Adaptive Control

Assignment 1

By

Mahmoud Nasser

Master of Power Electronics

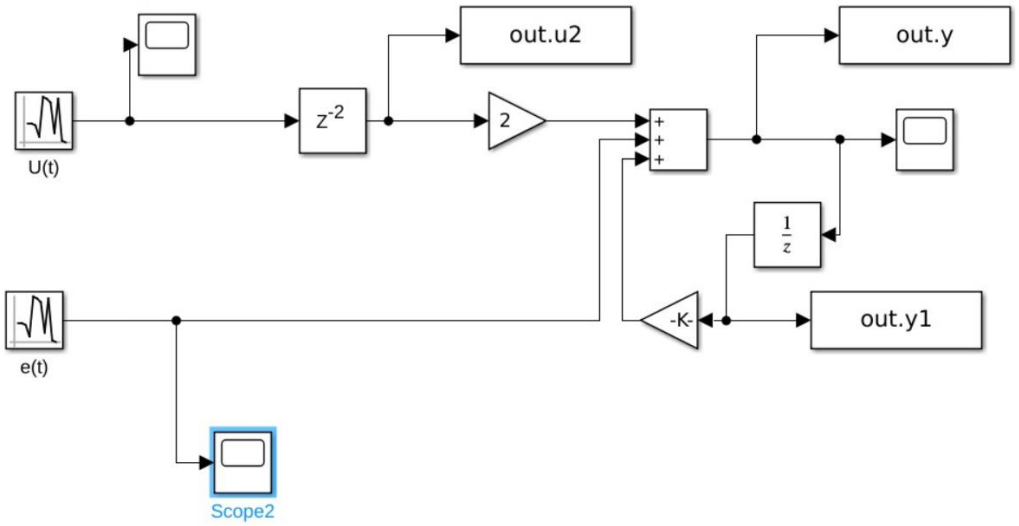
Department of Electric Power Engineering

Cairo University

2023

Submitted to : Dr. Abdellatif Elshaefi

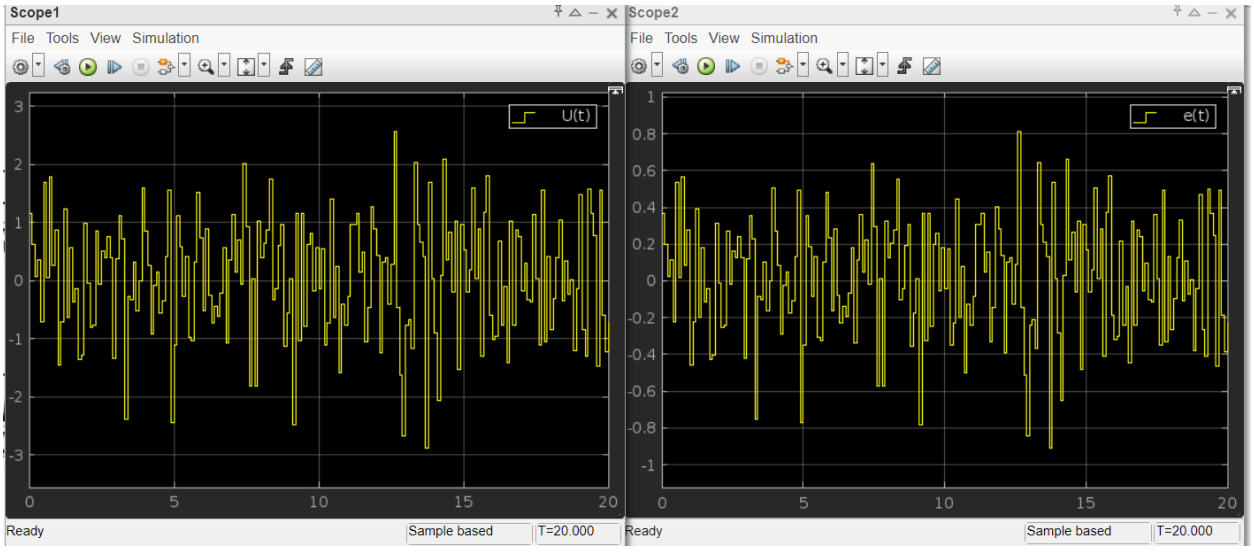
Problem 1 solution:



Simulink model

```
% Taking inputs from the Simulink model
y_t = out.y;
y_t_1 = out.y1;
u_t_2 = out.u2;

t = length(y_t);

% Form the phi matrix for least squares estimation
phi = [-y_t_1(3:end), u_t_2(3:end)];

% Estimation of Parameters
Theta = inv(phi'*phi)*phi'*y_t(3:end);

% Calculation of Residual Error
E = y_t(3:end) - phi*Theta;

% Variance and Standard Deviation Calculation
Variance = (E'*E)/(t-2);
Standard_deviation = sqrt(Variance);

% 95% Confidence Intervals
a_min = Theta(1) - 2*Standard_deviation;
a_max = Theta(1) + 2*Standard_deviation;
b_min = Theta(2) - 2*Standard_deviation;
b_max = Theta(2) + 2*Standard_deviation;

% Display the Results
fprintf("Estimated a: %f (95%% CI: %f to %f)\n", Theta(1), a_min, a_max);
fprintf("Estimated b: %f (95%% CI: %f to %f)\n", Theta(2), b_min, b_max);

% Plot the actual vs. estimated values
figure;
subplot(2,1,1);
plot(1:t, y_t, 'b', 'LineWidth', 2);
title('Actual Output');
xlabel('Time');
ylabel('Output');

subplot(2,1,2);
plot([1,2], [Theta(1), Theta(2)], 'rx', 'MarkerSize', 10, 'LineWidth', 2);
title('Estimated Parameters');
xlabel('Parameters');
ylabel('Value');
xticks([1,2]);
xticklabels({'a', 'b'});
legend('Estimated');
```

Matlab script

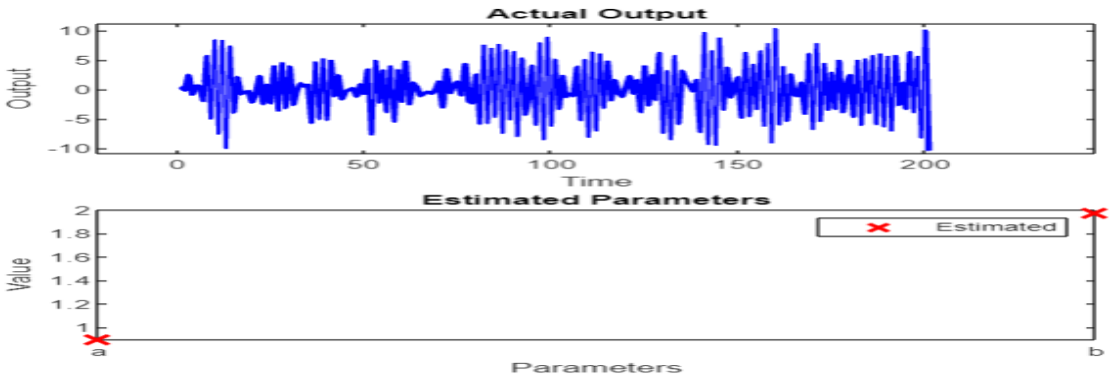


Input signal and error signal

Results:

```
Estimated a: 0.897658 (95% CI: 0.264036 to 1.531281)
Estimated b: 1.975823 (95% CI: 1.342201 to 2.609446)
```

Confidence interval for a and b



Actual output and estimated parameters

Comment on results:

For estimated a:

- **Estimated Value:** 0.897658
- **95% Confidence Interval (CI):** 0.264036 to 1.531281
- The 95% confidence interval suggests that we can be 95% confident that the true value of **a** lies somewhere between 0.264036 and 1.531281.

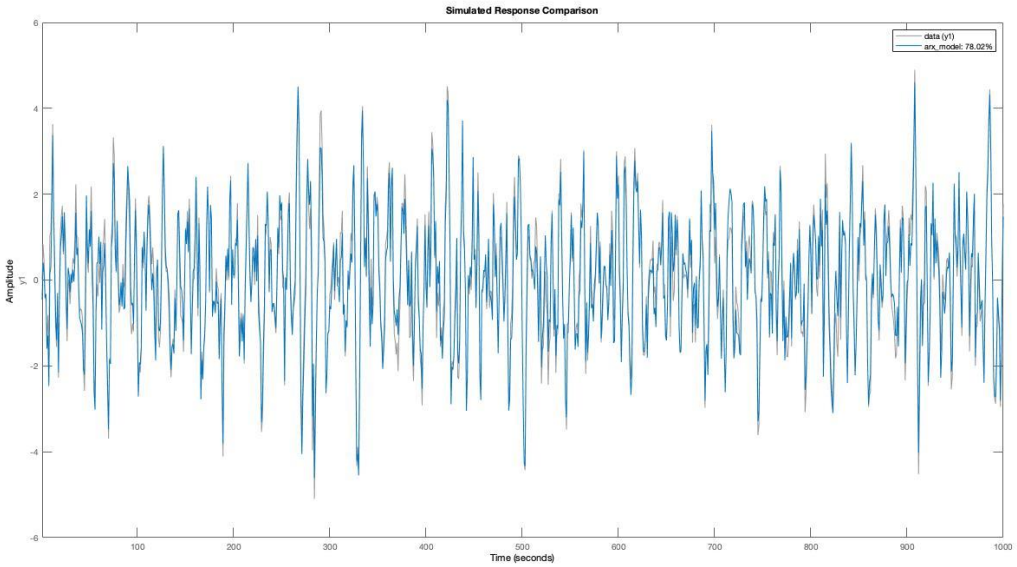
For estimated b:

Estimated b:

- **Estimated Value:** 1.975823
- **95% Confidence Interval (CI):** 1.342201 to 2.609446
- The 95% confidence interval suggests that we can be 95% confident that the true value of **b** lies somewhere between 1.342201 and 2.609446.

Problem 2 solution:

Results:



Comment on results:

I tried different orders for the polynomial function and the simplest and nearest order was second order and I got 78.02%

Problem 3 solution:

```
% Initialization
N = 500; % number of samples
a_true = 0.5;
b_true = 0.8;
y = zeros(N, 1);
e = sqrt(0.1) * randn(N, 1); % white noise with zero mean and 0.1 variance

for scenario = 1:5
    % Reinitialize for each case
    P = 10^10 * eye(2); % Initialize covariance matrix
    theta = [0; 1]; % Initialize parameter vector [a; b]
    a_est = zeros(N, 1);
    b_est = zeros(N, 1);
    y(1) = 0;
    u = zeros(N, 1);

    switch scenario
        case 1
            % u(t) = 0 is already initialized
        case 2
            %u = ones(N, 1);
            u = 0.5 * square(2*pi*(1:N)/25) + 0.5;
        case 3
            u = randn(N, 1);
        % Cases 4 and 5 will have u(t) updated inside the loop
    end

    % RLS Implementation
    for t = 2:N
        % For case 4: u(t) = 0.1y(t)
        if scenario == 4
            u(t) = 0.1 * y(t-1);
        end
```

```
        % For case 5: u(t) = 0.1*sign(y(t))
        if scenario == 5
            u(t) = 0.1 * sign(y(t-1));
        end

        % Vector phi represents the regressors
        phi = [y(t-1); u(t-1)];
        K = (P*phi) / (1 + phi'*P*phi);
        y(t) = a_true*y(t-1) + b_true*u(t-1) + e(t);
        e_est = y(t) - theta'*phi;
        theta = theta + K*e_est;
        P = P - K*phi'*P;
        a_est(t) = theta(1);
        b_est(t) = theta(2);
    end

    % Plotting
    figure(scenario);
    subplot(4,1,1);
    plot(y);
    title(['Output y(t) for Case ', num2str(scenario)]);

    subplot(4,1,2);
    plot(u);
    title(['Input u(t) for Case ', num2str(scenario)]);

    subplot(4,1,3);
    plot(a_est);
    hold on;
    plot(repmat(a_true, N, 1), 'r--');
    legend('Estimated a', ' a actual');
    title(['Estimated Parameters for Case ', num2str(scenario)]);


    subplot(4,1,4);
    plot(b_est);
    hold on;
    plot(repmat(b_true, N, 1), 'r--');
```

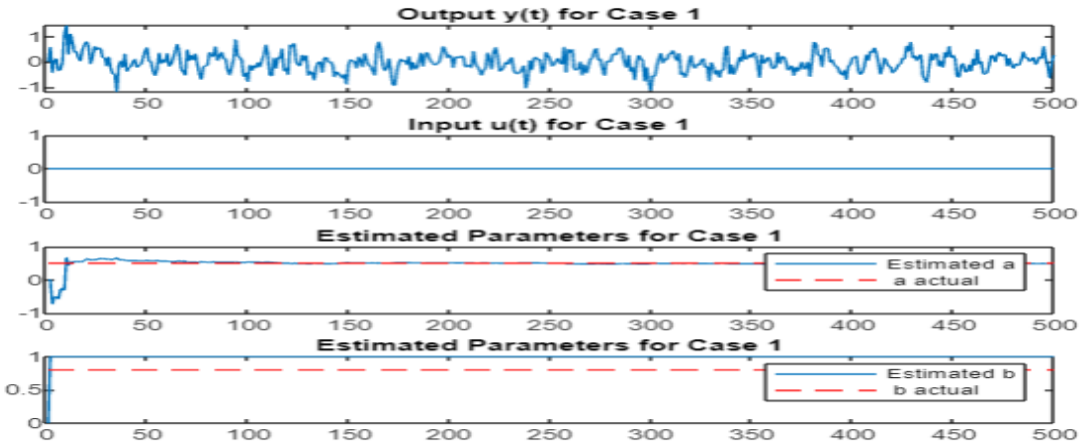
```
    % Plotting
    figure(scenario);
    subplot(4,1,1);
    plot(y);
    title(['Output y(t) for Case ', num2str(scenario)]);

    subplot(4,1,2);
    plot(u);
    title(['Input u(t) for Case ', num2str(scenario)]);

    subplot(4,1,3);
    plot(a_est);
    hold on;
    plot(repmat(a_true, N, 1), 'r--');
    legend('Estimated a', ' a actual');
    title(['Estimated Parameters for Case ', num2str(scenario)]);


    subplot(4,1,4);
    plot(b_est);
    hold on;
    plot(repmat(b_true, N, 1), 'r--');
    legend('Estimated b', ' b actual');
    title(['Estimated Parameters for Case ', num2str(scenario)]);


end
```

Matlab script for problem3

Results:

1- Case 1:

input u(t) = 0

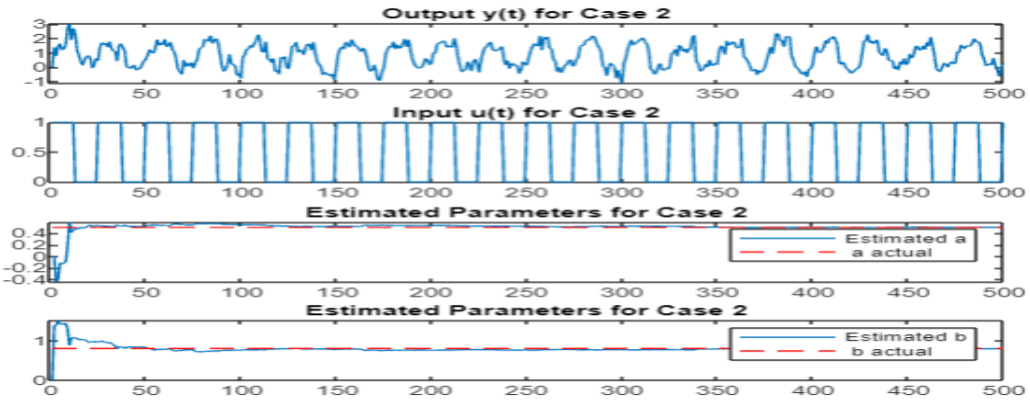


Output, input and estimated parameters

Comments on case1:

1. Convergence of Parameter *a*:
   - The estimate for the parameter *a* appears to converge rapidly to its true value of 0.5. After an initial period of adjustment, the estimate settles close to this actual value and remains stable for most of the time.
2. Convergence of Parameter *b*:
   - The estimate for parameter *b* does not show a significant change from its initialized value, staying close to 1. This is quite distant from its actual value of 0.8.

2- Case 2:
Square wave input with amplitude equals 1.



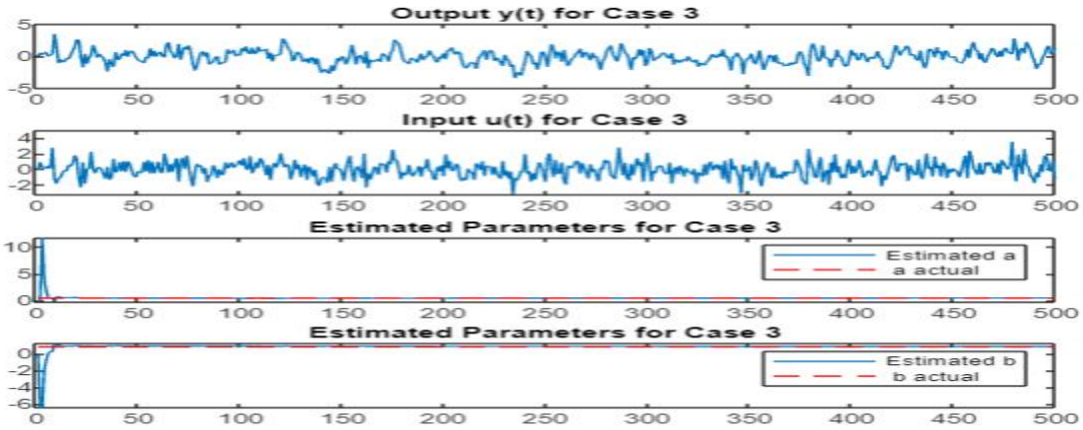
Comments on case 2:

1. Convergence of Parameter *a*:

- The estimate for the parameter *a* does converge to its true value of 0.5. After an initial transient period, the estimate settles around this actual value and remains stable.

2. Convergence of Parameter *b*:
   - The estimate for parameter *b* also converges towards its true value of 0.8. There's a rapid adjustment at the beginning, after which the estimate stabilizes and remains close to the true value for the rest of the duration.

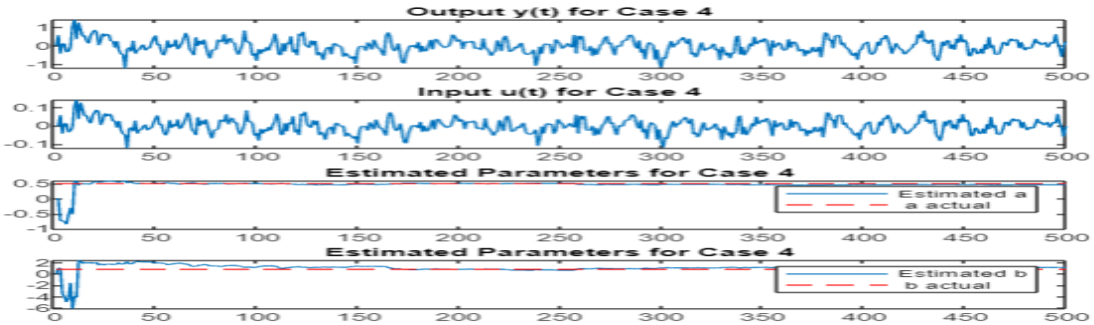
3- Case 3:
Input is Gaussian white noise with zero mean.



Comments on case 3:

1. Convergence of Parameter *a*:
   - The estimate for parameter *a* does not seem to converge precisely to the true value. Instead, it fluctuates over the entire duration, showing significant uncertainty and variation.
2. Convergence of Parameter *b*:
   - Similarly, the estimate for parameter *b* fluctuates significantly throughout and does not converge to its actual value.

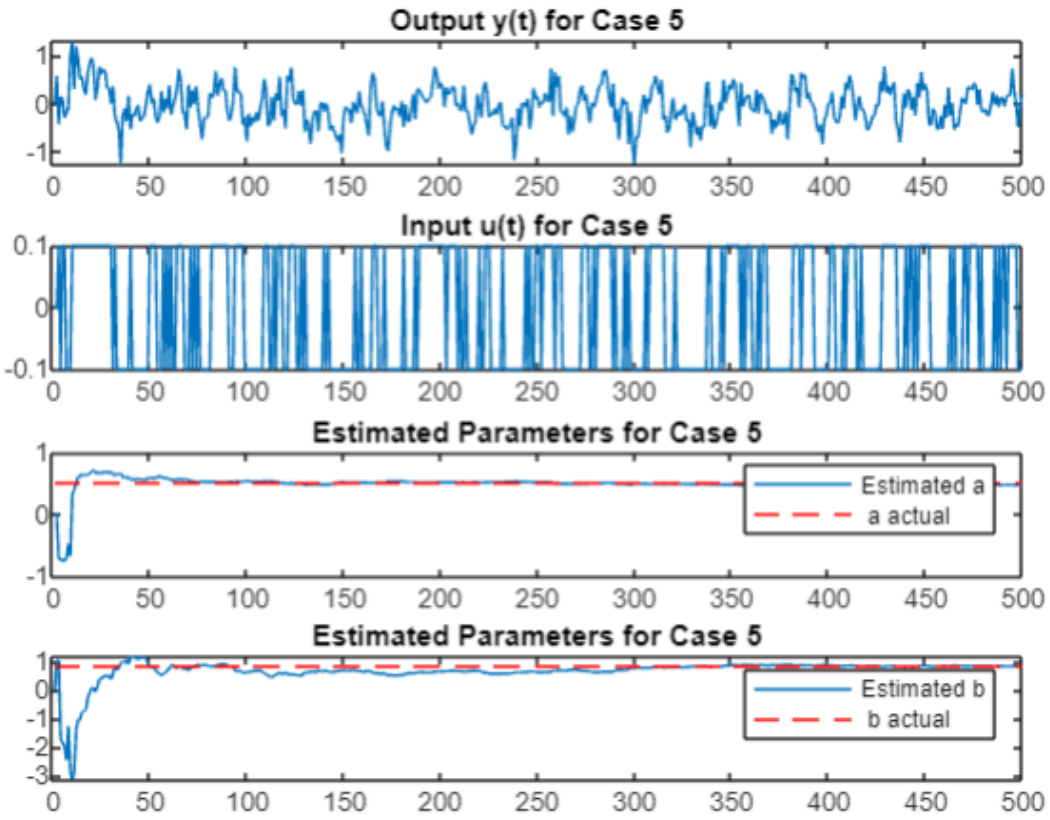
4- Case 4 :
u(t) = 0.1y(t)



Comments on case4:

1. Convergence of Parameter a:
   - The estimate for parameter a start with a deviation from its true value but, over time, it appears to stabilize and get closer to the true value. The estimate oscillates around the true value with minor deviations.
2. Convergence of Parameter b:
   - The estimate for parameter b also begins with some deviations. However, unlike a, it converges more quickly to its actual value and remains relatively steady, with only slight oscillations around the true value.

5- Case 5:
u(t) = 0.1sign(y(t))



Comments on case5:

1. Convergence of Parameter *a*:
    - The estimated parameter *a* starts off with a deviation from the actual value. As time progresses, the estimate seems to stabilize and come closer to the actual value, albeit with a consistent offset throughout.
2. Convergence of Parameter *b*:
    - Parameter *b*'s estimate starts with a considerable deviation. Over time, it begins to converge towards the true value but stabilizes at a value slightly away from the actual value.

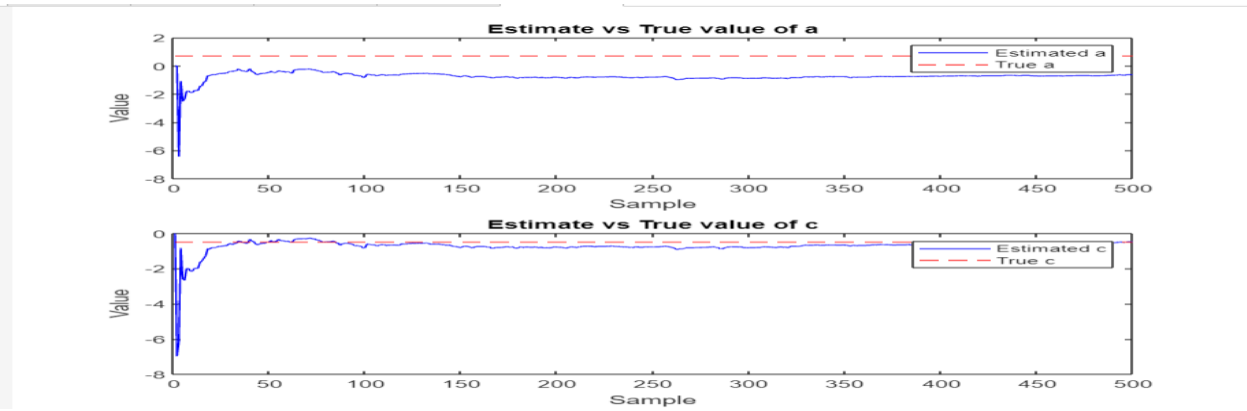
Problem 4 solution:

1- Case 1 RLS):

```
 if x == 1
a = 0.7; c = -0.5;
e = randn(N, 1); % White noise signal
y = zeros(N, 1);

for i = 2:N
    y(i) = a * y(i - 1) + c * e(i - 1) + e(i);
end

% RLS Initialization
P = 10*eye(2); % Covariance matrix
a_hat = 0; c_hat = 0; % Initial estimates

a_estimates = zeros(N, 1);
c_estimates = zeros(N, 1);

for i = 2:N
    phi = [-y(i-1) e(i-1)]'; % Regressor vector
    k = P * phi / (1 + phi' * P * phi);
    e_estimate = y(i) - [a_hat c_hat] * phi;

    a_hat = a_hat + k(1) * e_estimate;
    c_hat = c_hat + k(2) * e_estimate;

    P = P - k * phi' * P;

    a_estimates(i) = a_hat;
    c_estimates(i) = c_hat;
end

% Plotting
figure;
subplot(2,1,1);
plot(a_estimates, 'b'); hold on;
plot(repmat(a, N, 1), 'r--'); % True value of a
legend('Estimated a', 'True a');
xlabel('Sample'); ylabel('Value');
title('Estimate vs True value of a');

subplot(2,1,2);
plot(c_estimates, 'b'); hold on;
plot(repmat(c, N, 1), 'r--'); % True value of c
legend('Estimated c', 'True c');
xlabel('Sample'); ylabel('Value');
title('Estimate vs True value of c');
```

Matlab script for RLS algorithm

Results:



Comments on results:

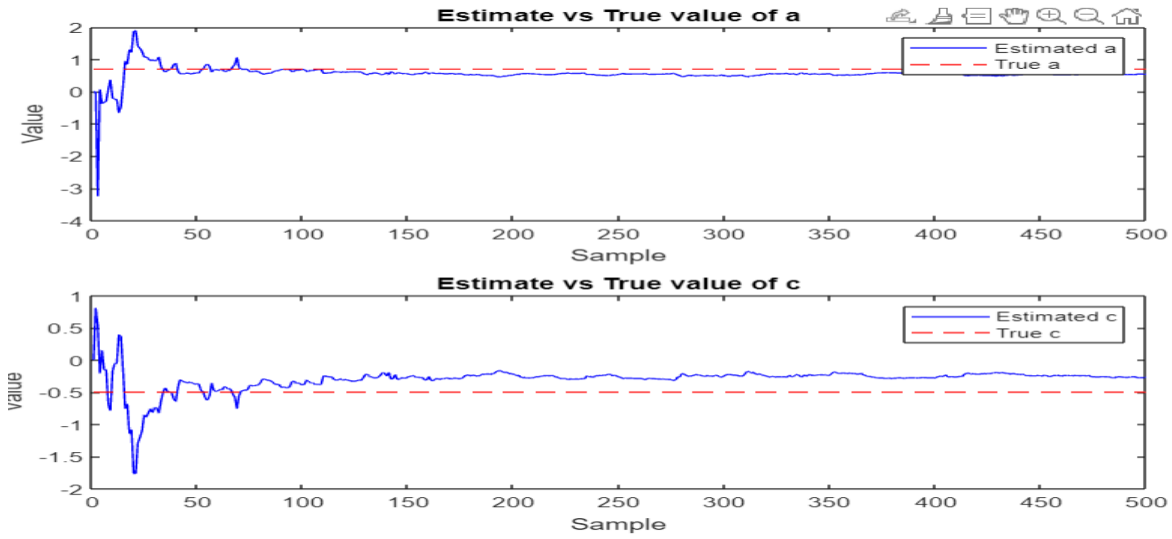
1. Convergence Behavior:
   - For Parameter 'a': The estimate seems to have some initial oscillations or transient behavior but stabilizes after around 250 samples. After stabilization, the estimated 'a' stays relatively close to the true value.
   - For Parameter 'c': The behavior is similar, with initial fluctuations followed by a stabilization after approximately 250 samples. The estimate then follows the true value with minor deviations.

Case2: ERLS

```
N = 1000; % Number of samples
a = 0.7; c = -0.5;
e = randn(N, 1); % white noise signal
y = zeros(N, 1);

% System simulation
for i = 2:N
    y(i) = a * y(i - 1) + c * e(i - 1) + e(i);
end

% ERLS Initialization
P = 1000 * eye(2); % Covariance matrix initialization
theta_hat = [0; 0]; % Initial parameter estimates

a_estimates = zeros(N, 1);
c_estimates = zeros(N, 1);

for i = 2:N
    phi = [y(i-1); e(i-1)]; % Regression vector

    K = (P * phi) / (1 + phi' * P * phi); % Kalman gain
    e_estimate = y(i) - phi' * theta_hat;
    theta_hat = theta_hat + K * e_estimate; % Parameter update
    P = P - K * phi' * P; % Covariance matrix update

    a_estimates(i) = theta_hat(1);
    c_estimates(i) = theta_hat(2);
end

% Plotting results
figure;
subplot(2,1,1);
plot(a_estimates, 'b'); hold on;
plot(repmat(a, N, 1), 'r--');
legend('Estimated a', 'True a');
xlabel('Sample'); ylabel('Value');
title('Estimate vs True value of a');

subplot(2,1,2);
plot(c_estimates, 'b'); hold on;
plot(repmat(c, N, 1), 'r--');
legend('Estimated c', 'True c');
xlabel('Sample'); ylabel('Value');
title('Estimate vs True value of c');
end
```

Matlab script for ERLS

Results:



Comments on case 2:

1. Convergence Behavior:
    - For Parameter 'a': After some initial oscillations, the estimated 'a' gradually converges towards the true value, though with some deviations. The estimate appears to stabilize after around 250 samples but continues to fluctuate around the true value.
    - For Parameter 'c': The estimated 'c' also experiences initial fluctuations, but it stabilizes much quicker, close to the true value. The stabilized estimate seems to track the true value consistently, with minor deviations.

Case 3 :RAML

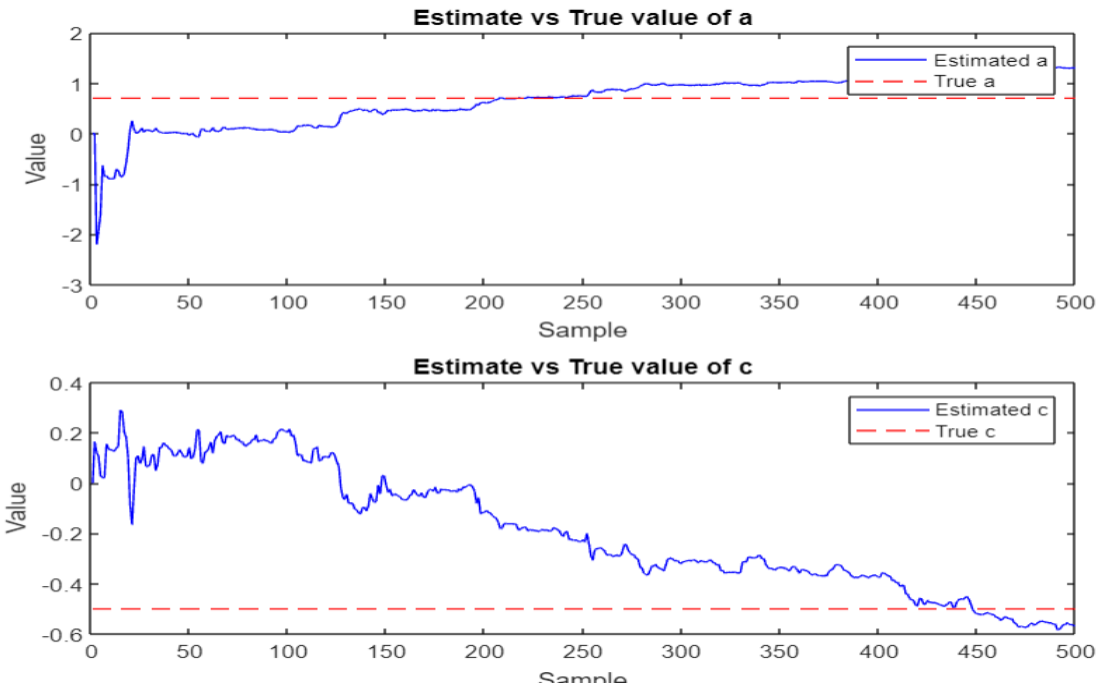
ive/wwww.m

```
% Sample Data Initialization
N = 1000; % Number of samples
true_a = 0.7;
true_c = -0.5;
e = randn(N, 1); % White noise with zero mean and unit variance
y = zeros(N, 1);

% System simulation
for t = 2:N
    y(t) = true_a * y(t-1) + true_c * e(t-1) + e(t);
end

% Initialization
phi = zeros(2, N); % [y(t-1), e(t-1)]
theta = zeros(2, N); % [a_hat; c_hat]
P =  10*eye(2); % Initial covariance matrix

% Iteration for RAML
for t = 2:N
    phi(:, t) = [y(t-1); e(t-1)];

    % Compute prediction errors
    e_p(t) = y(t) - phi(:, t-1).' * theta(:, t);

    % RAML Update rule for theta
    K = P * phi(:, t) / (1 + phi(:, t).' * P * phi(:, t));
    theta(:, t) = theta(:, t-1) + K * e_p(t);
    P = P - K * phi(:, t).' * P;
end

% Plotting the results
figure;
subplot(2,1,1);
plot(theta(1, :), 'b'); hold on;
plot(true_a * ones(1, N), 'r--');
xlabel('Sample'); ylabel('Value'); title('Estimate vs True value of a');
legend('Estimated a', 'True a');

subplot(2,1,2);
plot(theta(2, :), 'b'); hold on;
plot(true_c * ones(1, N), 'r--');
xlabel('Sample'); ylabel('Value'); title('Estimate vs True value of c');
legend('Estimated c', 'True c');

% Plot y and e for reference
figure;
```

Matlab script for RAML

Results:

Figure 1 × Figure 2 × +



Comments on case 3:

1. Convergence Behavior:
   - For Parameter 'a': The estimated 'a' experiences initial oscillations and then stabilizes towards the true value. After about 150 samples, it shows consistency with the true value, maintaining a steady behavior.
   - For Parameter 'c': The estimated 'c' showcases pronounced oscillations at the beginning, but it starts to settle as more samples are processed. However, there are still some minor fluctuations observed, even past 400 samples.