ME 6404 Final Project Report

Xinyi Cai, Praveen Ravishankar, Donghoon Yang (Team 4)

11/23/2020

Abstract

There were numerous control methods learned in this course. While some of the methods learned were applied in the labs and actually implemented on a physical system, other concepts were learned only theoretically. As a result, the team decided to further implement some of these methods on the real bridge crane. The final project topic is separated into 3 sections: estimating parameters using Recursive Least Square method, designing a full state feedback controller, and using the controller to draw predefined shapes.

Part 1: RLS Estimation

The first-order system example of RLS was covered from the lecture. So, the team extended the RLS estimation to a second-order system on the real bridge crane. To use RLS estimation, a system model is needed:

$$H(s) = \frac{\theta(s)}{V_t(s)} = \frac{-\frac{\omega_n^2}{g}s}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{1}$$

Equation 1 shows the bridge crane model that was provided in Lab 3 from the course. To validate the Simulink results, a system model assuming a cable length of 1 meter and damping ratio of 0.008 was made. Figure 1 showcases the RLS convergence and the estimated parameters converged after 150 samples.

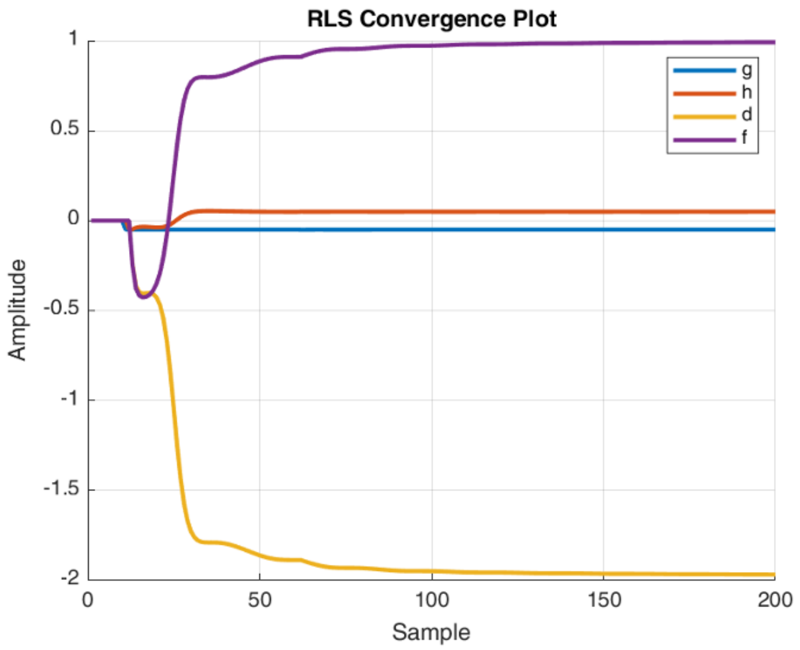


Figure 1: RLS Convergence Plot of Simulated Model

After calculating the four parameter values, the transfer function of the estimated model was then able to be obtained.

$$G(s) = \frac{-s}{s^2 + 0.05011s + 9.81} \quad (2)$$

$$G(s) = \frac{-0.9983s + 0.000163}{s^2 + 0.05016s + 9.81} \quad (3)$$

Equation 2 is the transfer function that is used for the input model, while Equation 3 represents the transfer function of the estimated model. Since the estimated parameters and the actual parameters were similar to one another, it's proven that the model was validated.

The data were collected from the bridge crane by giving the trolley trapezoid velocity command as an input, and measuring the payload deflection as an output. Additionally, a forgetting factor of 0.99 was used to disregard the earlier data values. The results of the RLS estimation algorithm of the actual bridge crane system is showcased in Figure 2 below, which shows that all four parameters started to converge after 150 samples.

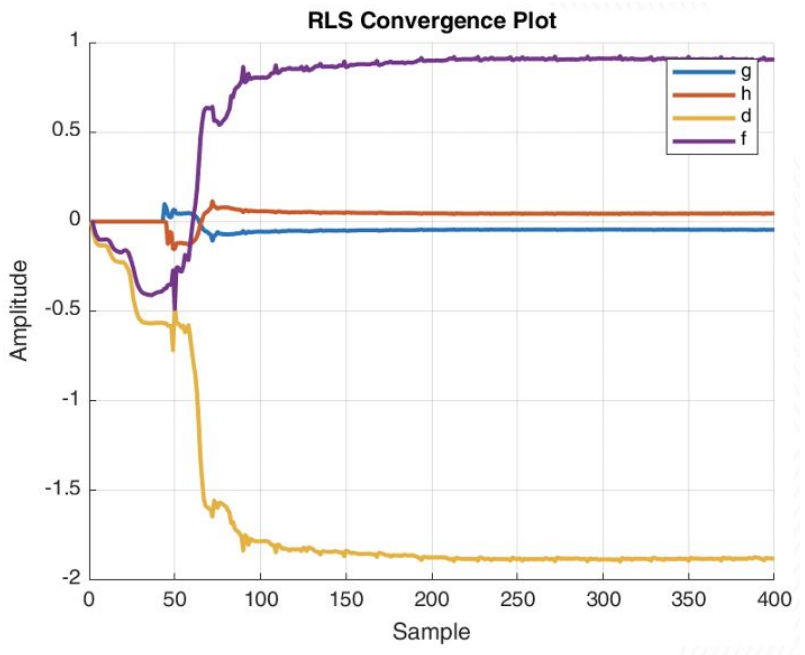


Figure 2: RLS Convergence Plot of the Actual Bridge Crane System

The actual length was 1 meter and the actual damping ratio was 0.0078. For the estimated value, shown in Table 1, the cable length was 0.9189 meter which is reasonable, but the damping ratio was 0.2938 which is significantly different from the actual damping ratio. Figure 3 displays a graph of the velocity inputted to the bridge crane system and the resulting payload deflection with respect to time. As seen in this figure, the damping ratio of the payload demonstrates a decreasing trend. As a result, the fact that the true damping ratio of the system does not seem to stay constant as time progresses may have caused the inaccurate RLS estimation.

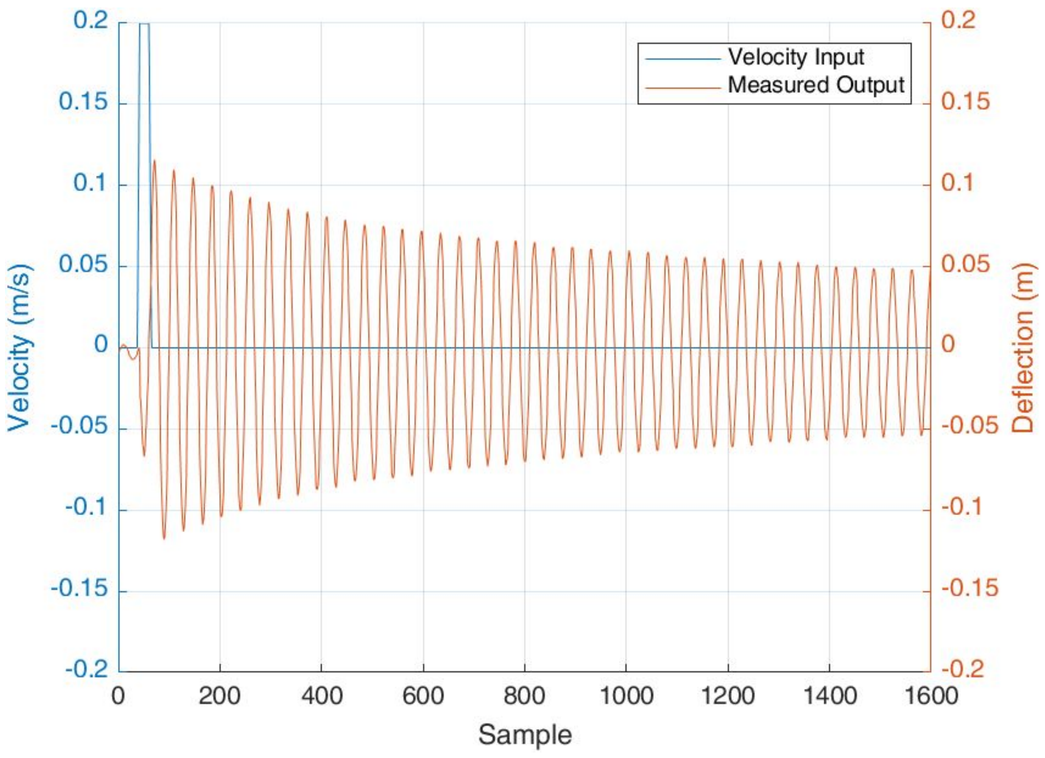


Figure 3: Bridge Crane Input-Output Relationship

RLS estimation data has been collected for three different cable lengths which were 1 meter, 0.8 meters, and 0.5 meters, and these results are displayed in Tables 1, 2, and 3, respectively. While the estimated cable length was similar to the actual one overall, the damping ratio ζ was extremely different. However, the ζ value shows a decreasing trend as the cable length decreases.

Table 1: RLS Estimation Data for a Cable Length of 1 Meter

	Actual	**Estimated**	**Error**
Cable Length (m)	1	0.9189	8.82%
Damping Ratio ζ	0.0078	0.2938	97.31%

Table 2: RLS Estimation Data for a Cable Length of 0.8 Meters

	Actual	**Estimated**	**Error**
Cable Length (m)	0.8	0.7545	6.03%
Damping Ratio ζ	0.0078	0.1948	96.00%

Table 3: RLS Estimation Data for a Cable Length of 0.5 Meters

	Actual	**Estimated**	**Error**
Cable Length (m)	0.5	0.4702	6.33%
Damping Ratio ζ	0.0078	0.0883	91.16%

Part 2: State-Feedback Controller

Next, it was desired to apply a state-feedback controller to the actual bridge crane system in order to minimize the residual oscillations of the payload. One of two types of state-feedback controllers were considered to be utilized: a full state feedback controller (where state feedback gains are calculated based on pole selection of the controlled system), and an LQR state-feedback controller (where state feedback gains are selected to minimize a quadratic cost function). Since a full state feedback controller offers more customizability in that it allows the user to create the desired characteristics and poles of the controlled system, and since the settling time of the system was prioritized much more than the energy usage of the system's actuators, it was decided to create and apply a full state feedback controller as opposed to an LQR controller to the actual bridge crane system.

Equation 4 represents the state-space model of the bridge crane system that was utilized to create the state feedback gains, where ω_n and ζ represent the natural frequency and damping of the uncontrolled system, respectively:

$$\begin{bmatrix} \theta L \\ \dot{\theta L} \\ x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_n^2 & -2\zeta\omega_n & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \int \theta L \\ \theta L \\ \int x \\ x \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} v(t) \tag{4}$$

The first two angular states represent the integral of the payload's deflection and the payload's deflection, respectively; whereas, the two translational states represent the integral of the payload's position and the payload's position, respectively. As stated in the previous section of this report, the damping of the original system was equal to approximately 0.0078. Additionally, as the angular states depend on the cable length of the pendulum, it was assumed that the pendulum would have a cable length of 1 meter when creating the controller; as a result, the natural frequency of the system was equal to approximately 3.132 rad/s.

The following represent the selected poles of the state feedback controller:

$$P_1 = -1.556 + 2.7125i, \qquad P_2 = -1.556 - 2.7125i, \qquad P_3 = -2 + 2i, \qquad P_4 = -2 - 2i$$

The first two poles, P_1 and P_2, are complex conjugates of one another and correspond to a desired damping of 0.5 and a desired natural frequency equal to the system's actual natural frequency of 3.312 rad/s. The last two poles, P_3 and P_4, are also complex conjugates of one another and were arbitrarily chosen to be near poles P_1 and P_2 (in reality, poles P_3 and P_4 were chosen by experimenting with a number of different poles and determining the poles that helped minimize the settling time of the controller the most in simulation). With the selection of these poles, the corresponding state feedback gains, and the resulting controller is shown in equation 5:

$$K_1 = 12.2101, \qquad K_2 = 0.5689, \qquad K_3 = 8, \qquad K_4 = 6.5144$$

$$u = -K\boldsymbol{x} = -[12.2101, 0.5689, 8, 6.5144] \begin{bmatrix} \int \theta L \\ \theta L \\ \int x \\ x \end{bmatrix} \tag{5}$$

The following figure showcases the performance of the full state feedback controller in simulation and on the actual system for a cable length of 1 meter. In addition, the figure displays input velocity commands applied to the bridge crane system over time and the resulting performance of the controller for cable lengths of 0.6 m and 0.8 m in order to test the robustness of the full state feedback controller:

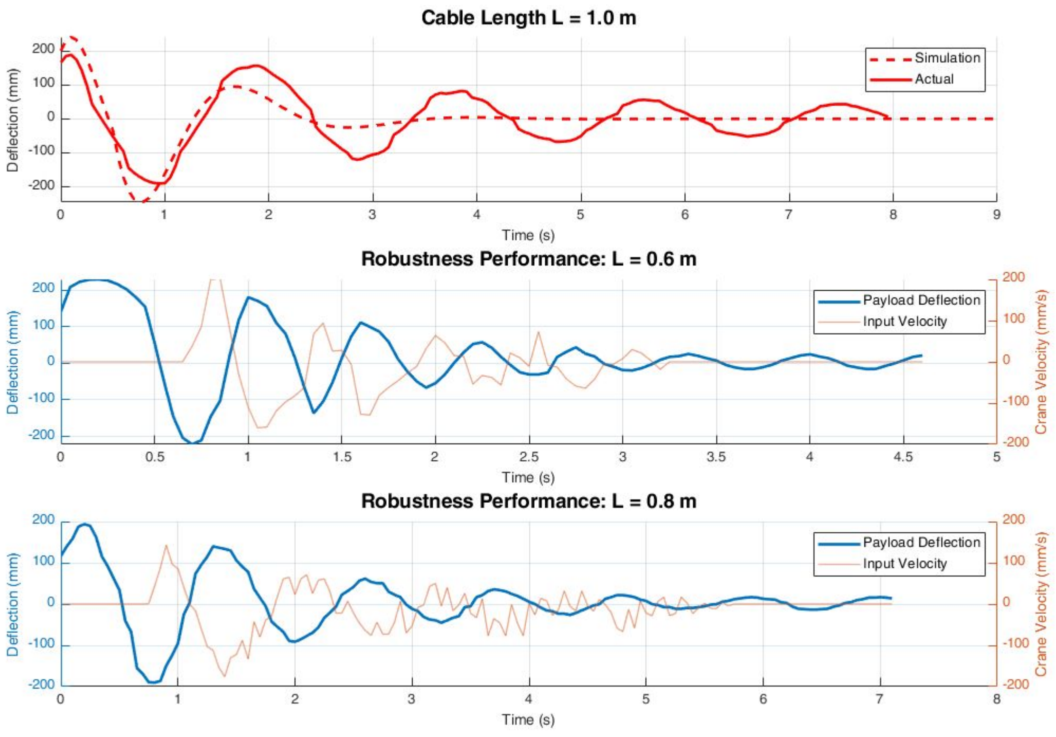


Figure 4: Full State Feedback Controller Performance

As observed in the first plot in Figure 4, it's seen that the controller was effectively able to reduce the deflection of the system towards zero over a short period of time. However, while the theoretical system's response in simulation and the actual system's response were similar for around the first two seconds, it's additionally noticed that the residual oscillations of the actual system does not decrease nearly as quickly as predicted by the simulation. This discrepancy can be explained by a number of different reasons/factors. Firstly, as the bridge crane system is, in reality, a nonlinear system, assumptions were made (in particular, the small-angle approximation) in order to linearize the system and facilitate calculations in creating a proper feedback controller; however, the process of applying these assumptions caused our model to not be entirely representative of the actual system, which may have caused the developed full state feedback controller to not function as effectively as in the simulations. Secondly, for any type of feedback controller, sensors must be utilized in an attempt to measure the instantaneous states of the system to be used for the control algorithm. For this specific system, a camera sensor was used to measure the deflection states of the system. However, all sensors are additionally subject to at least a small level of noise that interferes with the accuracy of the sensors' readings. As a result, for this specific system, the noise applied to the camera sensor may have caused the sensor to overestimate the deflection of the payload, which would cause the measured performance of the full state feedback controller on the actual system to be less consistent with the simulated results.

Additionally, as stated previously, to test the robustness of the full state feedback controller, the controller was applied to the system for cable lengths of 0.6 m and 0.8 m, whose results are displayed in the second and third plots of the figure, respectively. As seen in these plots, it's observed that the controller was properly able to stabilize the system for these different cable lengths because the system's deflections clearly reduced towards zero as time progressed; to elaborate, the controller was able to successfully direct the system towards the zero state even despite the fact that disturbances were exerted on the system (as simulated by the input velocities applied to the controller). Hence, the developed full state feedback controller displays a high, satisfactory level of robustness when applied to the bridge crane system.

Part 3: Pre-Programmed Trajectory

Furthermore, the last objective of the project was to draw a complicated shape using a pre-programmed trajectory with the bridge crane. The original plan was to use the full state feedback controller from Part 2 to minimize the vibration. However, the team was not able to find a way to implement the controller and the trajectory at the same time to the bridge crane PLC. As a result, the team decided to apply an input shaper to trace the trajectory instead of a full state feedback controller.

The star shape was drawn for this part because it has sharp corners which helps to determine the robustness of the input shaper. For the input shaper the ZV, ZVD, and EI shapers were tested. Of all shapers tested, the ZVD shaper displays the best trajectory geometry at the sharp corners of the star. Figure 5 shows the result of the pre-programmed trajectory for both the unshaped model and the ZVD-shaped model:

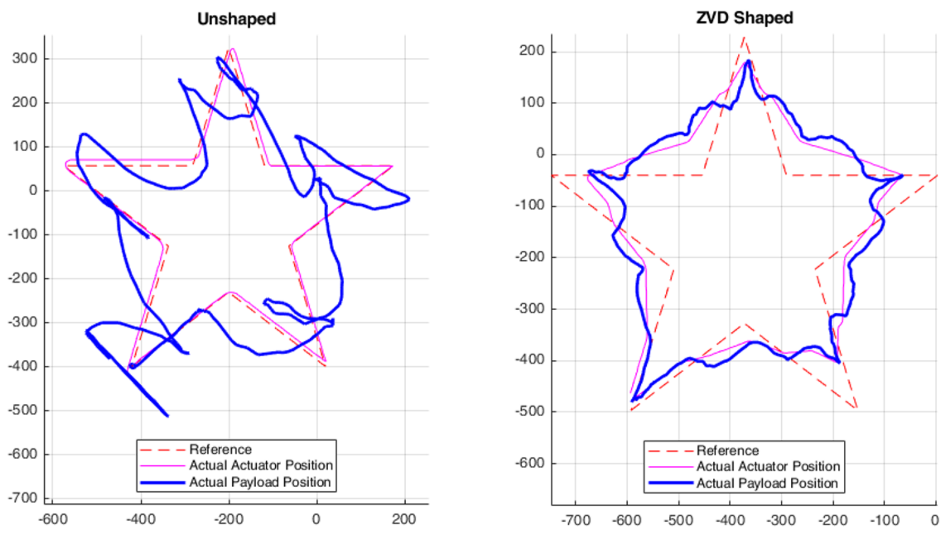


Figure 5: Result of the Star Shape Trajectory for an Unshaped Model (Left) and a ZVD-Shaped Model (Right)

As observed in the figure, the unshaped model predictably didn't follow the trajectory of the star at all due to the major oscillations of the payload. For the ZVD-shaped model, despite the fact that a small amount of vibration is present, the positions of the actuator and the payload are almost identical at all points along the trajectory, indicating that the ZVD shaper effectively minimized the oscillations of the payload. However, the trajectory followed by the actual system is slightly different from the reference trajectory. This is attributed to the ZVD shaper automatically slowing down when the trolley rapidly changes direction in order to reduce the oscillation.

Moreover, the pre-programmed trajectory was attempted for three cable lengths to test the robustness of the designed shaper, and the results of this testing is displayed in Figure 6, as shown below.

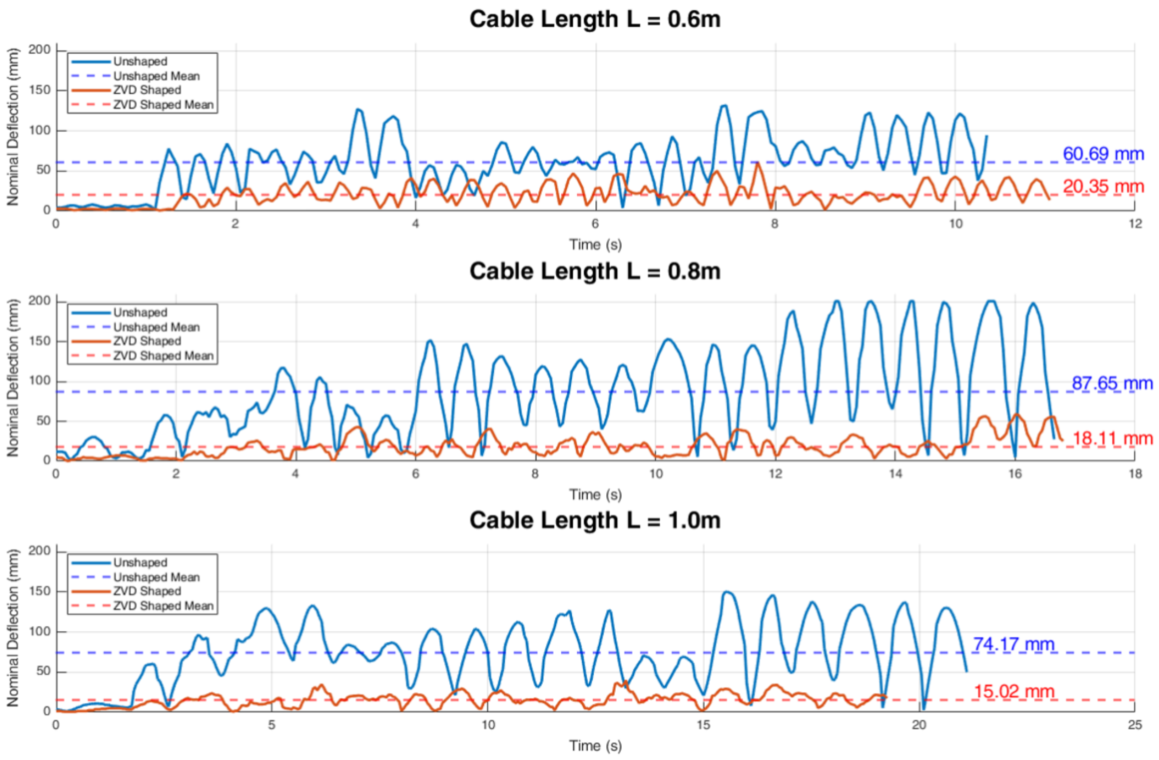


Figure 6: Payload Deflection for ZVD-Shaped Model and Unshaped Model for Three Different Cable Lengths

As indicated in the above figure, the payload deflection for the shaped trajectory shows a much smaller average value than that for the unshaped case across all three different cable length values. As the ZVD shaper was designed for a cable length of 1 meter, the mean deflection of 1 m cable length was predictably the smallest at approximately 15.02 mm; additionally, the mean deflection of the 0.8 m and 0.6 m cable lengths were 18.11 mm and 20.35 mm, respectively. Hence, it can be stated that as the actual cable length deviates more from the designed cable length, the modeling error increases. The shaper became less robust against vibration when the actual natural frequency deviates from the designed natural frequency.

Conclusion

As stated previously, the primary goal of this project is to connect the bridge between the theoretical knowledge that we learned in the lectures and the application of this knowledge on real world systems. On the bridge crane system, RLS estimation, full state feedback controller, and input shaping on the predefined trajectory were applied. The team faced a number of challenges during the process of the project. For instance, the RLS estimator was not able to capture the real system entirely accurately, and additionally the state feedback controller and the pre-made trajectory were not able to be simultaneously implemented into the system. In the end, the team was able to overcome the majority of them, and successfully implemented all of the topics on the real world system.

In summation, the team was able to estimate the system natural frequency with less than 10% of error. The full-state feedback controller was able to stabilize induced vibration across all cable lengths. Therefore, we successfully applied the knowledge that we learned in the lectures to the bridge crane.

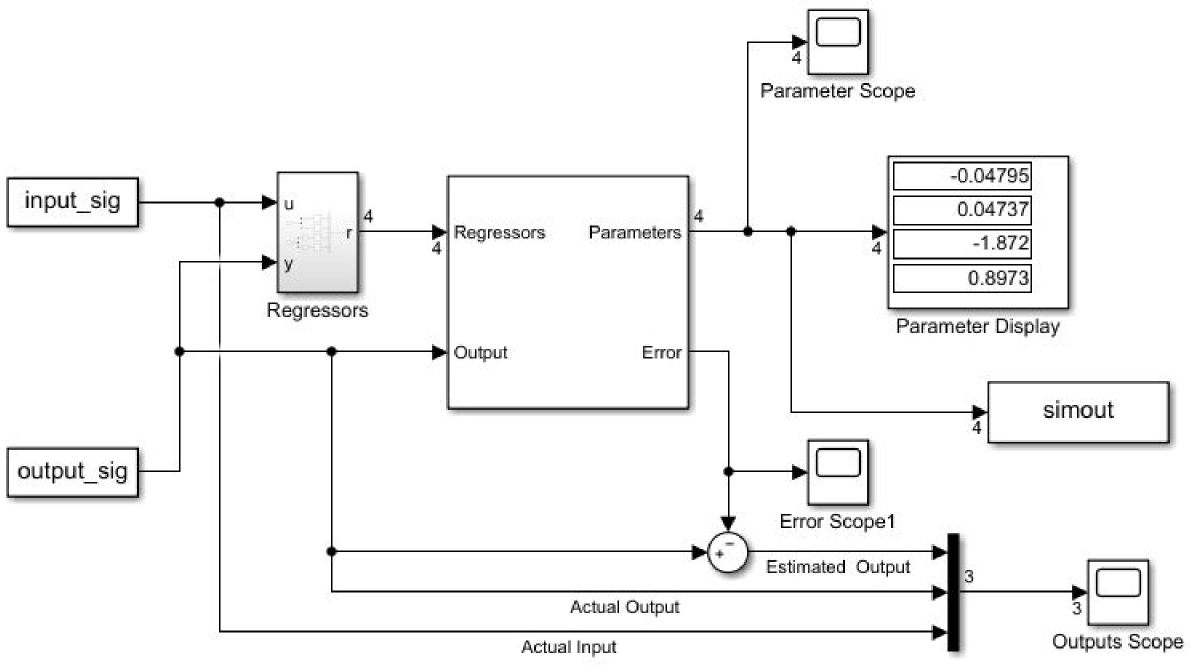
Appendix

MATLAB Code for RLS Estimation:

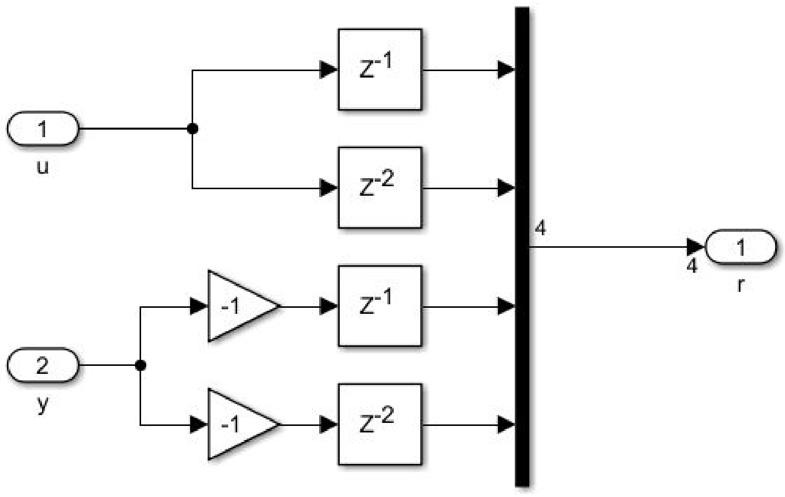
```
%% Housekeeping
clear all
clc

%% Initialization
addpath('./Data')

%% Data Extraction
filename = 'GT_BridgeCrane_Data_RSL_1.csv';
data = importfile_bridge(filename, [2 inf]);

t = data.TimeYdirsec;
u = data.YActualVelocitymmsec/1000;
y = -data.YPayloadDeflectionmm/1000;
y = y - mean(y(1:10));

Ts = 0.05;

%% Plotting
figure(); hold on; grid on;
yyaxis left
plot(u)
ylabel('Velocity (m/s)')
ylim([-0.2, 0.2])
yyaxis right
plot(y)
ylabel('Deflection (m)')
ylim([-0.2, 0.2])
xlabel('Sample')
legend('Velocity Input', 'Measured Output')

%%
input_sig = timeseries(u,t);
output_sig = timeseries(y,t);

sim('RLS_Estimator_Block');

%%
g = simout.Data(:, 1);
h = simout.Data(:, 2);
```

```
d = simout.Data(:, 3);
f = simout.Data(:, 4);

%% Plot the estimated parameters
figure; hold on; grid on;
plot(g, 'linewidth', 2)
plot(h, 'linewidth', 2)
plot(d, 'linewidth', 2)
plot(f, 'linewidth', 2)
xlabel('Sample')
ylabel('Amplitude')
xlim([0, 400])
title('RLS Convergence Plot')
legend('g', 'h', 'd', 'f')

%%
d = mean(d(end-20:end));
f = mean(f(end-20:end));
g = mean(g(end-20:end));
h = mean(h(end-20:end));

sys_Z = tf([g, h], [1, d, f], Ts);
sys_C = d2c(sys_Z);

%%
wn = sqrt(sys_C.Denominator{1}(3));
zeta = sys_C.Denominator{1}(2)/(2*wn);
L = 9.81/wn^2;

fprintf('The estimated cable length is %.4f (m) \n', L)
fprintf('The estimated natural frequency of %.4f (rad/s) \n', wn)
fprintf('The estimated zeta = %.4f \n', zeta)

%% Actual parameters
amp1 = 0.1237;
amp2 = 0.1177;
delta = log(amp1/amp2);
zeta_actual = 1/sqrt(1+(2*pi/delta)^2);
L_actual = 1;
wn_actual = sqrt(9.81/L_actual);

fprintf('The actual cable length is %.4f (m) \n', L_actual)
fprintf('The actual natural frequency of %.4f (rad/s) \n', wn_actual)
fprintf('The actual zeta = %.4f \n', zeta_actual)
```

```
%% State-Space Representation
[A, B, C, D] = tf2ss(sys_C.Numerator{1}, sys_C.Denominator{1});

save bridgeCraneID.mat A B C D wn zeta

% RLS_Estimator_Block.slx
```



```
% Regressors:
```



MATLAB Code for Full State Feedback Controller:

```
%% Full State Feedback Controller (u = -Kx)

% Housekeeping:
close all
clear
clc

g = 9.81; % Acceleration due to gravity in m/s^2
L = 1; % Cable Length
wn = sqrt(g/L); % Natural Frequency of Pendulum
zeta = 0.0078; % Damping of system

% Setting State Matrices of System:
A = [0, 1, 0, 0;
   -g/L, (-2*wn*zeta), 0, 0;
   0, 0, 0, 1;
   0, 0, 0, 0];
B = [0; 1/L; 0; 1];
C = [0 (-1/L) 0 0; 0 0 0 0; 0 0 0 1];
D = 0;

% Creating the desired poles of the controlled system:
wn_des = 1 .* sqrt(g/L); % Desired natural frequency of the controlled system
zeta_des = 0.5; % Desired damping of the controlled system
Pd1 = (-zeta_des .* wn) + (sqrt((zeta_des.^2) - 1).*wn_des);
Pd2 = conj(Pd1);
Pd3 = -2 + 2i;
Pd4 = conj(Pd3);
Pd = [Pd1; Pd2; Pd3; Pd4];

K = place(A, B, Pd); % Calculating the full state feedback controller gains

A_new = A - (B*K); % Calculating the system's new state matrix

csys = ss(A_new, B, C, D); % Creating controlled continuous-time system

% Performing Linear Simulation:
t = [0:0.01:9];
u = ones(length(t), 1);
x0 = [0 (0.2*L) 0 0];
[y1, t1, x1] = lsim(csys, u, t, x0);

% Plotting Results:
```

```
figure
subplot(4, 1, 1)
plot(t1, x1(:, 1), 'b')
xlabel('Time (s)')
ylabel('Integral of Deflection')

subplot(4, 1, 2)
plot(t1, x1(:, 2), 'b')
xlabel('Time (s)')
ylabel('Deflection')

subplot(4, 1, 3)
plot(t1, x1(:, 3), 'r')
xlabel('Time (s)')
ylabel('Integral of Position')

subplot(4, 1, 4)
plot(t1, x1(:, 4), 'r')
xlabel('Time (s)')
ylabel('Position')
```

MATLAB Code for Pre-programmed Trajectory:

```
%% Housekeeping
clear all
clc

%% Velocity limits
x_vel_max = 100;
y_vel_max = 100;
vel_max = [x_vel_max, y_vel_max];
Ts = 0.05;
SPS = 1/Ts;

%% Define Waypoints
wp1 = [200, 200];
wp2 = [346.76, 312.55];
wp3 = [493.51, 200];
wp4 = [438.09, 382.59];
wp5 = [596.76, 504.28];
wp6 = [401.15, 504.28];
wp7 = [346.76, 683.45];
wp8 = [292.37, 504.28];
wp9 = [96.76, 504.28];
wp10 = [255.43, 382.59];
wp11 = [200, 200];
WPs = [wp1; wp2; wp3; wp4; wp5; wp6; wp7; wp8; wp9; wp10; wp11];

%% Calculated the time needed between each waypoints
[row, col] = size(WPs);
time = zeros(1, row);
time(1) = Ts;
actuator_pct = zeros(row, col);

for i = 1:row
  if i == row
     break
  end

  % Reset the array
  time_temp = zeros(1, 2);
  signs = ones(1, 2);

  % Looping x and y direction
  for j = 1:col
```

```
        % Get the direction of actuator moving
        signs(j) = sign(WPs(i+1, j) - WPs(i, j));

        time_temp(j) = abs(WPs(i+1, j) - WPs(i, j))/vel_max(j);
    end

    time(i+1) = time(i) + max(time_temp);

    actuator_pct(i+1, :) = signs.*time_temp/max(time_temp)*100;
end

actuator_pct = round(actuator_pct);

%% Time Adjustment
% time(end) = time(end) + 0.8;

%% Trajactory Planning
idx_start = zeros(1, row);
idx_stop = zeros(1, row);

for i = 1:row
    if i == row
        break
    end

    idx_start(i) = floor(time(i) * SPS);
    idx_stop(i) = floor(time(i+1) * SPS);

    x_traj(1, idx_start(i): idx_stop(i)) = actuator_pct(i+1, 1);
    y_traj(1, idx_start(i): idx_stop(i)) = actuator_pct(i+1, 2);
end

%% Shaper Design
[ZVShaper, ZVDShaper, EIShaper] = shaperDesign();

%% Input Shaping
[~, x_traj_shaped] = convolve(x_traj', ZVDShaper, Ts);
[~, y_traj_shaped] = convolve(y_traj', ZVDShaper, Ts);

x_traj_shaped = round(x_traj_shaped);
y_traj_shaped = round(y_traj_shaped);

%% Plotting
xpos = wp1(1);
```

```
ypos = wp1(2);

for i = 1:length(x_traj)
    xpos(i+1) = xpos(i) + Ts * x_vel_max * x_traj_shaped(i)/100;
    ypos(i+1) = ypos(i) + Ts * y_vel_max * y_traj_shaped(i)/100;
end

%% Animation
figure; hold on; grid on; axis equal;
scatter(WPs(:, 1), WPs(:, 2), 'go')

line = animatedline('Color', 'r', 'LineWidth', 2);

for k = 1:length(xpos)
    addpoints(line, xpos(k), ypos(k))
%     drawnow limitrate
end

drawnow

%% Save to the excel sheet
saveCSV(x_traj_shaped, y_traj_shaped)
```