Mechatronic Systems Engineering
School of Engineering Science
SIMON FRASER UNIVERSITY

MSE 312

Design Project – Robotic Arm

Control Model Simulation Report

Submitted to:	Dr. Farid Golnaraghi
Due Date:	July 19th 2015
Lab Section:	Thursday Morning

Submitted by:	Group 13, Summer 2015
Zheng Wang	301188444
Sohail Sangha	301186636
Rame Putris	301047157
Ane Tendo	301186203

Contents

1 Introduction 3
2 Design of control system 3
3 Design with PID controller 4
4 Simulation model 4
5 Simulation results 5
6 Conclusion 12
7 Appendix 13

List of Figures

Figure 1: controller design configuration 3
Figure 2: Top level simulation model 5
Figure 3: DC Motor Simulation "Subsystem" 5
Figure 4: Uncontrolled, closed loop system response 6
Figure 5: Unaided, closed loop system response longer time duration 6
Figure 6: Unaided, closed loop system dc motor voltage input 7
Figure 7: PID auto-tune GUI SimuLink 8
Figure 8: PID auto-tune configuration system response 8
Figure 9: PID auto-tune configuration PID output, dc motor voltage input [saturated] 9
Figure 10: system response, PID manual configuration P=4, I=0, D=0 10
Figure 11: system response, PID manual configuration P=4, I=0, D=2 10
Figure 12: system response, PID manual configuration P=4, I=0, D=3 11
Figure 13: system response, PID manual configuration P=4, I=0, D=2.45 11

List of Tables

Table 1: System performance comparison 12

1 Introduction

In order to apply our control knowledge to a practical problem, we designed a controller for our Robotic arm system. The Robotic arm system has a cantilever arm mounted to a motor shaft to create a simple robotic system conducting a pick and place operation as shown in figure 1. During the operations, a metal object is attracted by an electromagnet to the robot arm and moved from position $0^o$to a specified angular position with a specified overshoot and minimum overall time. Let us start the design of a controller for the system.

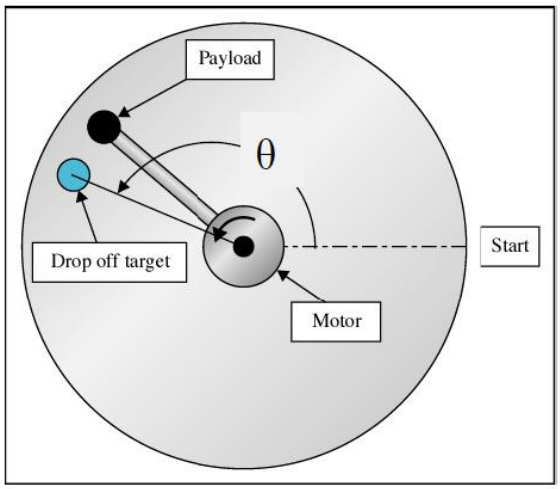

Figure 1: Control of a simple robotic arm and a payload.

2 Design of control system

As the measure of system performance, such as rise time, settling time and maximum overshoot, time-domain specifications are selected. Series (cascade) compensation is selected as our control system configuration. As shown in figure 2, the controller is placed in series with the controlled process.

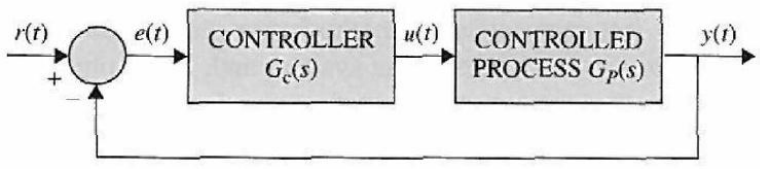

Figure 1: controller design configuration

After choosing a controller configuration, we need to choose a controller type that with proper selection of its element values, which are typically the coefficients of transfer functions making up the controller. While carrying out the design in the time domain, we also need to rely on the s-plane and the root loci as our design guidelines. A few time-domain characteristics are also focused on during design stages. Which are shown below :

1. Complex-conjugate poles of the closed-loop transfer function lead to a step response that is underdamped. If all system poles are real, the step response is overdamped. However, zeros of the closed-loop transfer function may cause overshoot even if the system is overdamped.
2. The response of a system is dominated by those poles closest to the origin in the s-plane. Transients due to those poles farther to the left decay faster.
3. The farther to the left in the s-plane the system's dominant poles are, the faster the system will respond and the greater its bandwidth will be.
4. The farther to the left in the s-plane the system's dominant poles are, the more expensive it will be and the larger its internal signals will be. While this can be justified analytically, it is obvious that striking a nail harder with a hammer drives the nail in faster but requires more energy per strike. Similarly, a sports car can accelerate faster, but it uses more fuel than an average car.
5. When a pole and zero of a system transfer function nearly cancel each other, the portion of the system response associated with the pole will have a small magnitude.
6. Time-domain and frequency-domain specifications are loosely associated with each other. Rise time and bandwidth are inversely proportional. Larger phase margin, larger gain margin, and lower M_r will improve damping.

3 Design with PID controller

A PID controller consists of a PI portion connected in cascade with a PD portion. The transfer function of PID controller is shown below:

$$G_c(s) = K_P + K_D s + \frac{K_I}{s} = (1 + K_{D1}s)\left(K_{P2} + \frac{K_{I2}}{s}\right)$$

By equating both side of the transfer function we have

$$K_P = K_{P2} + K_{D1}K_{I2}$$
$$K_D = K_{D1}K_{P2}$$
$$K_I = K_{I2}$$

We select the value of K_{D1} to achieve a portion of the desired relative stability, which is measured by the maximum overshoot. Parameters K_{I2} and K_{P2} are also selected for our PID controller to satisfy the relative stability.

4 Simulation model

The simulation model was designed in two distinct sections, Simulink model and a matlab script, the SimuLink model (given in figures below) was used to ease the development of frequency domain model of the system, and the matlab script (given in Appendix) provided easier grouping of all required variables and neater data plots.

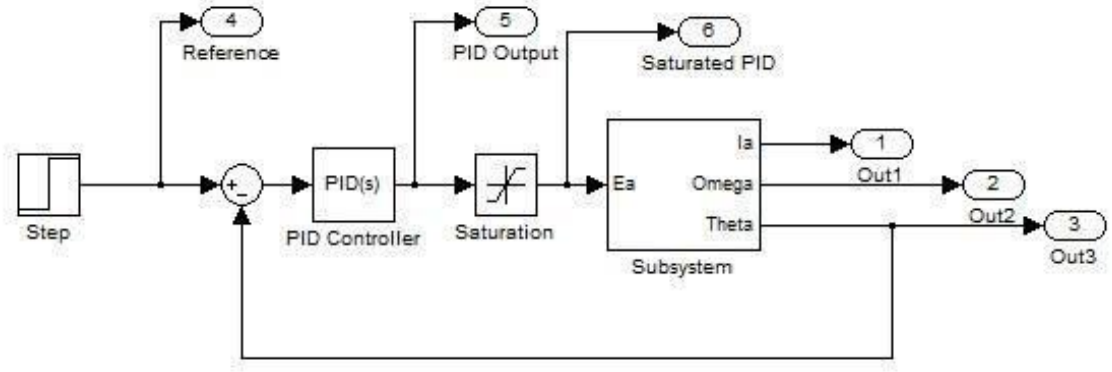

Figure 2: Top level simulation model

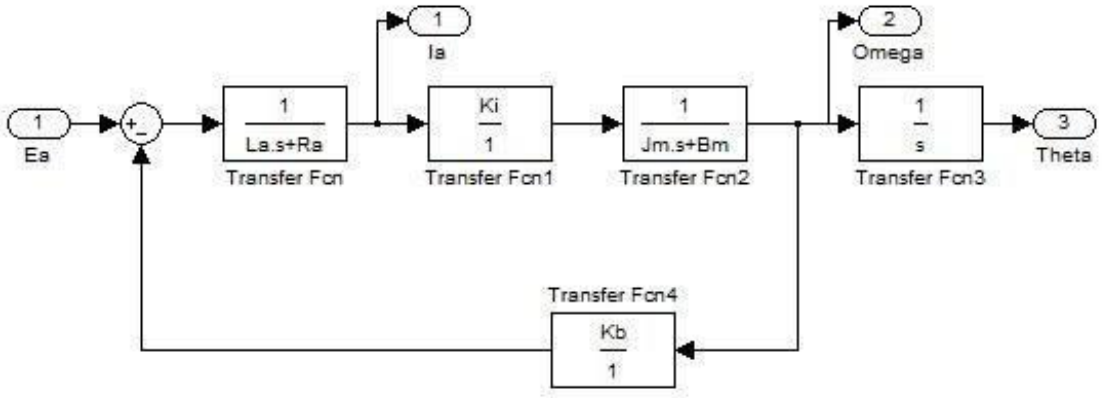

Figure 3: DC Motor Simulation "Subsystem"

5 Simulation results

For the consistency of the study, all the simulation configurations were given three different step inputs starting at 1 second with saturation at the PID output enabled for accurate results. The set initial position for the system was zero degrees while the final step value of reference inputs was 30, 90, and 120 degrees.

First step towards the eventual tuning of the system was to observe the controller unaided response of the system to capture the extent of performance enhancement offered by any control strategy. This was achieved by setting the PID controller parameters as 1 for proportional and zero for all other. Following figures display the data captured for uncontrolled run of the system.

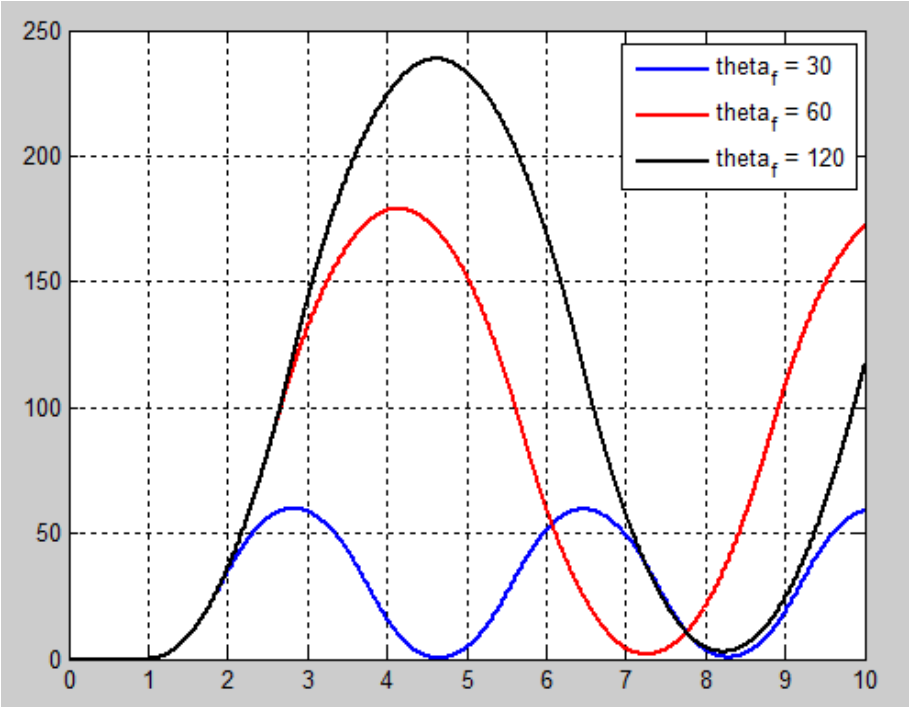

Figure 4: Uncontrolled, closed loop system response

In the figure above, the system is being supplied with reference input without being aided by a controller. A longer run for the system is given in the following image.

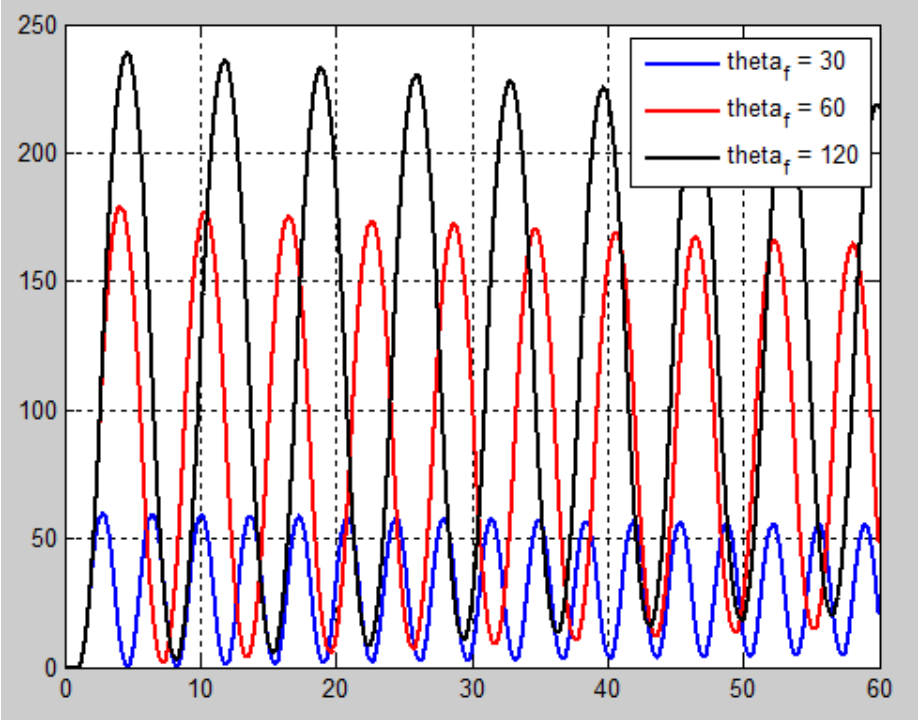

Figure 5: Unaided, closed loop system response longer time duration

From the preceding images it is clear that while the system is inherently stable, due to very low viscous effects the settling time of the system is incredibly low, thus warranting the need for a controller. Further clear picture can be formed by analyzing the voltage input to the system in the following figure.

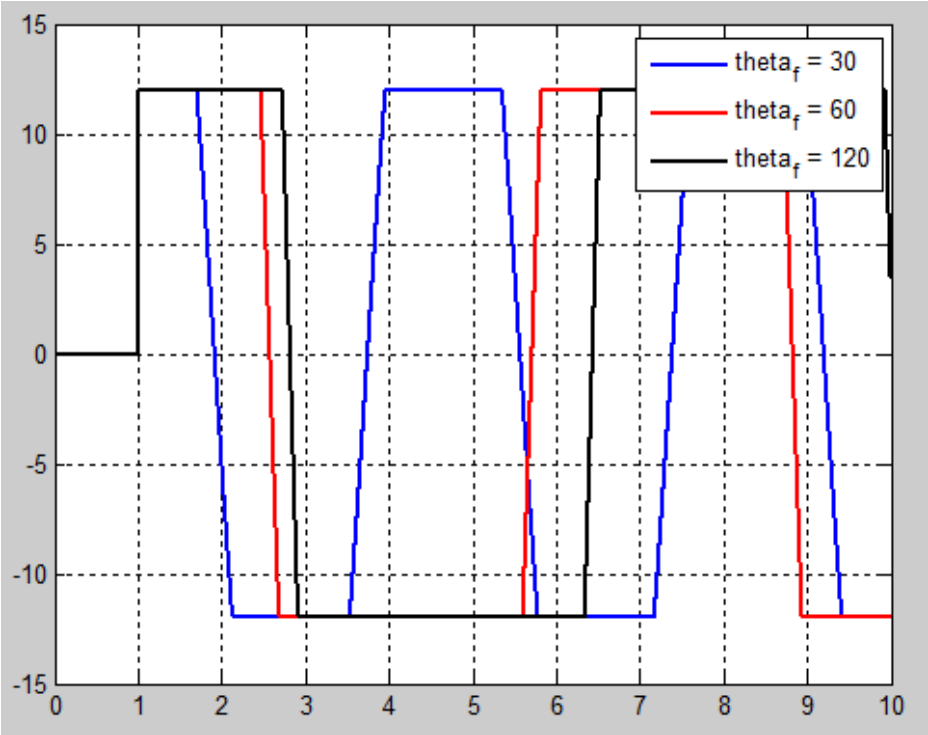

Figure 6: Unaided, closed loop system dc motor voltage input

From the above figure we can see that the non-linearity of the system plays important role in the observed behavior. Due to an absence of any control, the system is spending majority of its time saturated in maximum voltages and thus not able to oppose the stored energy in the system.

To design a system with desired parameters the most standard practice employed across industry is to use a PID controller. Given the design power offered by Simulink a PID controller can be autotuned by Simulink while providing the designer with comprehensive performance information.

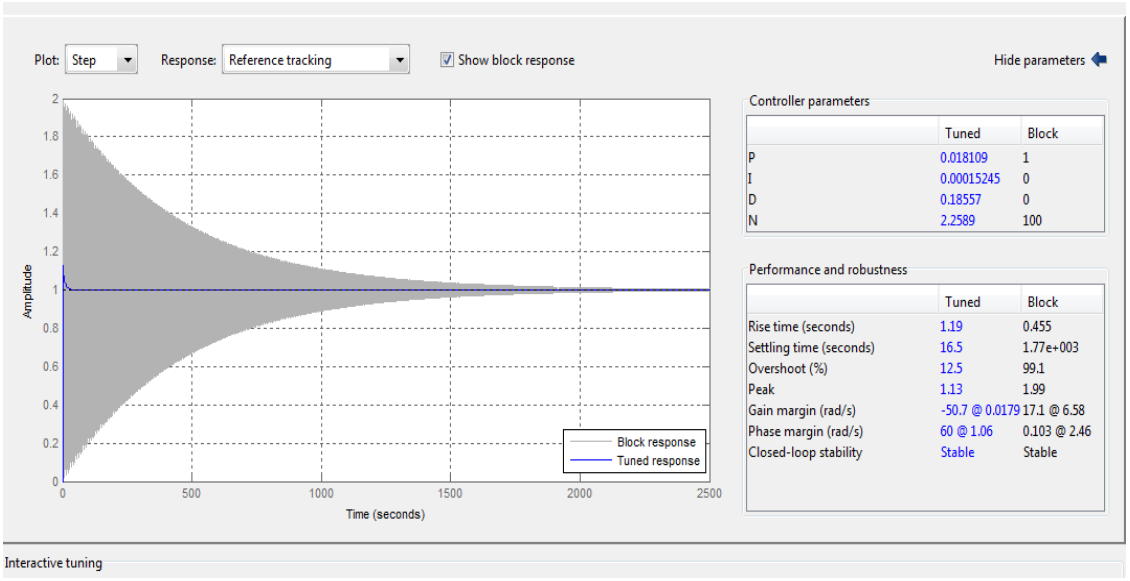

Figure 7: PID auto-tune GUI SimuLink

In the image above Simulink's PID autotune feature is being used to extract the desired performance from the system. Although, It can be clearly seen that for the given system autotune is not able to generate the desired parameters, this problem is exacerbated further by the inability of PID to deal with non-linear systems. This problem is better demonstrated in the following figures.

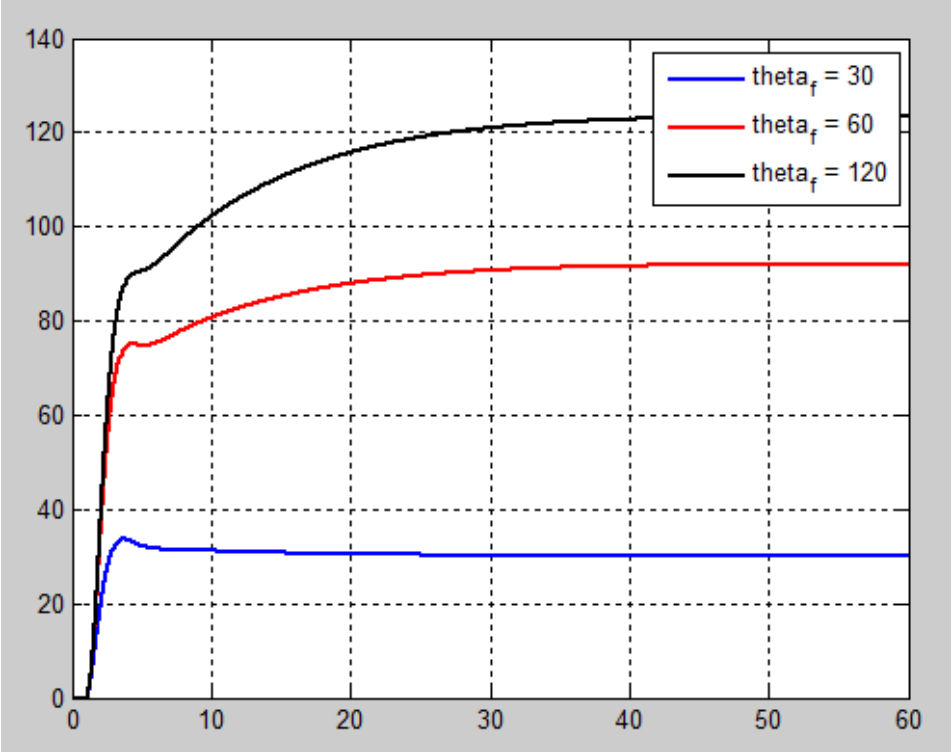

Figure 8: PID auto-tune configuration system response

In the above figure it can be seen that the control strategy is very inconsistent across the range of inputs. While the system is able to perform very well for smaller changes in degrees it is not so for the larger step values. This phenomenon can be better explained with the help of input voltage being sent into the system.

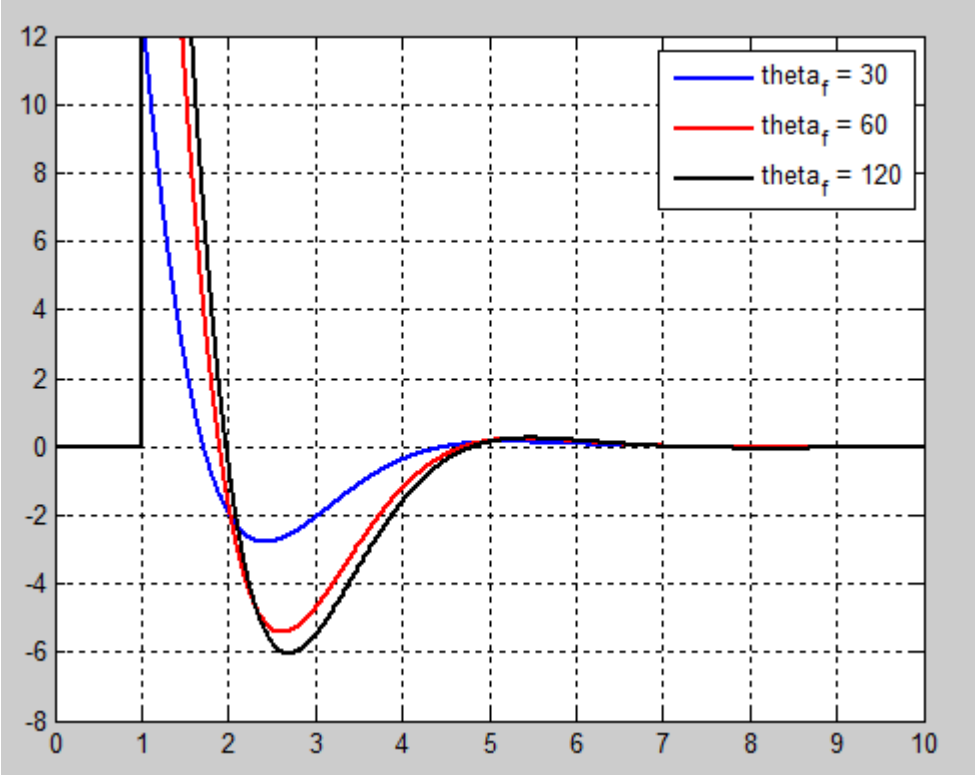

Figure 9: PID auto-tune configuration PID output, dc motor voltage input [saturated]

It can be seen above that for smaller step angles the PID controller is able to perform within the linearly defined boundaries of the system, while for larger step values the controller starts to saturate the system and unable to deliver the expected performance.

To overcome the problem of limited performance control offered by PID autotune, in the next step the controller was programmed with brute force strategy employed industry wide for tuning PID controllers manually.

1. Set proportional gain to a high enough value to make the system oscillate about the desired reference input.
2. Increase differential iteratively to obtain desired overshoot.
3. If system shows error after settling, set integral iteratively.

In the following figures the result from manual training of PID system is shown.

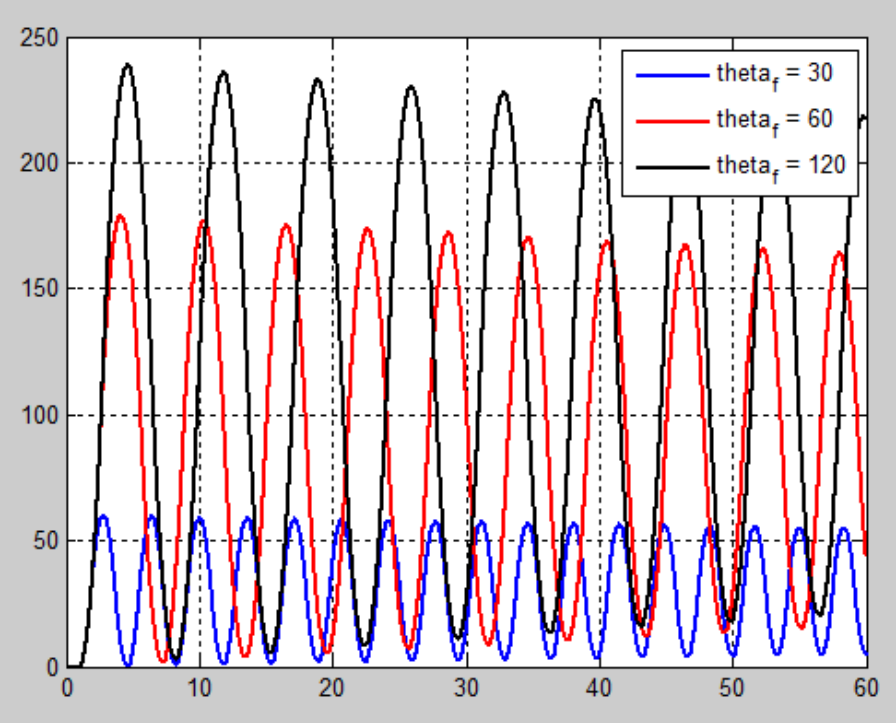

Figure 10: system response, PID manual configuration P=4, I=0, D=0

Above, response of the system for P = 4, I = 0, D = 0.

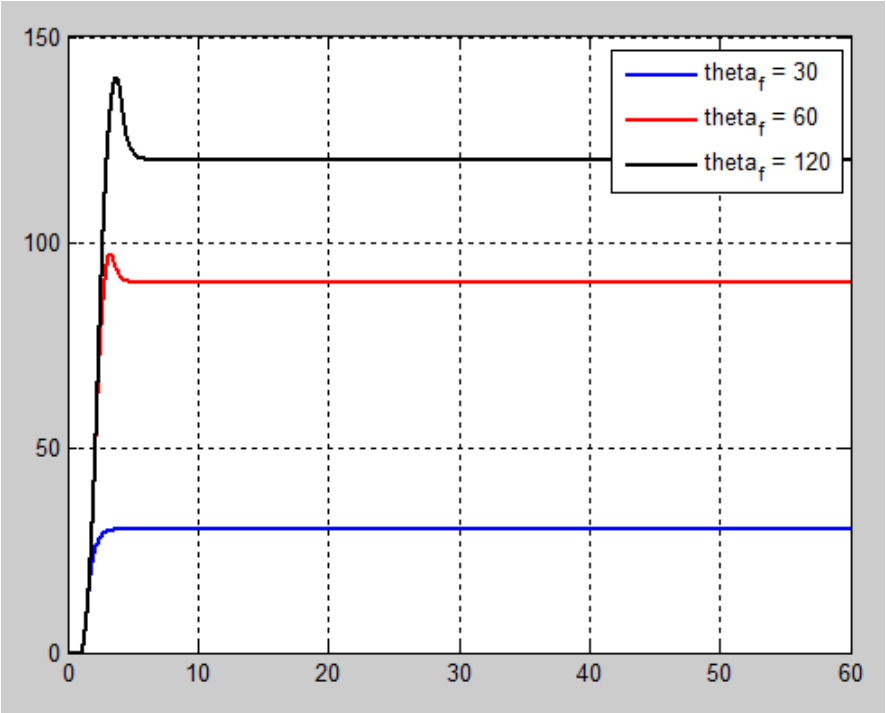

Figure 11: system response, PID manual configuration P=4, I=0, D=2

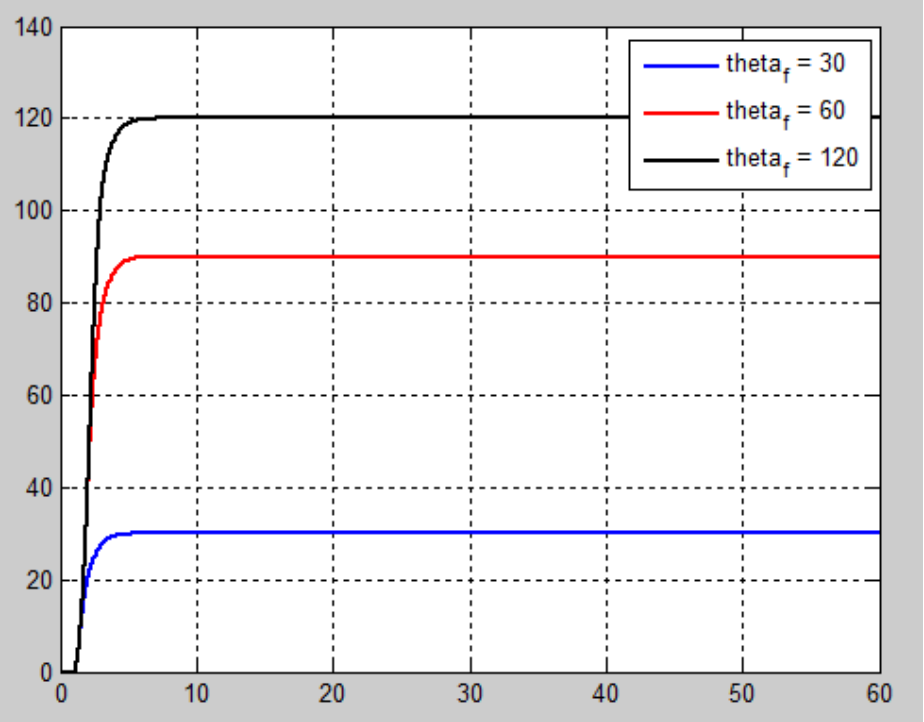

Figure 12: system response, PID manual configuration P=4, I=0, D=3

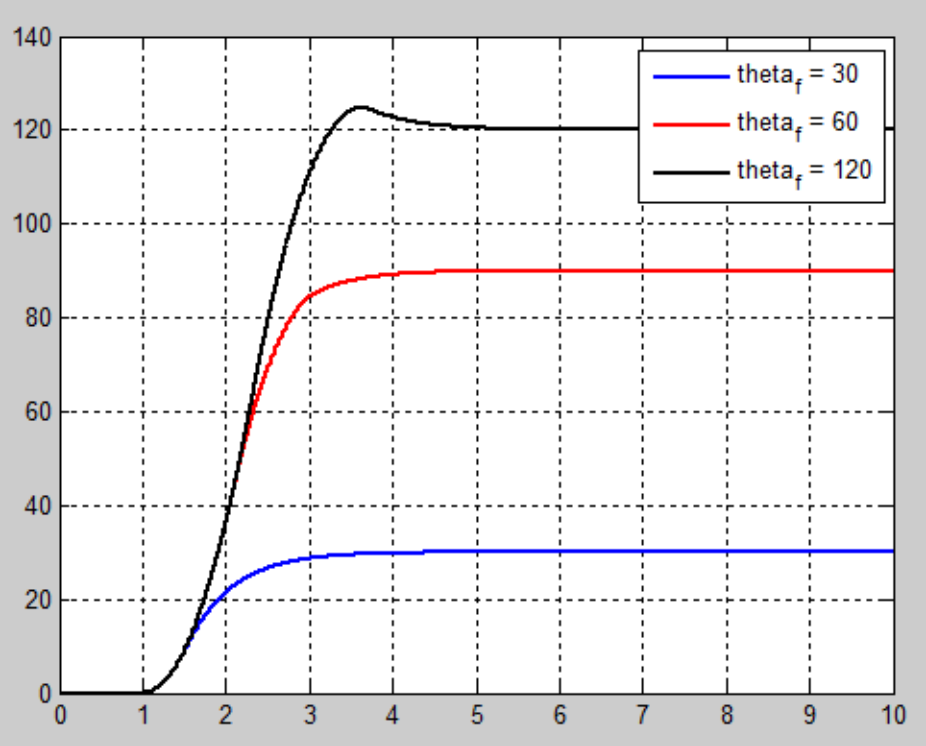

Figure 13: system response, PID manual configuration P=4, I=0, D=2.45

Above, response of the system for P = 4, I = 0, D = 2.45

It can be easily verified even from the above images that the system has no settling error thus does not warrant a need for inclusion of an Integral term. This may not hold true for the actual system because of non-linearity such as dead-zone. To overcome any un-foreseeable problem the best strategy is to use the found Proportional and Differential gains while manually increasing the integral term for the real life system to achieve desired outcome.

Following is a performance comparison for the various strategies implemented above.

	PID Configuration:			Settling Time [s] /Overshoot [%] for Step Input					
Description	**P**	**I**	**D**	**30**		**90**		**120**	
Unaided	1	0	0	> 10	> 100	> 10	> 100	> 10	> 100
Auto Tune	0.018	.0002	0.18	16.17	12.13	69.54	2.14	102.65	2.78
Manual	4	0	0	> 10	> 100	> 10	> 100	> 10	> 100
Manual	4	0	2	1.99	0	3.03	8.13	3.93	16.68
Manual	4	0	3	3.00	0	4.37	0	4.40	0
Manual	4	0	2.45	3.45	0	3.60	0	4.07	3.97

Table 1: System performance comparison

From the table above it can be readily said that the manual tuning of the PID controller offers great performance enhancement.
Further information obtained from the above table is the remarkable change in the settling time and overshoot for different step inputs for different differential gains. This can be kept in mind to develop a more intelligent strategy for PID control, where for smaller step input a smaller differential value can be used to decrease settling time, while the converse can be done to achieve better performance for higher step inputs.

6 Conclusion

The Robotic Arm project control segment involves the design of a controller simulation model with the utilization of a PID controller. Controller unaided response is observed and compared with auto-tuned PID controller, and manual tuning of PID controller with brute force. From the simulation data, we found that manual tuning of PID controller gives best enhancement of system performance; settling time and overshoot are also different for different step inputs of different differential gains.

7 Appendix

Maltab code plugin for Simulink model

```
clc
clear all;
close all;

%paramters defined here without any particular use in the script are used
%by the simulink model

%motor simulation parameters
max_Ea = 12;                        %max voltage
min_Ea = -12;                       %min voltage

Ra = 2.23;                          %Coil resistance [ohm]
La = 0.264*(10^-3);                 %Coil inductance [Henry]
Ki = 24.3*(10^-3);                  %Current to Torque proportionality factor [Nm/A]
Jm = (41.4 + 1030792)*(10^-7);      %Inertia motor rotor + arm [kg/m^2]
Bm = (1/36.1)*(10^-3)*60/(2*pi);    %damping coefficient [Nm/rad/s]
Kb = (1/394)*60/(2*pi);             %Back EMF factor

%PID controller parameters
P = 0.018;
I = 0.00015;
D = 0.18557;
N = 2.2589;

feedback_gain = 180/pi;

%setting system step input
theta_init = 0;
theta_final = 90;
step_time = 1;

%simulation parameters
run_time = 200;

%running model
[time_vec, state, output] = sim('motor_sim');
info = stepinfo(output(:,3), time_vec, theta_final);
disp(info)

%display results
plot(time_vec, output);
legend('current', 'omega', 'Theta', 'step input', 'pid out', 'Ea');
axis([0 run_time min_Ea max_Ea]);
grid on;
```