Combined Translational and Rotational Mechanical System with PID Controller Analysis of Controllability via Fitting Equation, Stability, and State-Space Representation

Paul Joshua C. Beltran
Gokongwei College of Engineering
De La Salle University Manila
Pasig City, Philippines
paul_beltran@dlsu.edu.ph

Felix Gabriel S. Montanez
Gokongwei College of Engineering
De La Salle University Manila
Quezon City, Philippines
felix_gabriel_montanez@dlsu.edu.ph

Abstract— **This study analyzed the characteristics of a combined translational and rotational mechanical design by obtaining the transfer function with mechanical linkage, applying PID controller, and adding controllability by inputting the value of the rise time, percent overshoot, and settling time. Trial and error were used to collect the data for values Kd, Ki, and Kp (output) based on specific rise time, percent overshoot, and settling time (input). Multiple linear and polynomial regression were implemented to find the relationship between the inputs and outputs. The step response function was plotted based on the transfer function with the PID controller and the Kd, Kp, and Ki (with its value derived from inputs and some polynomial non-transcendental function). In assessing the relationship of the input variable to the output, the importance of the information was increased and decreased accordingly based on the output of the step response info command, which shows the rise time, settling time, and percent overshoot. Afterward, input values were used based on the results. However, since its result is sometimes turbulent and unpredictable, we tried to find sufficiently accurate results. It is recommended to do additional mathematical treatments that have higher accuracy.**

Keywords—Translational and Rotational Mechanical system, Matlab, Fitting Equation, Stability, PID controller, State-Space Representation

I. Introduction

In analyzing mechanical systems, there are differential equations modeling which is based on the type of motion being done in the system. Translational mechanical systems travel in a straight line. These systems primarily have three fundamental components. Mass, spring, and damper are those.A translational mechanical system's mass, elasticity, and friction act as opposing forces when a force is applied to it. The algebraic total of the forces operating on the system is zero since the opposing forces and the applied force are acting in the opposite directions. A body's mass is a quality that retains kinetic energy. A force is countered by an opposite force related to mass if it is applied to a body with mass M. The body's acceleration is inversely proportional to this opposing force. An element that retains potential energy is spring. Due to the spring's elasticity, when a force is exerted to spring K, it will be met by an opposite force. The spring's displacement is inversely proportional to this opposing force. Whenever a force is exerted to the damper, the damper's friction produces an opposite force that counteracts it. The relationship between this opposing force and the body's velocity is linear.

Mechanical systems that rotate around a fixed axis. These systems primarily have three fundamental components. They are damper, spring, and moment of inertia. When a torque is given to a rotating mechanical system, counter torques result from the system's moment of inertia, elasticity, and friction. The algebraic total of torques operating on the system is 0 since the applied and opposed torques are in different directions. The mass of a mechanical translation system stores kinetic energy. Similar to this, the moment of inertia in rotating mechanical systems stores kinetic energy. When a torque is applied to a body that also has a moment of inertia, the opposite torque is generated as a result of the moment of inertia. The body's angular acceleration is inversely proportional to this opposing torque. A spring holds potential energy in a mechanical translation system. Similar to this, a torsional spring in a rotating mechanical system stores potential energy. Because torsional springs are elastic, if a torque is exerted on one, it will be countered by another due to the spring's elasticity. The angular displacement of the spring is directly related to this opposing torque. Due to the damper's rotational friction, if a torque is exerted to the damper, an opposite torque will result. The relationship between this opposing torque and the body's angular velocity is linear [1].

Step response have characteristics that can be analyzed. These include rise time, percent overshoot, and settling time. Rise time measures how long it takes a reaction to increase from 0% to 100% of its ultimate value. This is true for underdamped systems. Consider the time period from 10% to 90% of the final value for the overdamped systems.Rise time is the amount of time necessary for the reaction to stabilize and remain within the designated tolerance ranges around the final result. The tolerance bands typically range between 2% and 5%[2]. The highest value less the step value divided by the step value is known as the percentage overshoot. The overshoot in the case of the unit step is just the step response's maximum value minus one[3].

Systems that combine the two systems result in a combined translational and rotational mechanical system. This form of the system has multiple applications specifically those that have linear and rotational motion like conveyor belts, linear actuators, 3d printers, etc. Given the applications of this system, this study aims to simulate a combined translational and rotational mechanical system and evaluate the transfer function, PID controller, stability, and state space representation of the system as well as the relationship between rise time, percent overshoot, and settling time to the PID controller.

II. METHODOLOGY

The translational and rotational mechanical system to be analyzed is shown in Figure 1, where there are two gear trains, the first one contains the moment of inertia J_1, torque, inertia θ_1 rotational friction D_1 connected to the ground, and rotational friction D_2. The number of teeth of gear 1 is 15 while gear 2 has 25. Gear 2 contains rotational friction D_3, D_4 which is connected to the ground, and a moment of inertia J_2 attached to a pulley with a radius of 4. This pulley is connected to a linear translational system which has a mass of 6kg at x(t) and is connected to a damper and spring which are both connected to the ground.

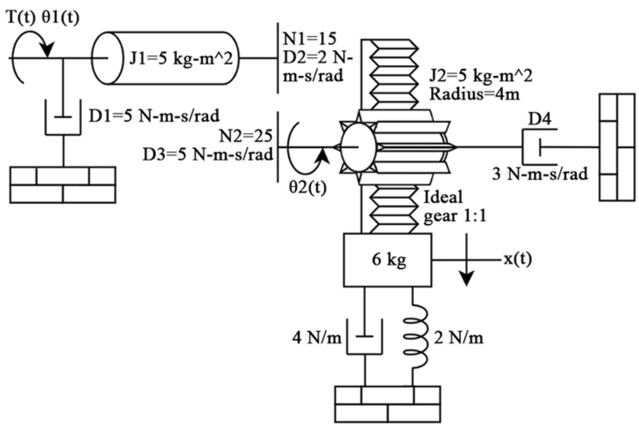


Fig. 1. Diagram of Translational and Rotational Mechanical System

Given this diagram, the system's transfer function can be obtained. The system's transfer function is significant because this is needed to obtain stability status, state space representation, and PID controllers. The transfer function can be obtained by analyzing equations of motions and solving for X(s)/T(s). Each bounded input must result in a bounded output for a system to be stable. The natural response must be close to zero for a linear, time-invariant system to be stable as the time approaches infinity[4]. The number of poles in each sector of the s-plane is determined using the Routh-Hurwitz stability criterion. In order to determine how many closed-loop system poles are located in the left half-plane, the right half-plane, and on the j-axis, the approach involves two steps: (1) creating a data table known as a Routh table; and (2) interpreting the Routh table. After completing the Routh table, this can be interpreted to determine the stability of the system.

When representing a system as a first-order differential equation of the input (u), output (y), and state (z), state space representation is used (x). The state and observation equations are represented in Eq.1 and 2, respectively, by and if the system is linear.

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du \tag{2}$$

where A, B, C, and D are the matrices. Because D is the feedthrough term, it should be emphasized that D = 0 in the majority of situations[5].

The forced response, also known as the steady-state response or specific solution, is added to the natural response to create the output response of a system. A transfer function's poles are any roots of the denominator that are also roots of the numerator, or (1) the values of the Laplace transform variable, s, that lead to the transfer function becoming infinite[6].The zeros of a transfer function are either (1) any Laplace transform variable, s, values that result in the transfer function being zero, or (2) any roots of the transfer function's numerator that are also roots of the function's denominator. This can be plotted that results in a pole-zero map which can also evaluate the stability of the system.

Many businesses use PID control to govern an output by adjusting an input variable[7]. PID control is similar to proportional control, but with the inclusion of algorithm components pertaining to the integral and derivative values of the error data. Instead of being sensitive to the present error value alone, this gives the algorithm a component of history[8]. Controllers and controlled subjects make up PID control systems. The control technique can provide acceptable outcomes by modifying parameters like proportional coefficient, integral coefficient, and differential coefficient. The PID control approach is based on proportional control. Integral control can reduce overadjustment but can also remove steady-state faults. Differential control can reduce overadjustment while simultaneously speeding up reaction time[9].

By including a term proportional to the time derivative of the error signal, the stability and overshoot issues that happen when a proportional controller is employed at a high gain can be reduced. To get a severely damped response, the damping value can be changed[10].

PD control effectively addresses the proportional control's overshoot and ringing issues, but it does not resolve the steady-state error issue. Fortunately, by including an integral term to the control function, which becomes, it is feasible to avoid this despite using relatively modest gain. A proportional controller (Kp) will lower the rising time and the steady-state error, but never completely remove it. The steady-state error will be eliminated by an integral control (Ki), but the transient response could get worse. A derivative control (Kd) will result in an improvement in the transient response, a decrease in overshoot, and an increase in system stability. Table 1 shows the results of raising each controller variable. For simulating the system, MATLAB was used.

TABLE 1

PID Tuning Table [11]

Response	Rise Time	Overshoot	Settling time	S-S Error
Kp	Decrease	Increase	Minor change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Minor change	Decrease	Decrease	Minor change

III. RESULTS AND DISCUSSION

Simplifying the diagram, the equations for the Equivalent moment of inertia and the equivalent damper are shown in Eq. 3 and 5, and substituting the values obtain in Eq. 4 and 6.

$$J_e = J_1\left(\frac{N_2}{N_1}\right)^2 + J_2 \tag{3}$$

$$J_e = 5\left(\frac{25}{15}\right)^2 + 5 = 18.99 \tag{4}$$

$$D_e = D_1\left(\frac{N_2}{N_1}\right)^2 + D_2\left(\frac{N_2}{N_1}\right)^2 + D_3 + D_4 \tag{5}$$

$$D_e = 5\left(\frac{25}{15}\right)^2 + 2\left(\frac{25}{15}\right)^2 + 5 + 3 = 27.44 \tag{6}$$

Figure 2 shows the free body diagram of the mass in the system showing the forces acting on it. Since the mass is connected to the pulley it has a downward force acting on it called F(s). The opposing forces on the mass include inertia, damper, and spring. This equation is shown in Eq. 7. Obtaining the equation for the simplified diagram is shown in Eq. 8 where torque multiplied by gear ratio is forced applied while opposing force is the force of the mass multiplied by the radius and the moment of inertia equivalent and damper equivalent multiplied by $\theta_2(s)$. Substituting the values result in Eq. 9.

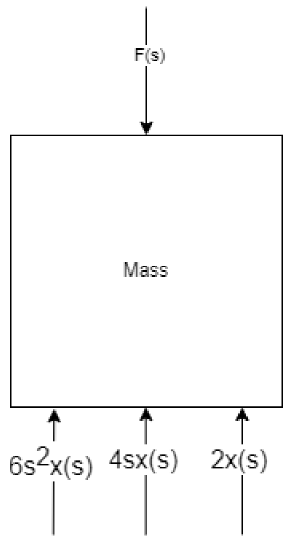


Fig. 2. Freebody diagram of Mass

$$F(s) = (6s^2 + 4s + 2)\,X(s) \tag{7}$$

$$(J_e s^2 + D_e)\,\theta_2(s) + 4F(s) = \left(\frac{N_2}{N_1}\right)T(s) \tag{8}$$

$$(18.89s^2 + 27.44)\theta_2(s) + (24s^2 + 16s + 8)X(s) = 1.67 \tag{9}$$

Since x(s) is connected to the pulley, its relation is shown in Eq. 10. This can be used to write $\theta_2(s)$ in terms of x(s) which is shown in Eq. 11 This can be substituted to Eq. 9 resulting to Eq. 12. Simplifying this further arrives to Eq. 13 and to get the transfer function the ratio of x(s) over T(s) was obtained resulting to Eq.14

$$x(s) = r\theta_2(s) \tag{10}$$

$$\theta_2(s) = \frac{x(s)}{4} \tag{11}$$

$$\frac{(18.89s^2+27.44)}{4}X(s) + (24s^2 + 16s + 8)X(s) = 1.67T(s) \tag{12}$$

$$\frac{(114.89s^2+91.44s+32}{4}X(s) = 1.67T(s) \tag{13}$$

$$\frac{X(s)}{T(s)} = \frac{6.68}{114.89s^2+91.44s+32} \tag{14}$$

Using the Routh-Hurwitz stability criterion, the routh table was done and is shown in table 2. Following the rules for the Routh-Hurwitz stability criterion, the results were obtained which show that there is no sign change demonstrating the system is stable.

TABLE 2

PID Routh Table

s2	114.89	32
s1	91.44	0
s0	32	0

To obtain the poles and the zeroes the denominator was set equal to 0 to obtain the roots. Applying the quadratic formula, the equation is shown in Eq. 16 and the resulting roots are shown in Eq. 18. MATLAB simulation was also done and shown in Figure 3.

$$114.89s^2 + 91.44s + 32 = 0 \tag{15}$$

$$s = \frac{-91.44 \pm \sqrt{4(114.89)(32)}}{2(114.89)} \tag{16}$$

$$s = \frac{-91.44 \pm 79.65i}{229.78} \tag{17}$$

$$s = -0.40 \pm 0.35i \tag{18}$$

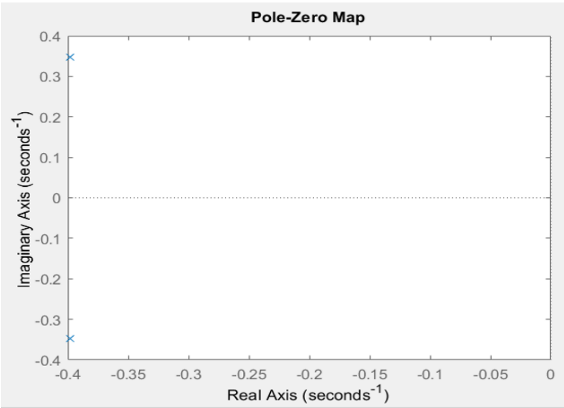


Fig. 3. Pole Zero Map

In obtaining the state space representation, the transfer function is cross-multiplied and the resulting equation is shown in Eq. 19. Applying Laplace Transform results in Eq 18. Since state space representation needs the equation for the highest order variable, ẍ, the equation obtained is shown in Eq. 22. Arranging Eq.22 to matrix forms outputs the state space representation is shown in Eq. 23, and since y is the output and in Eq. 22, the output is x, this results to Eq.24

$$X(s)\,(114.89s^2 + 91.44s + 32) = 6.68\,T(s) \tag{19}$$

$$114.89\ddot{x} + 91.44\dot{x} + 32x = 6.68\,T(s) \tag{20}$$

$$114.89\ddot{x} = -91.44\dot{x} - 32x + 6.68\,T(s) \tag{21}$$

$$\ddot{x} = -0.80\dot{x} - 0.28x + 0.06\,T(s) \tag{22}$$

$$\begin{pmatrix}\dot{x}\\ \ddot{x}\end{pmatrix} = \begin{pmatrix}0 & 1\\ -0.28 & -0.80\end{pmatrix} * x + \begin{pmatrix}0\\ 1\end{pmatrix} * T(s) \tag{23}$$

$$y = (1 \quad 0) * x \tag{24}$$

Simulating the system in MATLAB obtain Figure 4. where the step function was used to obtain the step response. The figure shows that the system has a significant rise time as well as a response approximately equal to 0.2. Applying PID controller results in Figure 5. where with the use of trial and error the values for Kp, Kd, and Ki were determined and a modified transfer function was made. As shown in Figure 5, the rise time is significantly smaller than the original, and the step response is at 1 which makes the modified system almost a perfect system.

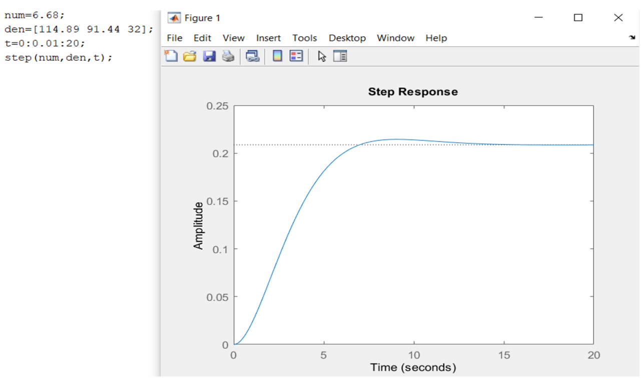


Fig. 4. Original Transfer function Step Response

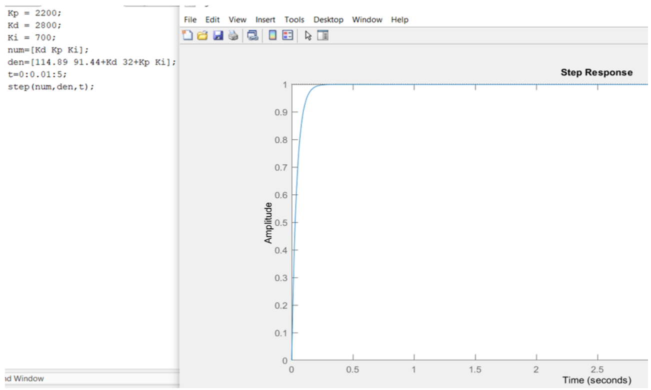


Fig. 5. Modified Transfer function Step Response

Because of the modification of the PID Controller a new block diagram can be made which is shown in Figure 6 and thus a new transfer function which is shown in Eq. 25. With a new transfer function, the Steady-state error can be determined for the modified system. The disparity between a system's input (command) and output in the limit as the time approaches infinity is known as a steady-state error The steady-state error is dependent on the system type and the input type [12].

The relationship between steady-state error and Ki is calculated by first obtaining the modified transfer function shown in Eq. 25. To obtain R(s), the Reference input signal is 1/s^2 which is a ramp unit, as the other types will make a zero error which is shown in Eq.26. This is a type 1 system due to the cubic function of the denominator[13]. Using the formula for Error constant shown in Eq. 27 and steady-state error in Eq. 28, the relationship between Ki and Steady-state error can be determined by obtaining ki in terms of SSE which is shown in Eq. 29.

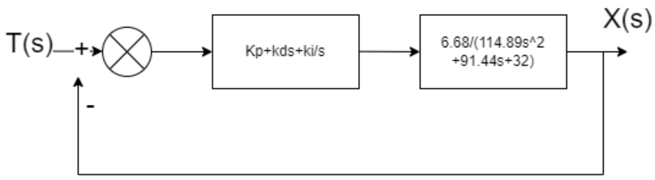


Fig. 6. Feedback Loop with PID controller

$$\frac{X(s)}{T(s)} = \frac{6.68k_d s^2 + 6.68\,k_p s + 6.68k_i}{114.89s^3 + (91.44+6.68k_d)s^2 + (32 + 6.68\,k_p)s + 6.68k_i} \tag{25}$$

$$R(s) = \frac{1}{s^2}(ramp\ unit)\ \&\ G(s) = \frac{6.68k_d s^2 + 6.68\,k_p s + 6.68k_i}{114.89s^3 + 91.44s^2 + 32s} \tag{26}$$

$$Error\ const:\ k_v = lim_{s->0}\, s\,G(s) = 0.20875\,ki \tag{27}$$

$$SSE = \frac{1}{k_v} = \frac{1}{0.20875k_i} \tag{28}$$

$$k_i = \frac{1}{0.20875*SSE} \tag{29}$$

In obtaining the relationships between the PID controllers, multiple linear polynomial regression, 2nd, and

3rd order polynomial regression, and multivariable input linear regression were done. Multiple polynomial regression gives a more accurate result yet it doesn't fully consider its relationship with another input variable. Columns 1, 2, 2, and 1 for tables 3,4,5, and 6 respectively were either small change variables or disregarded only on another input variable. Columns 4 on both tables 5 and 6 were completely disregarded to reduce the amount of fitting equations for simplicity. However, this also shows one problem of this approach which is that it has a lot of limiting assumptions that may potentially tamper with the accuracy. Multiple linear regression was done due to it considering the relationship of multiple input variables based on the specific output value. However, since it's a linear equation, it will have lower accuracy. Still, this can be resolved by finding its average with the polynomial regression. 3rd order polynomial regression are shown in tables 3 and 4, while 2nd order for tables 5 and 6. 3rd order polynomial regression uses the same methodology as 2nd order. The only difference is that there is the variable X3. Thus, the tables and matrix will be enlarged. Specifically, the table for summations will be 13 columns. As for the matrices, it'll be a 4x4, and 2 1x4s for A, X, and B, respectively. Table 7 shows the variables that needed to be listed and to be summed and table 8 shows the matrix of summations and variables to create the fitting equation while table 9 and 10 shows the table for summations for 3 input linear regression and matrix for summations for 3 input linear regression respectively. The table for multivariable input linear regression is shown in table 11. The inputs (X) were derived by trial-and-error with the values of Kd, Kp, and Ki. These are the variables to be changed where it is used in modifying the value of the numerator and denominator based on the modified transfer function. A step function is applied to the numerator, denominator, and time as well as a stepinfo function to determine the information of rise time, settling time, and overshoot to streamline the trial and error process.

TABLE 3

TABLE RELATIONSHIP OF KP AND KI VS DESTINED RISE TIME

Destined Rise Time	Kd	Kp	Ki
0.1	50	2050	2050
0.15	50	900	900
0.2	50	495	495
0.25	50	310	310
0.3	50	210	210
$y = -440000.0000x^3 + 330285.7143x^2 - 84114.2857x + 7593$			

TABLE 4

TABLE RELATIONSHIP OF KD AND KI WITH DESTINED SETTLING TIME

Destined Settling Time	Kd	Kp	Ki
0.75	17	50	17
0.575	53	50	53
0.4	102	50	102
0.225	190	50	190
0.05	190	50	190
$y = -7836.7347x^3 + 12149.2711x^2 - 6109.2128x + 1064.6099$			

TABLE 5

TABLE RELATIONSHIP OF KD WITH DESTINED PERCENT OVERSHOOT

Destined Percent Overshoot	Kd	Kp	Ki
13.515	50	50	50
4.3195	100	50	50
2.7049	150	50	50
2.1702	200	50	50
1.9227	250	50	50
$y = 4.5605x^2 - 85.544x + 373.4689$			

TABLE 6

TABLE RELATIONSHIP OF KP AND DESTINED PERCENT OVERSHOOT

Destined Percent Overshoot	Kd	Kp	Ki
13.515	50	50	50
15.4059	50	100	50
19.5231	50	150	50
23.2569	50	200	50
26.4618	50	250	50
$y = -0.1581x^2 + 20.9542x - 196.8111$			

TABLE 7

TABLE FOR SUMMATIONS FOR 2ND ORDER POLYNOMIAL REGRESSION

Y	X_1	X_2	X_1^2	X_2^2	X_1X_2	X_1Y	X_2Y
…							
Summation (Σ)							

TABLE 8

2ND ORDER POLYNOMIAL REGRESSION

n	ΣX_1	ΣX_2	A	=	ΣY
ΣX_1	$\Sigma(X_1)^2$	$\Sigma(X_1X_2)$	B		ΣX_1Y
ΣX_2	$\Sigma(X_1X_2)$	$\Sigma(X_2)^2$	C		ΣX_2Y
Fitting Equation: $A + B * X_1 + C * X_2$					

TABLE 9

TABLE FOR SUMMATIONS FOR 3 INPUT LINEAR REGRESSION

Y	X_1	X_2	X_3	X_1^2	X_2^2	X_3^2	X_1X_1	X_1X_3	X_2X_3	X_1Y	X_2Y	X_3Y
…												
Summation (Σ)												

TABLE 10

MATRIX FOR SUMMATIONS FOR 3 INPUT LINEAR REGRESSION

n	ΣX_1	ΣX_2	ΣX_3	A		ΣY
ΣX_1	$\Sigma(X_1^2)$	$\Sigma(X_1X_2)$	$\Sigma(X_1X_3)$	B	=	ΣX_1Y
ΣX_2	$\Sigma(X_1X_2)$	$\Sigma(X_2^2)$	$\Sigma(X_2X_3)$	C		ΣX_2Y
$\Sigma X3$	$\Sigma(X_1X_3)$	$\Sigma(X_2X_3)$	$\Sigma(X_3^2)$	D		ΣX_3Y

TABLE 11

MULTIVARIABLE INPUT LINEAR REGRESSION

Outputs (Y)			Inputs (X)			Remarks
Kd	Kp	Ki	Rise	Overshoot	Settling	
1	5	1.3	5.0514	0	18.8623	Rise Time 5 to 0.1
15.5	14.7	3	2.5095	0	12.4115	
25	27	5	1.2303	0	9.9226	
325	650	105	0.1002	4.0692	0.6693	
30	0.67	0.27	43.1582	0.9322	55.0804	Percent Overshoot 1 to 0.1
15	0.83	0.43	26.1001	0.7747	33.8965	
20	0.23	0.26	40.6735	0.3513	53.6264	
25	0.18	0.2	52.8634	0.0918	71.9802	
101	100	128	0.3272	9.2012	5	Settling Time 5 to 0.1
205	46	35	0.2204	1.8565	3.0792	
320	53	80.55	0.1328	1.9999	1.6125	
215	197	3	0.1721	0	0.2968	

```
Kd = 1000; % change its value
Kp = 1000; % change its value
Ki = 1000; % change its value
num=[6.68*Kd 6.68*Kp 6.68*Ki];
den=[114.89 91.44+(6.68*Kd) 32+(6.68*Kp) (6.68*Ki)];
t=0:0.01:5;
step(num,den,t);
sys = tf(num,den);
s = stepinfo(sys) % shows the info of of rise time, settling time,
and overshoot
```

Figure. 7. MATLAB code to determine the input

For the MATLAB code, the original transfer function was used to determine the step response using a step function with a time parameter of 0 to 20 with a 0.01 interval. for the PID controller, initialization of inputs was first done which are rise, overshoot, and settling. Additional modifications to these variables are done based on the stepinfo results as shown in Figure 9. Due to inaccuracies, this is one of the probable treatments. However, certain input has a certain modifier and thus trial-and-error should be done. Afterwards, initialization of PID variables is done by Initialization of x and n values for the equation estimation via multivariable linear regression with 3 input variables and 1 output. The values used are those obtained in table 9. The values for kd1, kd2, and kd3 were obtained. kd1 comes from the multivariable linear regression, kd2 and kd3 are from polynomial regression, and the average of Kd1 to kd3 was obtained to fit more in the PID characteristic relationship. the same is done for kp and ki. After getting the values for PID Controller, the generation of the step response graph is done with the implementation of the modified block diagram. Pole Zero map and Transfer Function to State Space Representation were also simulated in Matlab as shown in the following figures.

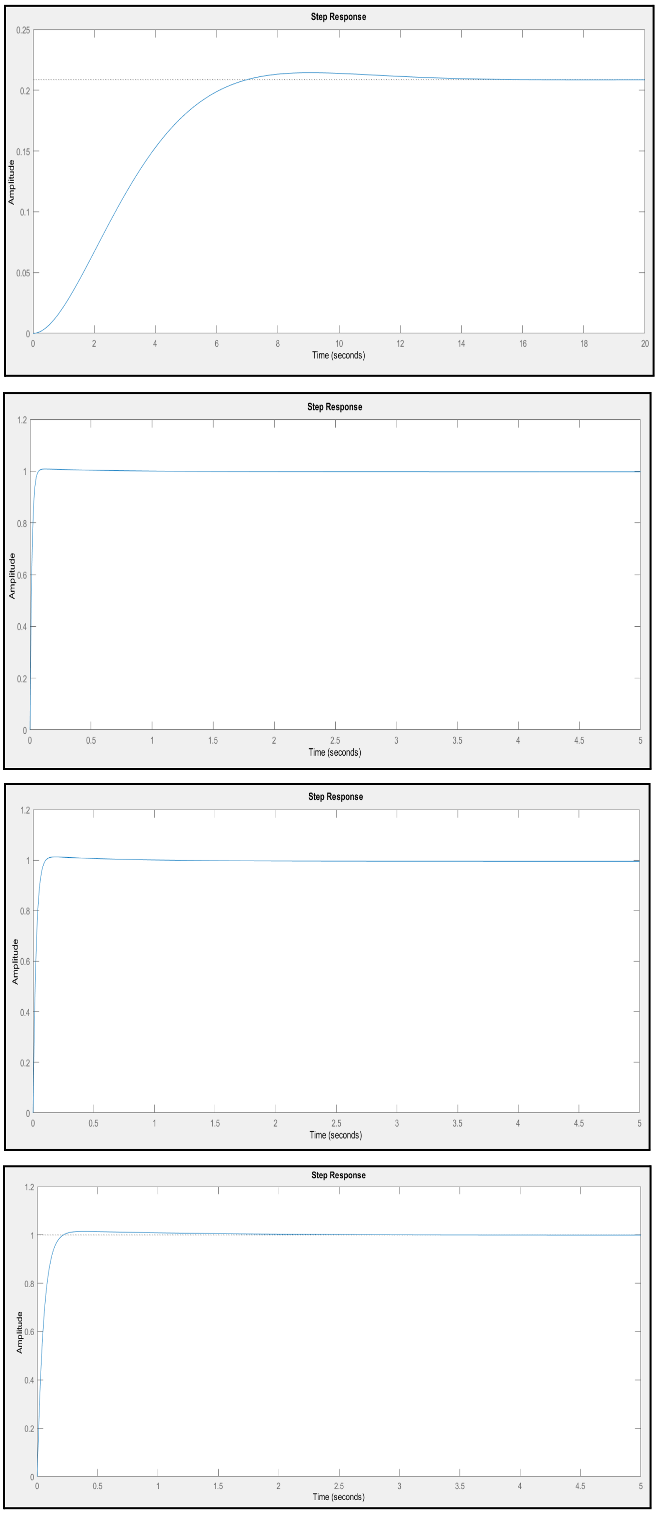


Figure 8. Step Response of (a) Original Transfer Function (first) (b) Sufficiently Accurate Result 1 (second) (c) Sufficiently Accurate Result 2 (third) (d) Sufficiently Accurate Result 3 (fourth)

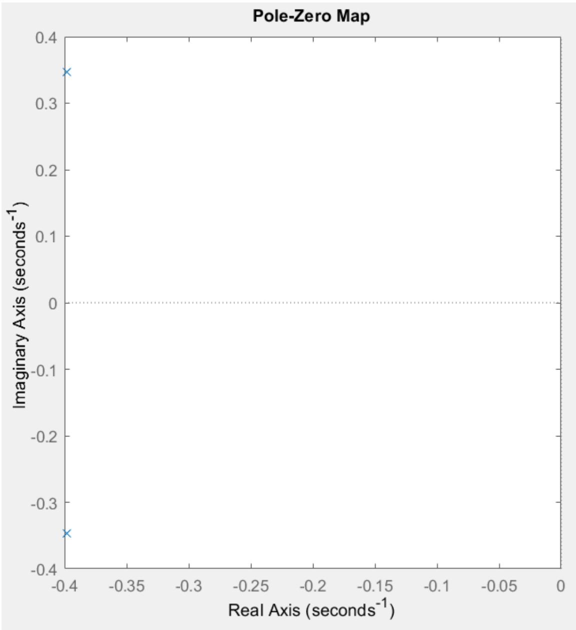
```
s = Rise Time    Percent Overshoot    Settling Time    Steady-State Error
s =

   0.108434378739751    1.421696863774646    0.174527150743875    0.035438453928316
```

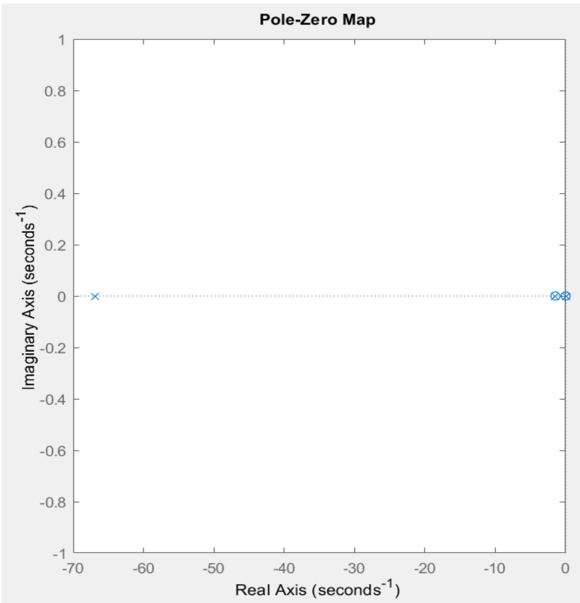
```
s = Rise Time    Percent Overshoot    Settling Time    Steady-State Error
s =

   0.031269953590701    1.076458575425066    0.051500661379724                  Inf
```

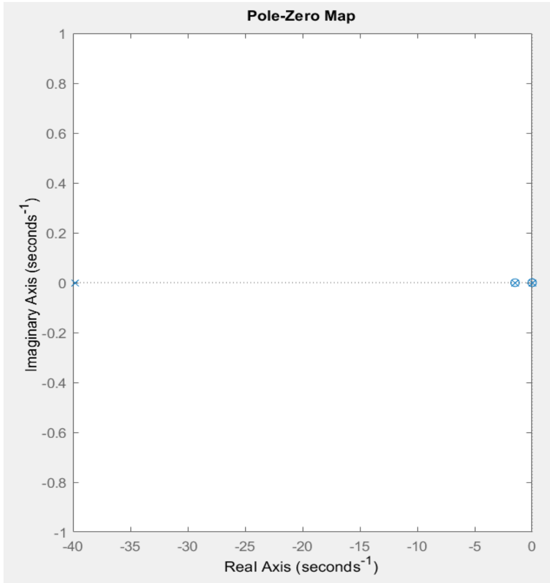
```
s = Rise Time    Percent Overshoot    Settling Time    Steady-State Error
s =

   0.050904911769387    1.724927334471005    0.080893328444680                  Inf
```

```
s = Rise Time    Percent Overshoot    Settling Time    Steady-State Error
s =

   0.108434378739751    1.421696863774646    0.174527150743875    0.035438453928316
```

Figure 9. Step Response Informations of (a) Original Transfer Function (first) (b) Sufficiently Accurate Result 1 (second) (c) Sufficiently Accurate Result 2 (third) (d) Sufficiently Accurate Result 3 (fourth)







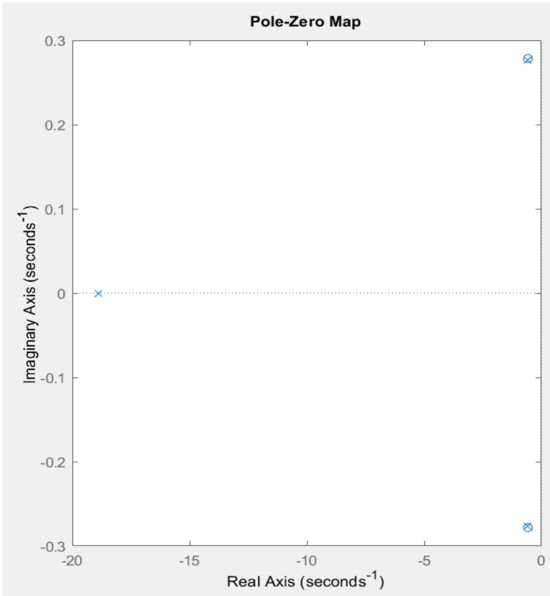


Figure 10. Pole Zero Map of (a) Original Transfer Function (first) (b) Sufficiently Accurate Result 1 (second) (c) Sufficiently Accurate Result 2 (third) (d) Sufficiently Accurate Result 3 (fourth)

```
ra =

[11489/100,  32]
[  2286/25,   0]
[        32,   0]


A =

  -0.795891722517190  -0.278527286970145
   1.000000000000000                   0


B =

     1
     0


C =

                   0   0.058142571155018


D =

     0
```

```
ra =

[                    11489/100, 3105825445678419/274877906944]
[8641431316314071/1099511627776,                            0]
[ 3105825445678419/274877906944,                            0]
[                             0,                            0]


A =

 -68.407485402753139 -98.345610148004724                   0
   1.000000000000000                   0                   0
                   0   1.000000000000000                   0


B =

     1
     0
     0


C =

  67.611593680235956  98.067082861034578                   0


D =

     0
```

```
ra =

[                    11489/100, 3806055959679251/549755813888]
[1304435775903067/274877906944,                            0]
[3806055959679251/549755813888,                            0]
[                            0,                            0]


A =

 -41.304810757660306 -60.259164940021876                   0
   1.000000000000000                   0                   0
                   0   1.000000000000000                   0


B =

     1
     0
     0


C =

  40.508919035143123  59.980637653051730                   0


D =

     0
```

```
ra =

[                                                 11489/100, 5666284022388659/2199023255552]
[                            5067797356400769/2199023255552,   496415167546553/549755813888]
[7053477599202037824688967538600091/2786051060287559567904079872000,                              0]
[                              496415167546553/549755813888,                              0]


A =

 -20.058903495975940 -22.427780037875714  -7.859464962369405
   1.000000000000000                   0                   0
                   0   1.000000000000000                   0


B =

     1
     0
     0


C =

  19.263011773458750  22.149252750905568   7.859464962369405


D =

     0
```

Figure 11. Stability and State-Space Representation of (a) Original Transfer Function (first) (b) Sufficiently Accurate Result 1 (second) (c) Sufficiently Accurate Result 2 (third) (d) Sufficiently Accurate Result 3 (fourth)

TABLE 12
SUFFICIENTLY ACCURATE TEST RESULTS

Sufficiently Accurate Test Results								
Modifier			Actual Value			Input Value (True)		
Rise	Overshoot	Settling	Rise	Overshoot	Settling	Rise	Overshoot	Settling
0	-45	0.6	0.0313	1.0765	0.0515	0.01	1	1
-0.13	-69	2	0.0509	1.7249	0.0809	0.15	0.5	0.01
-0.13	-11	0.691	0.1084	1.4217	0.1745	0.25	0.75	0.35

Despite the consideration of accuracy and interconnection of the input variables via polynomial regression and multiple linear regression respectively, inputting specific input values doesn't align with the output step response function. However, it is reasonable due to the summing up of errors from the estimation of fitting equations. Moreover, due to the fact that multiple estimations were averaged, the error may increase. Table 12 shows some of the results that align with the expected result which are the inputted ones.

Based on the testing of inputting values for the rise time, percent overshoot, and settling time, increasing one of the variables has a direct relationship with the output value, shown in stepinfo command from Matlab. Multivariable polynomial regression was done to ease up the process. However, the existence of error is inevitable.

Further mathematical treatments that have higher accuracy need to be done. Additional linear or polynomial regression with the modifier is recommended, however, nesting is not recommended similar to a polynomial regression for the modifier's other polynomial regression because this will result in complications. Matlab's transfer function to state space function output a result that is not in the controllable canonical form, however, this can easily be converted by flipping matrix A horizontally and vertically while matrix B and B are interchanged and transposed while matrix D remains the same.

IV. CONCLUSION

Differential equation modeling, which is based on the kind of motion being done in the system, is used to analyze mechanical systems. There are two kinds which are translational and rotational mechanical systems. There are some systems however that incorporate both types of motion resulting in a combined rotational and translational mechanical system. With the many applications of this type of system, a simulation was done to analyze the characteristics of a particular system where the transfer function and stability can be analyzed. In addition, applying a PID controller can add controllability to the system changing its characteristics like rise time, percent overshoot, and settling time. This study analyzed also the relationship of these characteristics to the PID controller where the characteristics are the input and the PID controller is the output. Using polynomial and linear regression, a program was made to do this. The results showed that the system was able to demonstrate sufficiently accurate results, however, due to the estimated fitting equations and limitations from the assumptions, errors were encountered. It is recommended that future research utilize more accurate mathematical models and measurements to mitigate the errors encountered.

ACKNOWLEDGMENT

The researchers would like to thank their professors/instructors Engr. Reggie Gustilo for the opportunity to conduct this project and for the term filled with fruitful lectures and learnings.

REFERENCES

[1] F. Axisa and P. Trompette, *Modelling of mechanical systems: Structural elements: Structural elements*. Woburn, MA: Butterworth-Heinemann, 2014.

[2] "Time Domain Specifications," *Tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/control_systems/control_systems_time_domain_specifications.htm. [Accessed: 26-Aug-2022].

[3] F. Golnaraghi and B. C. Kuo, *Automatic Control Systems*, 9th ed. Wiley, 2015.

[4] N. S. Nise, *Control Systems Engineering*, 7th ed. Nashville, TN: John Wiley & Sons, 2013.

[5] T. Watanabe, "Hand design—hybrid soft and hard structures based on human fingertips for dexterity," in *Human Inspired Dexterity in Robotic Manipulation*, Elsevier, 2018, pp. 115–147.

[6] "Chapter learning outcomes," *Njit.edu*. [Online]. Available: https://web.njit.edu/~mad29/refs/2ndorder_adfklhfw21.pdf. [Accessed: 26-Aug-2022].

[7] M. Rashid *et al.*, "Automated closed-loop insulin delivery: system components, performance, and limitations," in *Glucose Monitoring Devices*, Elsevier, 2020, pp. 293–326.

[8] R. Toulson and T. Wilmshurst, "An introduction to control systems," in *Fast and Effective Embedded Systems Design*, Elsevier, 2012, pp. 273–295.

[9] K. Jiao *et al.*, "System-level modeling of proton exchange membrane fuel cell," in *Water and Thermal Management of Proton Exchange Membrane Fuel Cells*, Elsevier, 2021, pp. 265–314.

[10] Pid, "Tuning for PID Controllers PID Controllers," *Mercer.edu*. [Online]. Available: http://faculty.mercer.edu/jenkins_he/documents/TuningforPIDControllers.pdf. [Accessed: 26-Aug-2022].

[11] "Introduction: PID Controller Design," *Umich.edu*. [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID. [Accessed: 26-Aug-2022].

[12] "Extras: Steady-State Error," *Umich.edu*. [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_Ess. [Accessed: 26-Aug-2022].

[13] "Extras: Type 1 Systems Examples," *Umich.edu*. [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_ess2. [Accessed: 26-Aug-2022].

APPENDIX

A. Main Matlab code

```
% 1. Original Transfer Function
num=6.68;
den=[114.89 91.44 32];
t=0:0.01:20;
step(num,den,t);

% 2. PID Controller
% 2.A. Initialization of Inputs
rise = 0.25; % Rise time
overshoot = 0.75; % Overshoot percentage
settling = 0.35; % Settling time

% Extra treatments for input variables to align
% with stepinfo results from 2.B. (modifiers chosen via
trial-and-error)
rise = rise - 0.13;
overshoot = overshoot - 11;
settling = settling + 0.691;

% 2.B. Initialization of PID variables
% Initialization of x and n values for the equation estimation
% via multivariable linear regression (with 3 input variables & 1
output)
nval = 12;
x1val = transpose([5.0514 2.5095 1.2303 0.1002 43.1582
26.1001 40.6735 52.8634 0.3272 0.2204 0.1328 0.1721]);
x2val = transpose([0 0 0 4.0692 0.9322 0.7747 0.3513 0.0918
9.2012 1.8565 1.9999 0]);
x3val = transpose([18.8623 12.4115 9.9226 0.6693 55.0804
33.8965 53.6264 71.9802 5 3.0792 1.6125 0.2968]);

% Equation estimation for Kd
syms x;
yval1 = transpose([1 15.5 25 325 30 15 20 25 101 205 320
215]);
eqny = triinputeqn(nval,yval1,x1val,x2val,x3val);
Kd1 = eqny(1) + (eqny(2)*rise) + (eqny(3)*overshoot) +
(eqny(4)*settling); % Multivariable linear regression
Kd2 =
double(subs((-7836.7347*(x^3))+(12149.2711*(x^2))-(6109.21
28*x)+1064.6099, x, settling)); % Eqn2 (Excel)
Kd3 = double(subs((4.5605*(x^2))-(85.544*x)+373.4686, x,
overshoot)); % Eqn3 (Excel)
% Average of Kd1-3 to fit more in the PID characteristic
relationship
Kd = max((Kd1 + Kd2 + Kd3) / 3, 0);

% Equation estimation for Kp
yval1 = transpose([5 14.7 27 650 0.67 0.83 0.23 0.18 100 46 53
197]);
eqny = triinputeqn(nval,yval1,x1val,x2val,x3val);
Kp1 = eqny(1) + (eqny(2)*rise) + (eqny(3)*overshoot) +
(eqny(4)*settling); % Multivariable linear regression
Kp2 =
double(subs((-440000*(x^3))+(330285.7143*(x^2))-(84114.285
7*x)+7593, x, rise)); % Eqn1 (Excel)
Kp3 = double(subs((-0.1581*(x^2))+(20.9542*x)-196.8111, x,
overshoot)); % Eqn4 (Excel)
% Average of Kp1-3 to fit more in the PID characteristic
relationship
Kp = max((Kp1 + Kp2 + Kp3) / 3, 0);

% Equation estimation for Ki
```

```
yval1 = transpose([1.3 3 5 105 0.27 0.43 0.26 0.2 128 35 80.55
3 ]);
eqny = triinputeqn(nval,yval1,x1val,x2val,x3val);
Ki1 = eqny(1) + (eqny(2)*rise) + (eqny(3)*overshoot) +
(eqny(4)*settling); % Multivariable linear regression
Ki2 =
double(subs((-440000*(x^3))+(330285.7143*(x^2))-(84114.285
7*x)+7593, x, rise)); % Eqn1 (Excel)
Ki3 =
double(subs((-7836.7347*(x^3))+(12149.2711*(x^2))-(6109.21
28*x)+1064.6099, x, settling)); % Eqn2 (Excel)
% Average of Kp1-3 to fit more in the PID characteristic
relationship
Ki = max((Ki1 + Ki2 + Ki3) / 3, 0);

% 2.C. Generation of Step Response Graph
num=[6.68*Kd 6.68*Kp 6.68*Ki];
den=[114.89 91.44+(6.68*Kd) 32+(6.68*Kp) (6.68*Ki)];
t=0:0.01:5;
step(num,den,t);
sys = tf(num,den);
s1 = [stepinfo(sys).RiseTime stepinfo(sys).Overshoot];
s2 = [stepinfo(sys).SettlingTime 1/(0.20875*Ki)];
fprintf('s = Rise Time\tPercent Overshoot\tSettling
Time\tSteady-State Error')
s = [s1 s2]

% 3. Routh Table
syms EPS
ra=routh(den, EPS)

% 4. Pole Zero Map
sys = tf(num,den);
h = pzplot(sys);

% 5. Transfer Function to State Space Representation
b = num;
a = den;
[A,B,C,D] = tf2ss(b,a)
```

B. Multiple variable linear regression

```
function [ans] = triinputeqn(n,y,x1,x2,x3)
mat1a = [y x1 x2 x3 (x1.^2) (x2.^2) (x3.^2) (x1.*x2) (x1.*x3)
(x2.*x3)];
mat1b = [(x1.*y) (x2.*y) (x3.*y)];
mat1 = [mat1a mat1b];
msum = sum(mat1([1:n], [1:13]));

mat2a = [n msum(2) msum(3) msum(4)];
mat2b = [msum(2) msum(5) msum(8) msum(9)];
mat2c = [msum(3) msum(8) msum(6) msum(10)];
mat2d = [msum(4) msum(9) msum(10) msum(7)];
mat2 = [mat2a; mat2b; mat2c; mat2d];
mat3 = [msum(1); msum(11); msum(12); msum(13)];
ans = linsolve(mat2, mat3);
end
```

C. Routh table

```
function RA=routh(poli,epsilon);

%ROUTH   Routh array.
%   RA=ROUTH(R,EPSILON) returns the symbolic Routh
array RA for
%   polynomial R(s). The following special cases are
```

```
considered:
%   1) zero first elements and 2) rows of zeros. All zero first
%   elements are replaced with the symbolic variable EPSILON
%   which can be later substituted with positive and negative
%   small numbers using SUBS(RA,EPSILON,...). When a row
of
%   zeros is found, the auxiliary polynomial is used.
%
%           Examples:
%
%           1) Routh array for s^3+2*s^2+3*s+1
%
%                       >>syms EPS
%                       >>ra=routh([1 2 3 1],EPS)
%                       ra =
%
%                         1.0000    3.0000
%                         2.0000    1.0000
%                         2.5000         0
%                         1.0000         0
%
%           2) Routh array for s^3+a*s^2+b*s+c
%
%                       >>syms a b c EPS;
%                       >>ra=routh([1 a b c],EPS);
%                       ra =
%
%                       [          1,          b]
%                       [          a,          c]
%                       [ (-c+b*a)/a,          0]
%                       [          c,          0]
%
%
%   Author:Rivera-Santos, Edmundo J.
%   E-mail:edmundo@alum.mit.edu
%

if(nargin<2),
            fprintf('\nError: Not enough input arguments given.');
            return
end

dim=size(poli);                     %get size of poli

coeff=dim(2);                                       %get number
of coefficients
RA=sym(zeros(coeff,ceil(coeff/2)));      %initialize symbolic
Routh array

for i=1:coeff,
            RA(2-rem(i,2),ceil(i/2))=poli(i); %assemble 1st and
2nd rows
end

rows=coeff-2;                       %number of rows that need
determinants
index=zeros(rows,1);                %initialize columns-per-row
index vector

for i=1:rows,
            index(rows-i+1)=ceil(i/2); %form index vector from
bottom to top
end

for i=3:coeff,                                      %go from 3rd
row to last
            if(all(RA(i-1,:)==0)),                  %row of zeros
                                    fprintf('\nSpecial Case: Row of
zeros detected.');
                                    a=coeff-i+2;
%order of auxiliary equation
                                    b=ceil(a/2)-rem(a,2)+1; %number
of auxiliary coefficients
                                    temp1=RA(i-2,1:b);
%get auxiliary polynomial
                                    temp2=a:-2:0;
%auxiliry polynomial powers
                                    RA(i-1,1:b)=temp1.*temp2;
%derivative of auxiliary
            elseif(RA(i-1,1)==0),                   %first
element in row is zero
                                    fprintf('\nSpecial Case: First
element is zero.');
                                    RA(i-1,1)=epsilon; %replace by
epsilon
            end
                                                %compute the Routh
array elements
            for j=1:index(i-2),
                        RA(i,j)=-det([RA(i-2,1)
RA(i-2,j+1);RA(i-1,1) RA(i-1,j+1)])/RA(i-1,1);
            end
end
```