1}{\lambda}\Big(P(N-1) - \frac{P(N-1)\phi(N)\phi(N)^T P(N-1)}{\lambda + \phi(N)^T P(N-1)\phi(N)}\Big)\\ &= \frac{1}{\lambda}\Big(P(N-1) - \frac{NUM}{DEN}\Big)\\ \hat{\theta}(N) &= \hat{\theta}(N-1) + P(N)\phi(N)e(N) \end{aligned}$$

$$\phi = \begin{bmatrix}in1\\ in2\end{bmatrix} \qquad \hat{\theta} = \begin{bmatrix}gain0\\ gain1\end{bmatrix} \qquad P = \begin{bmatrix}P11 & P12\\ P21 & P22\end{bmatrix}$$

$$err = out - (in1 * gain0 + in2 * gain1)$$

$$\begin{aligned} DEN &= lambda + in1 * (P11 * in1 + P21 * in2) + in2 * (P12 * in1 + P22 * in2)\\ NUM11 &= P11 * in1 * (P11 * in1 + P12 * in2) + P21 * in2 * (P11 * in1 + P12 * in2)\\ NUM12 &= P12 * in1 * (P11 * in1 + P12 * in2) + P22 * in2 * (P11 * in1 + P12 * in2)\\ NUM21 &= P11 * in1 * (P21 * in1 + P22 * in2) + P21 * in2 * (P21 * in1 + P22 * in2)\\ NUM22 &= P12 * in1 * (P21 * in1 + P22 * in2) + P22 * in2 * (P21 * in1 + P22 * in2) \end{aligned}$$

$$\begin{bmatrix}P11 & P12\\ P21 & P22\end{bmatrix} = \frac{1}{lambda}\Big(\begin{bmatrix}P11 & P12\\ P21 & P22\end{bmatrix} - \frac{1}{DEN}\begin{bmatrix}NUM11 & NUM12\\ NUM21 & NUM22\end{bmatrix}\Big)$$

$$\begin{bmatrix}gain0\\ gain1\end{bmatrix} = \begin{bmatrix}gain0 + err * (P11 * in1 + P12 * in2)\\ gain1 + err * (P21 * in1 + P22 * in2)\end{bmatrix}$$

Most of the preceding calculations are done in MATLAB by creating symbolic variables. The code can be found in the appendix on page 27.

Optimization

RLS algorithm is much more time-consuming than LMS. We optimize in the following ways.

Firstly, we cache multiplications in `v1` to `v6`. Secondly, we precompute the reciprocal of λ in MATLAB because floating point multiplication is faster than division. The reciprocal of DEN is also precomputed and cached.

```
float lambda = 0.98;
float lambda_reciprocal = 1.020408163;

float v1 = P11*in1;
float v2 = P12*in1;
float v3 = P12*in2;
float v4 = P21*in1;
float v5 = P21*in2;
float v6 = P22*in2;

float den_reciprocal = 1 / ( lambda + in1*(v1 + v5) + in2*(v2 + v6) );

P11 = (P11 - (v1*(v1 + v3) + v5*(v1 + v3)) * den_reciprocal) * lambda_reciprocal;
P12 = (P12 - (v2*(v1 + v3) + v6*(v1 + v3)) * den_reciprocal) * lambda_reciprocal;
P21 = (P21 - (v1*(v4 + v6) + v5*(v4 + v6)) * den_reciprocal) * lambda_reciprocal;
P22 = (P22 - (v2*(v4 + v6) + v6*(v4 + v6)) * den_reciprocal) * lambda_reciprocal;

float err = out - in1*gain[0] - in2*gain[1];
gain[0] += err * (v1 + v3);
gain[1] += err * (v4 + v6);
```

3.

As is shown in Fig. 13, smaller μ results in longer transition time. Thus, we can make μ smaller to make it disappear after five seconds instead of two seconds.

In terms of RLS, $\lambda = 1$ means all past data points are taken into consideration. Smaller μ leads to faster reaction to amplitude variances. Hence, we can increase λ to approximate $\lambda = 1$.

According to the handout, μ can be estimated using the following guidance.

$$0 < \mu \ll \frac{1}{E[||u_1(t)||^2]}$$
$$0 < \mu \ll \frac{1}{E[||u_2(t)||^2]}$$

Task 2 Implementation

(c)

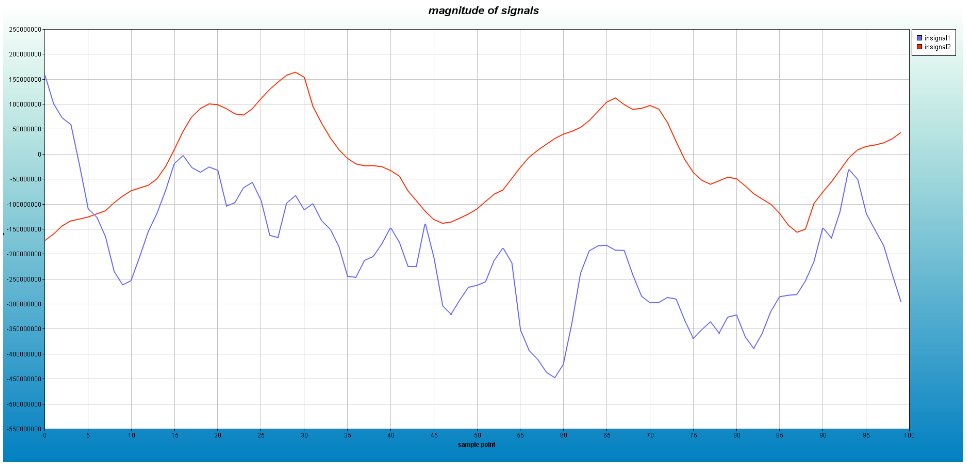

Figure 53: Input signal amplitudes

As we observed in the input signal plot, the average amplitude is around 10^8. Substitute this value into the guideline we obtained before, we can roughly derive a reasonable $\mu = 10^{-18} \ll \frac{1}{(10^8)^2}$.

(d)

If $\lambda = 1$, it takes longer for $u_2(t)$ to disappear.

If we choose μ too large, LMS filter goes unstable and produces loud and unpleasant output.

LMS

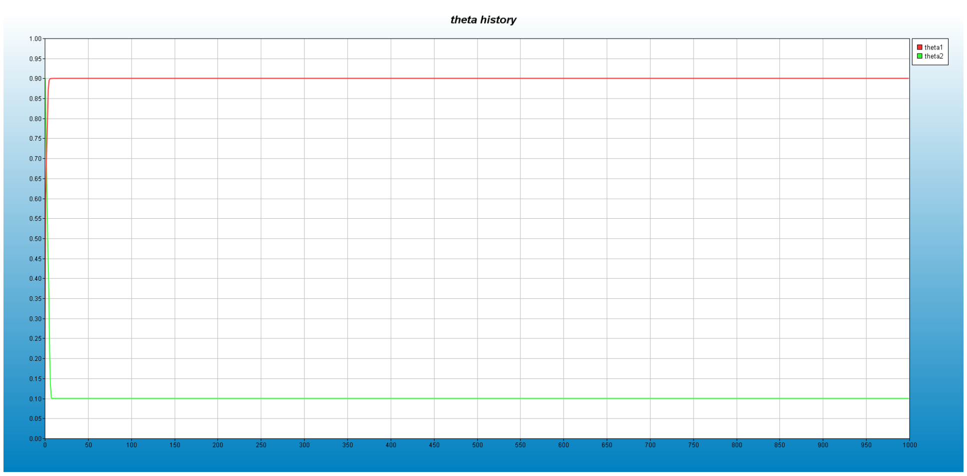

Figure 54: LMS `gain[0]` and `gain[1]`

RLS

Suggestion from the demonstrator

$$P(0) = \begin{bmatrix} 10^{-19} & 0 \\ 0 & 10^{-19} \end{bmatrix}$$
$$\lambda = 0.98$$

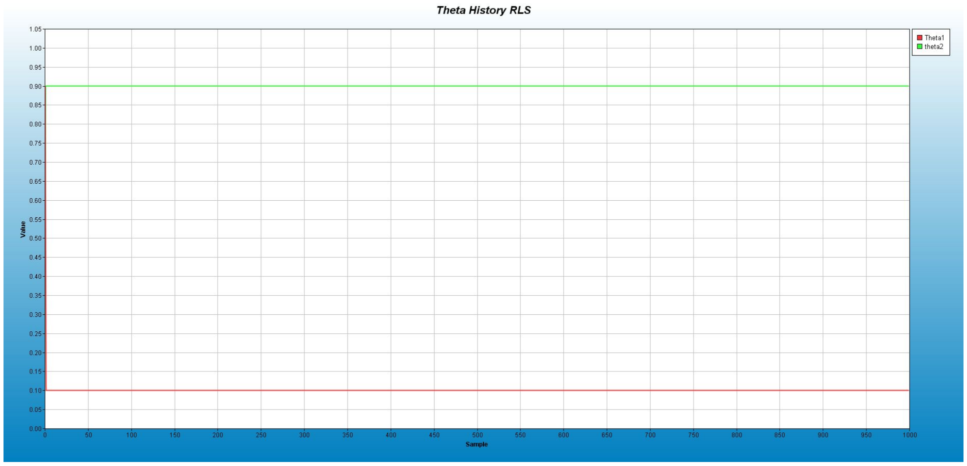

Figure 55: RLS `gain[0]` and `gain[1]`

Appendix

1 (a) & (b) & (c) RSL

```
clear;

load('data1');
load('data2');

question = 'a';
% 'a', 'b' or 'c'

is_noisy = false;
% true (noisy) or false (noise-free)

export_fig = false;
% print plots

% parameters
switch question
    case 'a'
        lambda = 0.999;
        rho = 1000;

        if (is_noisy)
            rho = 0.1;
        end

    case 'b'
        lambda = 0.999;
        rho = 1000;

        if (is_noisy)
            rho = 0.6;
        end

    case 'c'
        lambda = 0.999;
        rho = 100;

        if (is_noisy)
            rho = 0.001;
        end
end

% inputs
switch question
    case 'a'
        mike = mike1;
        if (is_noisy)
            mike = noisymike1;
        end
        side = 0;
        % known delay

    case 'b'
        mike = mike2;
        if (is_noisy)
            mike = noisymike2;
        end
        side = 0;
```

```
        % known delay

    case 'c'
        mike = mike2;
        if (is_noisy)
            mike = noisymike2;
        end
        side = 8;
        % 8192Hz * 0.001s = 8
end

P = rho * eye(2+4*side);

% the signal played by loudspeaker
u = loudspeaker;

noise_string = 'noise-free';
if (is_noisy)
    noise_string = 'noisy';
end

y = mike - loudspeaker;

% sampling frequency
fs = 8192;

% data length
L = length(u);

% echo cancellation result
err = zeros(L, 1);

% echo cancellation result
output = zeros(L, 1);

k1 = 1 * fs;
k2 = 2.5 * fs;

k1_first = k1 - side;
k1_last = k1 + side;

k2_first = k2 - side;
k2_last = k2 + side;

theta = zeros(2+4*side, 1);
weight1 = zeros(L, 1);
weight2 = zeros(L, 1);

tic;
for k = 1:L
    if k <= k1_first
        phi = [zeros(side*2+1, 1); zeros(side*2+1, 1)];
    elseif k <= k1_last
        phi = [zeros(k1_last-k+1, 1); u(1:k-k1_first); zeros(side*2+1, 1)];
    elseif k <= k2_first
        phi = [u(k-k1_last:k-k1_first); zeros(side*2+1, 1)];
    elseif k <= k2_last
        phi = [u(k-k1_last:k-k1_first); zeros(k2_last-k+1, 1); u(1:k-k2_first)];
    else
        phi = [u(k-k1_last:k-k1_first); u(k-k2_last:k-k2_first)];
    end
```

```
    P = (P - P * phi * transpose(phi) * P / ( lambda + transpose(phi) * P * phi )) / lambda;

    product = transpose(theta) * phi;
    err(k) = y(k) - product;
    output(k) = mike(k) - product;

    theta = theta + P * phi * err(k);

    weight1(k) = theta(side+1);
    weight2(k) = theta(side*2+1+side+1);
end
toc;

fprintf('lambda = %f\n', lambda);
fprintf('rho = %f\n', rho);
fprintf('norm(err) = %f\n', norm(err));

filename_prefix = [question '-rls-'];

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
plot(weight1);
hold on;
plot(weight2);
hold off;
title(['1(', question, ') RLS $\theta$', ' (', noise_string, ')'], 'interpreter', 'latex');
xlabel(['$\lambda$ = ' num2str(lambda) ' $\rho$ = ' num2str(rho)], 'interpreter', 'latex');
legend(['\theta_{' num2str(side+1) '}'], ['\theta_{' num2str(side*2+1+side+1) '}'], 'Location', 'southeast');
ylim([0 0.8]);
hold on;
plot(L, weight1(L), 'kx');
plot(L, weight2(L), 'kx');
text(0.9 * L, weight1(L) + 0.03, num2str(weight1(L)));
text(0.9 * L, weight2(L) + 0.03, num2str(weight2(L)));
if export_fig
    print([filename_prefix 'theta-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
subplot(2, 1, 1);
plot(loudspeaker);
title(['1(', question, ') \texttt{loudspeaker}'], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(mike);
if (is_noisy)
    title('\texttt{noisymike}', 'interpreter', 'latex');
else
    title('\texttt{mike}', 'interpreter', 'latex');
end
ylim([-1 1]);
if export_fig
    print([filename_prefix 'input-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
subplot(2, 1, 1);
plot(err);
title(['1(', question, ') \texttt{err}'], 'interpreter', 'latex');
```

```
xlabel(['\texttt{norm(err)} = ' sprintf('%f', norm(err))], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(output);
title('\texttt{output}', 'interpreter', 'latex');
ylim([-1 1]);
if export_fig
    print([filename_prefix 'output-', noise_string], '-dpng', '-r300');
end

if export_fig
    close all;
end
```

1 (a) & (b) & (c) LMS

```
clear;

load('data1');
load('data2');

question = 'a';
% 'a', 'b' or 'c'

is_noisy = false;
% true (noisy) or false (noise-free)

export_fig = false;
% print plots

% parameters
switch question
    case 'a'
        step_size = 10;

        if (is_noisy)
            step_size = 0.5;
        end

    case 'b'
        step_size = 10;

        if (is_noisy)
            step_size = 0.6;
        end

    case 'c'
        step_size = 0.3;

        if (is_noisy)
            step_size = 0.1;
        end
end

% inputs
switch question
    case 'a'
        mike = mike1;
        if (is_noisy)
            mike = noisymike1;
        end
        side = 0;
```

```
        % known delay

    case 'b'
        mike = mike2;
        if (is_noisy)
            mike = noisymike2;
        end
        side = 0;
        % known delay

    case 'c'
        mike = mike2;
        if (is_noisy)
            mike = noisymike2;
        end
        side = 8;
        % 8192Hz * 0.001s = 8
end

% the signal played by loudspeaker
u = loudspeaker;

noise_string = 'noise-free';
if (is_noisy)
    noise_string = 'noisy';
end

y = mike - loudspeaker;

% sampling frequency
fs = 8192;

% data length
L = length(u);

% echo cancellation result
err = zeros(L, 1);

% echo cancellation result
output = zeros(L, 1);

k1 = 1 * fs;
k2 = 2.5 * fs;

k1_first = k1 - side;
k1_last = k1 + side;

k2_first = k2 - side;
k2_last = k2 + side;

theta = zeros(2+4*side, 1);
weight1 = zeros(L, 1);
weight2 = zeros(L, 1);

tic;
for k = 1:L
    if k <= k1_first
        phi = [zeros(side*2+1, 1); zeros(side*2+1, 1)];
    elseif k <= k1_last
        phi = [zeros(k1_last-k+1, 1); u(1:k-k1_first); zeros(side*2+1, 1)];
    elseif k <= k2_first
        phi = [u(k-k1_last:k-k1_first); zeros(side*2+1, 1)];
```

```
    elseif k <= k2_last
        phi = [u(k-k1_last:k-k1_first); zeros(k2_last-k+1, 1); u(1:k-k2_first)];
    else
        phi = [u(k-k1_last:k-k1_first); u(k-k2_last:k-k2_first)];
    end

    product = transpose(theta) * phi;
    err(k) = y(k) - product;
    output(k) = mike(k) - product;

    theta = theta + step_size * phi * err(k);

    weight1(k) = theta(side+1);
    weight2(k) = theta(side*2+1+side+1);
end
toc;

fprintf('step_size = %f\n', step_size);
fprintf('norm(err) = %f\n', norm(err));

filename_prefix = [question '-lms-'];

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
plot(weight1);
hold on;
plot(weight2);
hold off;
title(['1(', question, ') LMS $\theta$', ' (', noise_string, ')'], 'interpreter', 'latex');
xlabel(['\texttt{step\_size} = $2\mu$ = ' num2str(step_size)], 'interpreter', 'latex');
legend(['\theta_{' num2str(side+1) '}'], ['\theta_{' num2str(side*2+1+side+1) '}'], 'Location', '
    southeast');
ylim([0 0.8]);
hold on;
plot(L, weight1(L), 'kx');
plot(L, weight2(L), 'kx');
text(0.9 * L, weight1(L) + 0.03, num2str(weight1(L)));
text(0.9 * L, weight2(L) + 0.03, num2str(weight2(L)));
if export_fig
    print([filename_prefix 'theta-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
subplot(2, 1, 1);
plot(loudspeaker);
title(['1(', question, ') \texttt{loudspeaker}'], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(mike);
if (is_noisy)
    title('\texttt{noisymike}', 'interpreter', 'latex');
else
    title('\texttt{mike}', 'interpreter', 'latex');
end
ylim([-1 1]);
if export_fig
    print([filename_prefix 'input-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
```

```
subplot(2, 1, 1);
plot(err);
title(['1(', question, ') \texttt{err}'], 'interpreter', 'latex');
xlabel(['\texttt{norm(err)} = ' sprintf('%f', norm(err))], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(output);
title('\texttt{output}', 'interpreter', 'latex');
ylim([-1 1]);
if export_fig
    print([filename_prefix 'output-', noise_string], '-dpng', '-r300');
end

if export_fig
    close all;
end
```

1 (d) RSL

```
clear;

load('data1');

is_noisy = false;
% true (noisy) or false (noise-free)

export_fig = false;
% print plots

% parameters
lambda = 1;
rho = 3;

P = rho * eye(2);

% inputs
mike = mike1;
if (is_noisy)
    mike = noisymike1;
end

noise_string = 'noise-free';
if (is_noisy)
    noise_string = 'noisy';
end

% sampling frequency
fs = 8192;

% data length
L = length(mike);

% echo cancellation result
output = zeros(L, 1);

% echo cancellation result
output = zeros(L, 1);

k1 = 1 * fs;
k2 = 2.5 * fs;

theta = zeros(2, 1);
```

```
weight1 = zeros(L, 1);
weight2 = zeros(L, 1);

tic;
for k = 1:L
    if k <= k1
        phi = [0; 0];
    elseif k <= k2
        phi = [output(k-k1); 0];
    else
        phi = [output(k-k1); output(k-k2)];
    end

    P = (P - P * phi * transpose(phi) * P / ( lambda + transpose(phi) * P * phi )) / lambda;

    err = mike(k) - transpose(theta) * phi;

    theta = theta + P * phi * err;

    output(k) = mike(k) - transpose(theta) * phi;

    weight1(k) = theta(1);
    weight2(k) = theta(2);
end
toc;

fprintf('lambda = %f\n', lambda);
fprintf('rho = %f\n', rho);
fprintf('norm(err) = %f\n', norm(output - loudspeaker));

filename_prefix = 'd-rls-';

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
plot(weight1);
hold on;
plot(weight2);
hold off;
title(['1(d) RLS $\theta$', ' (', noise_string, ')'], 'interpreter', 'latex');
legend(['\theta_{1}'], ['\theta_{2}'], 'Location', 'southeast');
ylim([0 0.8]);
hold on;
plot(L, weight1(L), 'kx');
plot(L, weight2(L), 'kx');
text(0.9 * L, weight1(L) + 0.03, num2str(weight1(L)));
text(0.9 * L, weight2(L) + 0.03, num2str(weight2(L)));
if export_fig
    print([filename_prefix 'theta-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
subplot(2, 1, 1);
plot(loudspeaker);
title(['1(d) \texttt{loudspeaker}'], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(mike);
if (is_noisy)
    title('\texttt{noisymike1}', 'interpreter', 'latex');
else
    title('\texttt{mike1}', 'interpreter', 'latex');
```

```
end
ylim([-1 1]);
if export_fig
    print([filename_prefix 'input-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
subplot(2, 1, 1);
plot(output);
title(['1(d) \texttt{output}'], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(output - loudspeaker);
title('\texttt{output - loudspeaker}', 'interpreter', 'latex');
ylim([-1 1]);
if export_fig
    print([filename_prefix 'output-', noise_string], '-dpng', '-r300');
end

if export_fig
    close all;
end
```

1 (d) LMS

```
clear;

load('data1');

is_noisy = false;
% true (noisy) or false (noise-free)

export_fig = false;
% print plots

% parameters
step_size = 0.015;

% inputs
mike = mike1;
if (is_noisy)
    mike = noisymike1;
end

noise_string = 'noise-free';
if (is_noisy)
    noise_string = 'noisy';
end

% sampling frequency
fs = 8192;

% data length
L = length(mike);

% echo cancellation result
output = zeros(L, 1);

% echo cancellation result
output = zeros(L, 1);

```

```
k1 = 1 * fs;
k2 = 2.5 * fs;

theta = zeros(2, 1);
weight1 = zeros(L, 1);
weight2 = zeros(L, 1);

tic;
for k = 1:L
    if k <= k1
        phi = [0; 0];
    elseif k <= k2
        phi = [output(k-k1); 0];
    else
        phi = [output(k-k1); output(k-k2)];
    end

    err = mike(k) - transpose(theta) * phi;

    theta = theta + step_size * phi * err;

    output(k) = mike(k) - transpose(theta) * phi;

    weight1(k) = theta(1);
    weight2(k) = theta(2);
end
toc;

fprintf('step_size = %f\n', step_size);
fprintf('norm(err) = %f\n', norm(output - loudspeaker));

filename_prefix = 'd-lms-';

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
plot(weight1);
hold on;
plot(weight2);
hold off;
title(['1(d) LMS $\theta$', ' (', noise_string, ')'], 'interpreter', 'latex');
legend(['\theta_{1}'], ['\theta_{2}'], 'Location', 'southeast');
ylim([0 0.8]);
hold on;
plot(L, weight1(L), 'kx');
plot(L, weight2(L), 'kx');
text(0.9 * L, weight1(L) + 0.03, num2str(weight1(L)));
text(0.9 * L, weight2(L) + 0.03, num2str(weight2(L)));
if export_fig
    print([filename_prefix 'theta-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
subplot(2, 1, 1);
plot(loudspeaker);
title(['1(d) \texttt{loudspeaker}'], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(mike);
if (is_noisy)
    title('\texttt{noisymike1}', 'interpreter', 'latex');
else
```

```
    title('\texttt{mike1}', 'interpreter', 'latex');
end
ylim([-1 1]);
if export_fig
    print([filename_prefix 'input-', noise_string], '-dpng', '-r300');
end

fig = figure;
fig.PaperPosition = [0 0 5 3.75];
subplot(2, 1, 1);
plot(output);
title(['1(d) \texttt{output}'], 'interpreter', 'latex');
ylim([-1 1]);
subplot(2, 1, 2);
plot(output - loudspeaker);
title('\texttt{output - loudspeaker}', 'interpreter', 'latex');
ylim([-1 1]);
if export_fig
    print([filename_prefix 'output-', noise_string], '-dpng', '-r300');
end

if export_fig
    close all;
end
```

gainestimate.c

```
#include "SP2WS1.h"
#include <time.h>


// input signal history
float insignal1[100], insignal2[100];


// function prototypes
void gainestimateLMS(float,float,float,float[2]);
void gainestimateRLS(float,float,float,float[2]);


void gainestimate(float in1, float in2, float out, float gain[2])
{
    // record input signal history for checking signal magnitude using plot facility
    for (int i = 99; i > 0; i--)
    {
        insignal1[i] = insignal1[i-1];
        insignal2[i] = insignal2[i-1];
    }
    insignal1[0] = in1;
    insignal2[0] = in2;


    // estimate gain
    // gain[0] = 0;
    // gain[1] = 1;
    gainestimateLMS(in1, in2, out, gain);
    //gainestimateRLS(in1, in2, out, gain);
}

void gainestimateLMS(float in1, float in2, float out, float gain[2])
{
    // TODO: Implement gain estimation using LMS algorithm
```

```
    float err = out - (in1 * gain[0] + in2 * gain[1]);
    float factor = 1e-18 * err;
    gain[0] += in1 * factor;
    gain[1] += in2 * factor;
}

void gainestimateRLS(float in1, float in2, float out, float gain[2])
{
    // TODO: Implement gain estimation using RLS algorithm
    static float P11 = 1e-19;
    static float P12 = 0.0;
    static float P21 = 0.0;
    static float P22 = 1e-19;

    float lambda = 0.98;
    float lambda_reciprocal = 1.020408163;

    static int index = 0;
    long clocks = clock();

    float v1 = P11*in1;
    float v2 = P12*in1;
    float v3 = P12*in2;
    float v4 = P21*in1;
    float v5 = P21*in2;
    float v6 = P22*in2;

    float den_reciprocal = 1 / ( lambda + in1*(v1 + v5) + in2*(v2 + v6) );

    P11 = (P11 - (v1*(v1 + v3) + v5*(v1 + v3)) * den_reciprocal) * lambda_reciprocal;
    P12 = (P12 - (v2*(v1 + v3) + v6*(v1 + v3)) * den_reciprocal) * lambda_reciprocal;
    P21 = (P21 - (v1*(v4 + v6) + v5*(v4 + v6)) * den_reciprocal) * lambda_reciprocal;
    P22 = (P22 - (v2*(v4 + v6) + v6*(v4 + v6)) * den_reciprocal) * lambda_reciprocal;

    float err = out - in1*gain[0] - in2*gain[1];
    gain[0] += err * (v1 + v3);
    gain[1] += err * (v4 + v6);

    clocks = clock() - clocks;

    if (!index) {
        printf("Clocks: %d\n", clocks);
        printf("CLOCKS_PER_SEC: %d\n", CLOCKS_PER_SEC);
    }

    index++;
    index %= 48000;
}
```

RLS calculations in MATLAB by creating symbolic variables

```
clear;
clc;

syms in1 in2;
syms out;

syms P11 P12 P21 P22;
syms lambda;

P = [P11, P12; P21, P22];
phi = [in1; in2];
```

```
den = (lambda + transpose(phi) * P * phi)

syms den_sym;
num = P * phi * transpose(phi) * P
P = (P - num/den_sym) / lambda

syms gain0 gain1;
syms err;

P_sym = [P11, P12; P21, P22];
gain = [gain0; gain1];
gain = gain + P_sym * phi * err
```