Simulation of Engineering System 3 (ENG3036)

Assignment Report Template Part 1

Student Name: **Elsie Msalila** Student Number: **2778939M**

Assignment Topic: Position Control System for a Robotic Arm

1.0 Introduction for Part 1

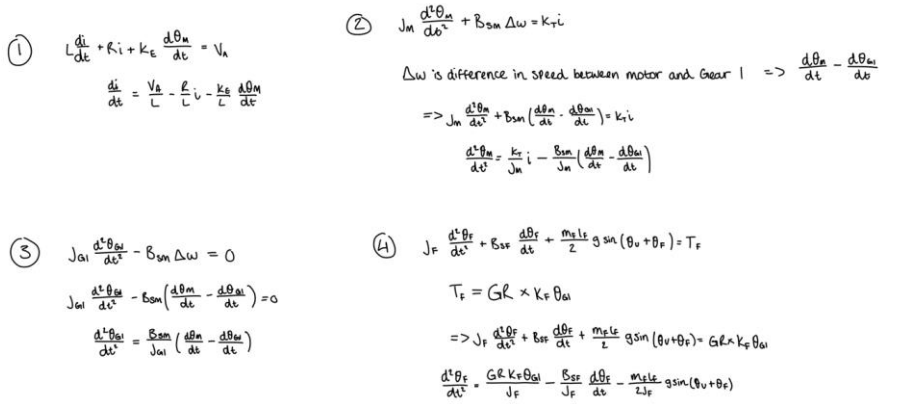
Position control systems are a fundamental component of robotic manipulators, where precise control of joint angles is essential for achieving accurate and reliable movement. The system is designed to ensure the forearm moves to a desired angular position (θ_F) by comparing the actual position (θ_M) to the desired position (θ_{Ref}) which are measured using sensors. The difference between these is known as the error ($\Delta\theta$) and tuning of gains in the system are used to ensure stable and precise movement.

The purpose of this assignment is to develop and simulate a model of a position control system for the elbow joint of a robotic arm. This involves deriving a state space model of the system along with its state space variables, implementing them in a MATLAB script and validating the results using a block diagram model constructed in Simulink.

<u>Mathematical Modelling & Continuous Time Simulation</u>

1.1 State Space Model Derivation

To derive the state space model of the robot arm system the dynamic equations given in the lab sheet were converted into the reduced form (figure 1)



① $L\frac{di}{dt} + Ri + K_E\frac{d\theta_M}{dt} = V_A$

$\frac{di}{dt} = \frac{V_A}{L} - \frac{R}{L}i - \frac{K_E}{L}\frac{d\theta_M}{dt}$

② $J_M\frac{d^2\theta_M}{dt^2} + B_{SM}\Delta\omega = K_Ti$

$\Delta\omega$ is difference in speed between motor and Gear 1 $\Rightarrow \frac{d\theta_M}{dt} - \frac{d\theta_{G1}}{dt}$

$\Rightarrow J_M\frac{d^2\theta_M}{dt^2} + B_{SM}\left(\frac{d\theta_M}{dt} - \frac{d\theta_{G1}}{dt}\right) = K_Ti$

$\frac{d^2\theta_M}{dt^2} = \frac{K_T}{J_M}i - \frac{B_{SM}}{J_M}\left(\frac{d\theta_M}{dt} - \frac{d\theta_{G1}}{dt}\right)$

③ $J_{G1}\frac{d^2\theta_{G1}}{dt^2} - B_{SM}\Delta\omega = 0$

$J_{G1}\frac{d^2\theta_{G1}}{dt^2} - B_{SM}\left(\frac{d\theta_M}{dt} - \frac{d\theta_{G1}}{dt}\right) = 0$

$\frac{d^2\theta_{G1}}{dt^2} = \frac{B_{SM}}{J_{G1}}\left(\frac{d\theta_M}{dt} - \frac{d\theta_{G1}}{dt}\right)$

④ $J_F\frac{d^2\theta_F}{dt^2} + B_{SF}\frac{d\theta_F}{dt} + \frac{m_Fl_F}{2}g\sin(\theta_U + \theta_F) = T_F$

$T_F = GR \times K_F\theta_{G1}$

$\Rightarrow J_F\frac{d^2\theta_F}{dt^2} + B_{SF}\frac{d\theta_F}{dt} + \frac{m_Fl_F}{2}g\sin(\theta_U + \theta_F) = GR \times K_F\theta_{G1}$

$\frac{d^2\theta_F}{dt^2} = \frac{GRK_F\theta_{G1}}{J_F} - \frac{B_{SF}}{J_F}\frac{d\theta_F}{dt} - \frac{m_Fl_F}{2J_F}g\sin(\theta_U + \theta_F)$

Figure 1 - converting dynamic equations to reduced form

Seven state variables were then assigned to the lower order derivatives in the reduced form equations:

$$x_1 = i,\ x_2 = \theta_M,\ x_3 = \dot{\theta}_M,\ x_4 = \theta_{G_1},\ x_5 = \dot{\theta}_{G_1},\ x_6 = \theta_F,\ x_7 = \dot{\theta}_F$$

Taking the derivative of each state variable gives:

$$\dot{x}_1 = \frac{di}{dt},\quad \dot{x}_2 = x_3,\ \dot{x}_3 = \ddot{\theta}_M,\ \dot{x}_4 = x_5,\ \dot{x}_5 = \ddot{\theta}_{G_1},\ \dot{x}_6 = x_7,\ \dot{x}_7 = \ddot{\theta}_F$$

Finally, the reduced form equations and the state derivatives were substituted into the differentiated state variables which gives us a set of equations which make up the state space model. These can then be implemented into MATLAB to create a mathematical model of the system.

$$\dot{x}_1 = \frac{V_A}{L} - \frac{R}{L}x_1 - \frac{K_E}{L}x_3\ \ [1]$$

$$\dot{x}_2 = x_3\ \ [2]$$

$$\dot{x}_3 = \frac{K_T}{J_M}x_1 - \frac{B_{SM}}{J_M}(x_3 - x_5)\ [3]$$

$$\dot{x}_4 = x_5\ \ [4]$$

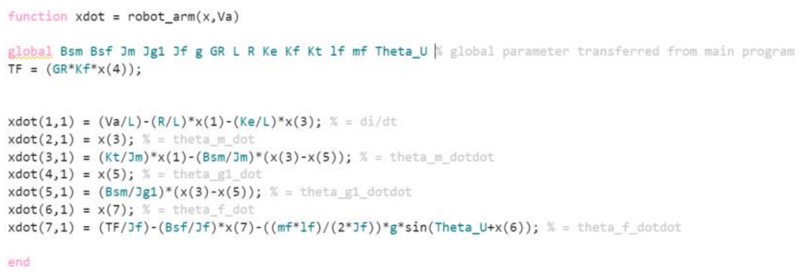
$$\dot{x}_5 = \frac{B_{SM}}{J_{G_1}}(x_3 - x_5)\ \ [5]$$

$$\dot{x}_6 = x_7\ \ [6]$$

$$\dot{x}_7 = \frac{GRK_F}{J_F}x_4 - \frac{B_{SF}}{J_F}x_7 - \frac{m_Fl_F}{2J_F}g\sin(\theta_U + x_6)\ \ [7]$$

1.2 MATLAB Model Function

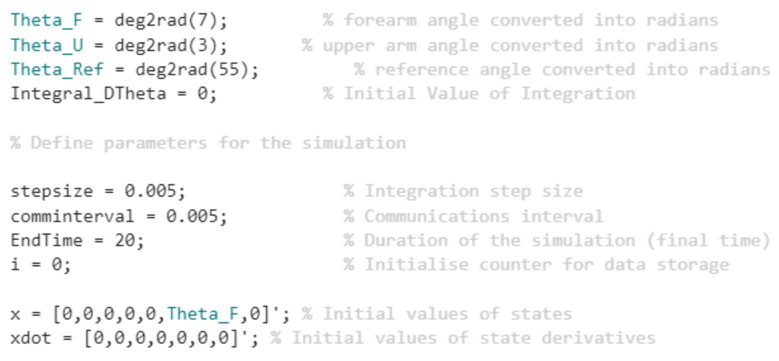
As previously mentioned, the state space model was implemented within a MATLAB script to model the robot arm system (figure 2). The global variables are assigned using the values given in Appendix A of the lab sheet. Implementing these as global variables allows their values to be shared between different functions.



```
function xdot = robot_arm(x,Va)

global Bsm Bsf Jm Jg1 Jf g GR L R Ke Kf Kt lf mf Theta_U % global parameter transferred from main program
TF = (GR*Kf*x(4));


xdot(1,1) = (Va/L)-(R/L)*x(1)-(Ke/L)*x(3); % = di/dt
xdot(2,1) = x(3); % = theta_m_dot
xdot(3,1) = (Kt/Jm)*x(1)-(Bsm/Jm)*(x(3)-x(5)); % = theta_m_dotdot
xdot(4,1) = x(5); % = theta_g1_dot
xdot(5,1) = (Bsm/Jg1)*(x(3)-x(5)); % = theta_g1_dotdot
xdot(6,1) = x(7); % = theta_f_dot
xdot(7,1) = (TF/Jf)-(Bsf/Jf)*x(7)-((mf*lf)/(2*Jf))*g*sin(Theta_U+x(6)); % = theta_f_dotdot

end
```

Figure 2 - Robot arm model script

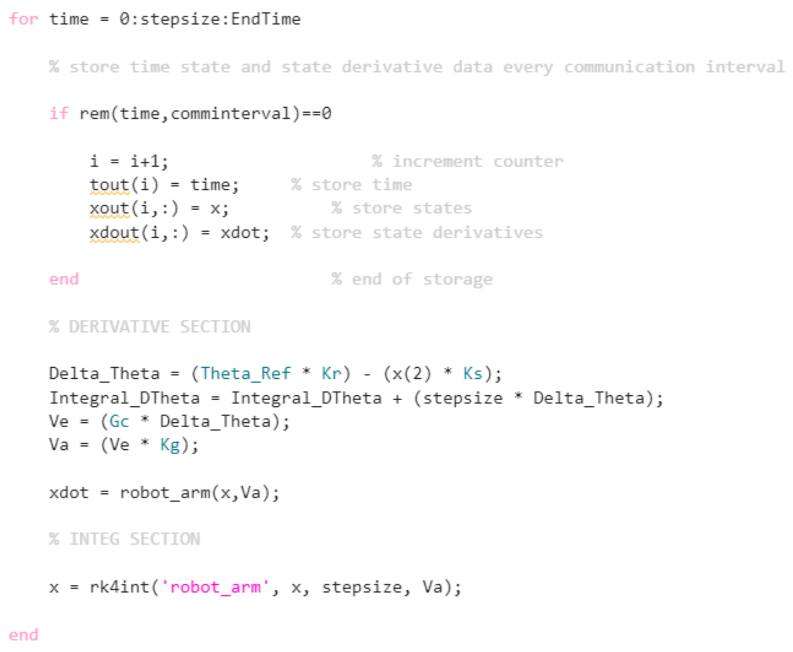
The initial conditions, initial states and simulation parameters are then defined (figure 3). The initial conditions are once again taken from the lab sheet. These values are converted from degrees into radians as this is the form MATLAB functions use.



```
Theta_F = deg2rad(7);          % forearm angle converted into radians
Theta_U = deg2rad(3);        % upper arm angle converted into radians
Theta_Ref = deg2rad(55);           % reference angle converted into radians
Integral_DTheta = 0;           % Initial Value of Integration

% Define parameters for the simulation

stepsize = 0.005;               % Integration step size
comminterval = 0.005;           % Communications interval
EndTime = 20;                   % Duration of the simulation (final time)
i = 0;                          % Initialise counter for data storage

x = [0,0,0,0,0,Theta_F,0]'; % Initial values of states
xdot = [0,0,0,0,0,0,0]'; % Initial values of state derivatives
```

Figure 3 - Defining initial conditions, value of states and simulations parameters

The simulation is processed through the dynamic segment (figure 4). Here the data is stored using the ‘tout’, ‘xout’ and ‘xdotout’ arrays and then to model is evaluated in the derivative section (figure 4).

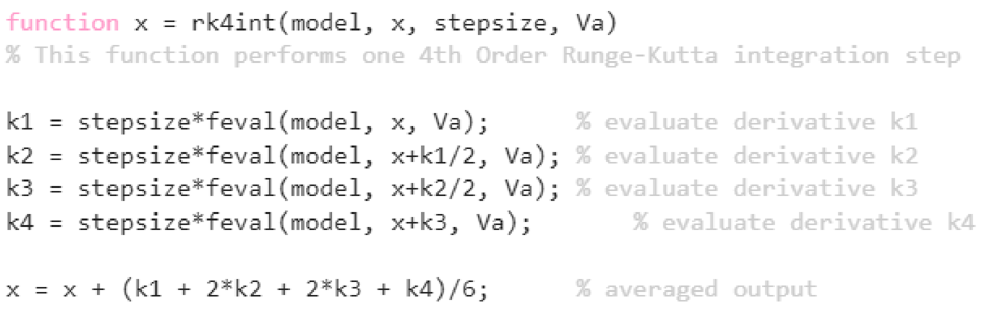


```
for time = 0:stepsize:EndTime

    % store time state and state derivative data every communication interval

    if rem(time,comminterval)==0

        i = i+1;                    % increment counter
        tout(i) = time;     % store time
        xout(i,:) = x;          % store states
        xdout(i,:) = xdot;  % store state derivatives

    end                         % end of storage

    % DERIVATIVE SECTION

    Delta_Theta = (Theta_Ref * Kr) - (x(2) * Ks);
    Integral_DTheta = Integral_DTheta + (stepsize * Delta_Theta);
    Ve = (Gc * Delta_Theta);
    Va = (Ve * Kg);

    xdot = robot_arm(x,Va);

    % INTEG SECTION

    x = rk4int('robot_arm', x, stepsize, Va);

end
```

Figure 4 - Dynamic segment

1.3 Initial Conditions, Numerical Integration Solver & Step-Size Selection

4TH Order Runge-Kutta was chosen as the integration method for the simulation. This method approximates the first 5 terms of the Taylor's Series by making 4 predictions of the state derivatives of the time interval. 4TH Order Runge-Kutta offers a food balance between accuracy and computational efficiency which is ideal for many engineering applications. It can be implemented in MATLAB as shown in figure 5.



```
function x = rk4int(model, x, stepsize, Va)
% This function performs one 4th Order Runge-Kutta integration step

k1 = stepsize*feval(model, x, Va);      % evaluate derivative k1
k2 = stepsize*feval(model, x+k1/2, Va); % evaluate derivative k2
k3 = stepsize*feval(model, x+k2/2, Va); % evaluate derivative k3
k4 = stepsize*feval(model, x+k3, Va);       % evaluate derivative k4

x = x + (k1 + 2*k2 + 2*k3 + k4)/6;      % averaged output
```

Figure 5 - 4TH order Runge-Kutta script

To determine the step size (figure 3), each state space equation was evaluated to determine a suitable step size for that equation. Subsequently, whichever equation yielded the lowest time constant was used as the step size in the simulation as this step size will be sufficient for simulating all the state equations. Equation 3 gave the lowest time constant and so this was used (figure 5).

$$s^2\theta_{G1}(s) = \frac{B_{sm}}{J_{G1}} s\theta_M(s) - \frac{B_{sm}}{J_{G1}} s\theta_{G1}(s)$$

$$\theta_{G1}(s)\left(s + \frac{B_{sm}}{J_{G1}}\right) = \frac{B_{sm}}{J_{G1}}\theta_M(s)$$

$$\frac{\theta_{G1}(s)}{\theta_M(s)} = \frac{B_{sm}/J_{G1}}{s + B_{sm}/J_{G1}}$$

$$\frac{X(s)}{U(s)} = \frac{b}{s+a} \Rightarrow a = \frac{B_{sm}}{J_{G1}}$$

$$\tau = \frac{1}{a} \Rightarrow \tau = \frac{1}{0.02/0.001}$$

$$= \frac{1}{20}$$

$$= 0.05\,s$$

Figure 6 - step size calculation

The ideal step size is given by $h = \frac{\tau}{10}$. Hence, $h = \frac{0.05}{10}$ giving a fixed step size of 0.005s.

1.4 Simulation Responses and Analysis

Each of the seven state variables were plotted against time as shown in figure 7.

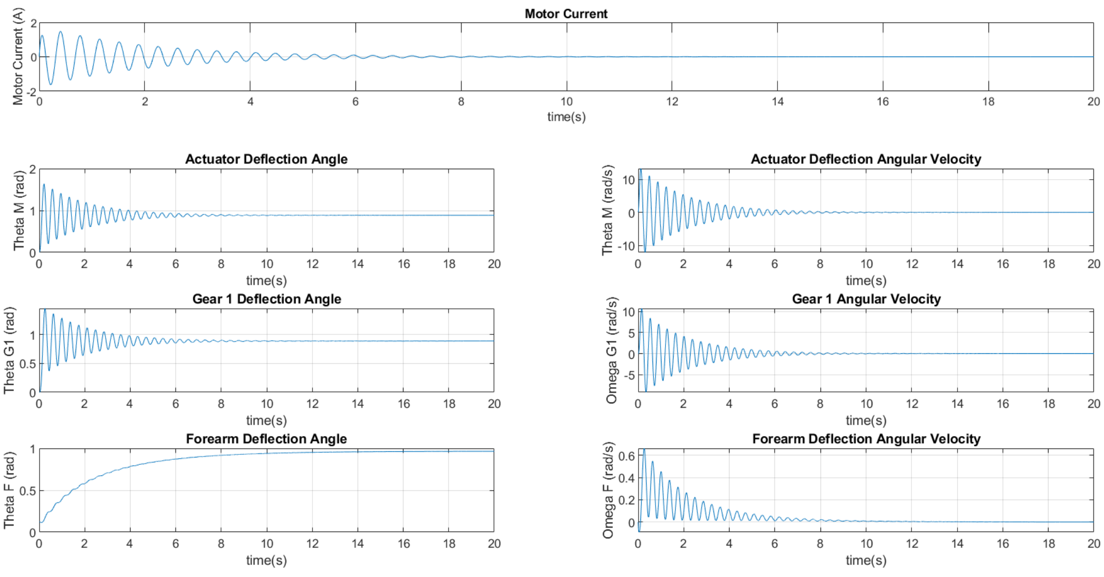


Figure 7 - MATLAB simulation results

We can see that the oscillations in motor current and angular velocity reduced over time which shows that the damping coefficients of the system are appropriate.

From the results of the forearm deflection can be seen to tend to $0.965\ rad/s\ \approx 55°$ as expected. However, the simulation shows there is some oscillation and not a smooth response.

The gear controller adjusts the proportionality between the error of the actuator deflection angle. A higher value of G_C makes the system react more aggressively to the error. Lowering the value of G_c would result in a smoother response.

<u>Block Diagram & Validation</u>

1.5 Simulink Block Diagram

Using the equations given from the lab sheet a block diagram simulation of the robot arm system was constructed in Simulink. The overall system contains subsystems following the description of the system and together they combine to simulate the response of the system.

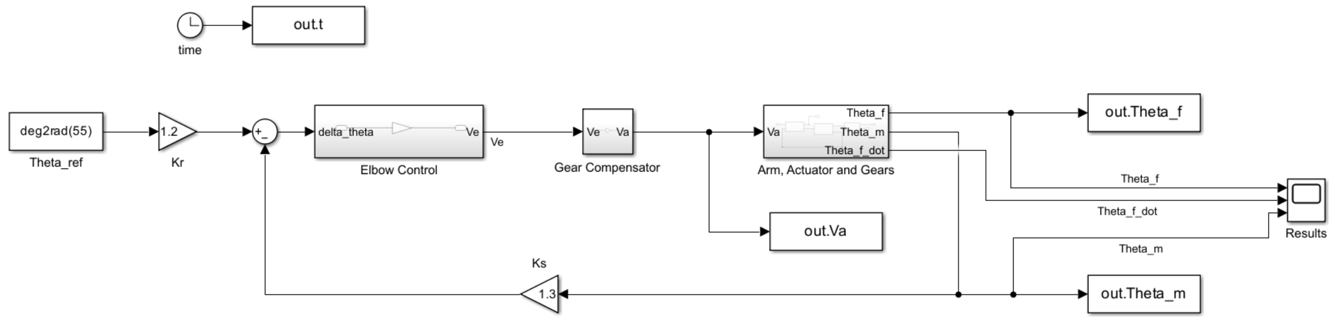


Figure 8 - Robot Arm System Block Diagram

The system starts with an input θ_{ref} which is then fed into the elbow control system. This system contains a gain block G_C which represents the proportional gain of the controller (figure 9). The resulting voltage V_E passes into the gear compensator (figure 10) which contains a gain block K_G which adjusts the voltage to a suitable level for the actuator.

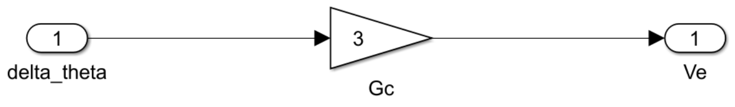


Figure 9 - Elbow control

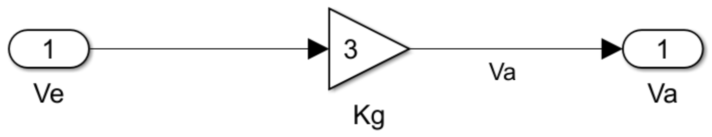


Figure 10 - Gear compensator

Within the arm, actuator & gears system there are three further subsystems (figure 11): the actuator, gear system, and forearm dynamics. The corresponding block diagrams were derived from the equations given in the lab sheet.

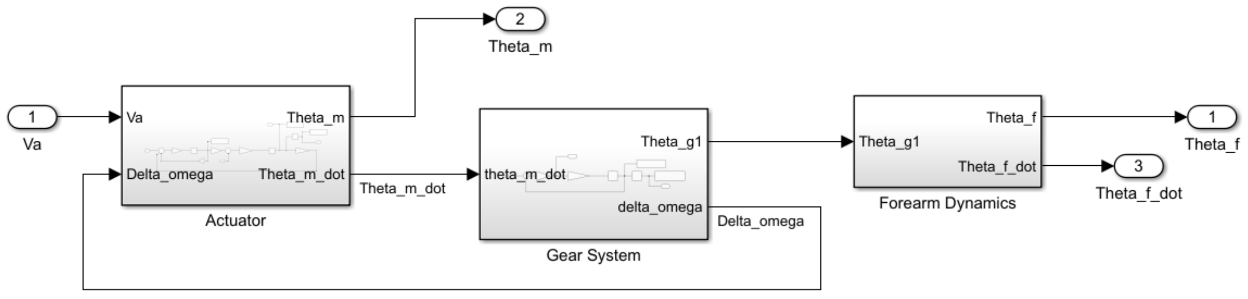


Figure 11 - Arm, actuator & gears

The actuator is made up of a D.C. motor which is implemented as shown in figure 12.

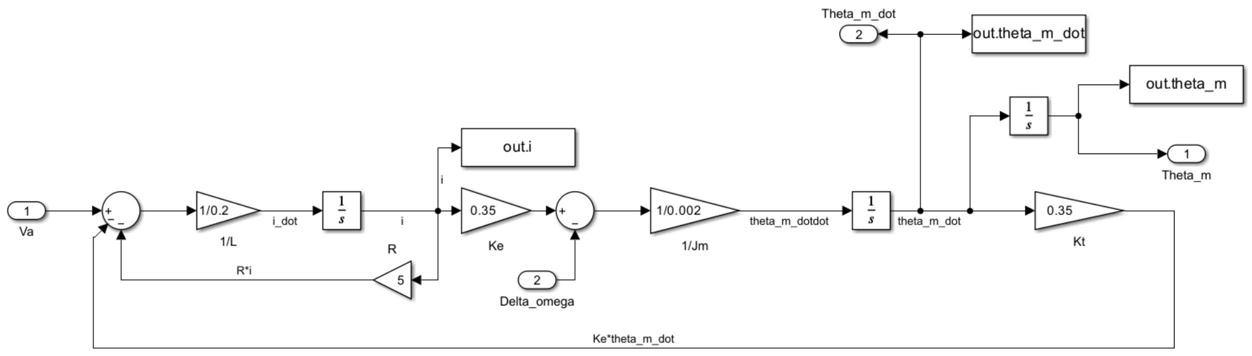


Figure 12 – Actuator

The value of $\dot{\theta}_M$ is then passed into the gear system subsystem (figure 13).

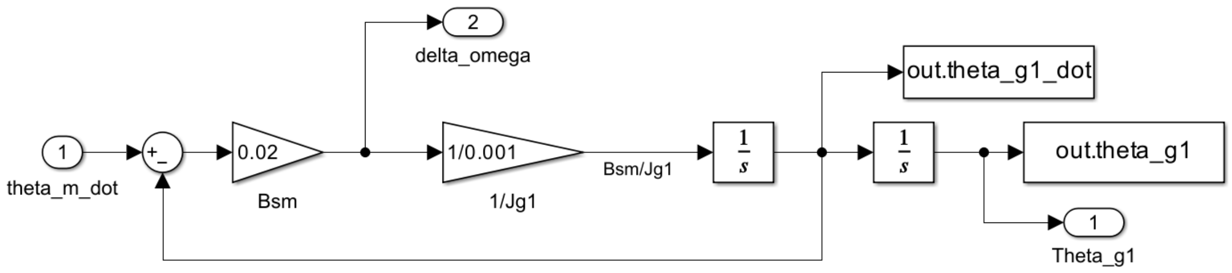


Figure 13 - Gear system

The final subsystem is the 'Forearm dynamics' (figure 14). This also represents the dynamics of the second gear.

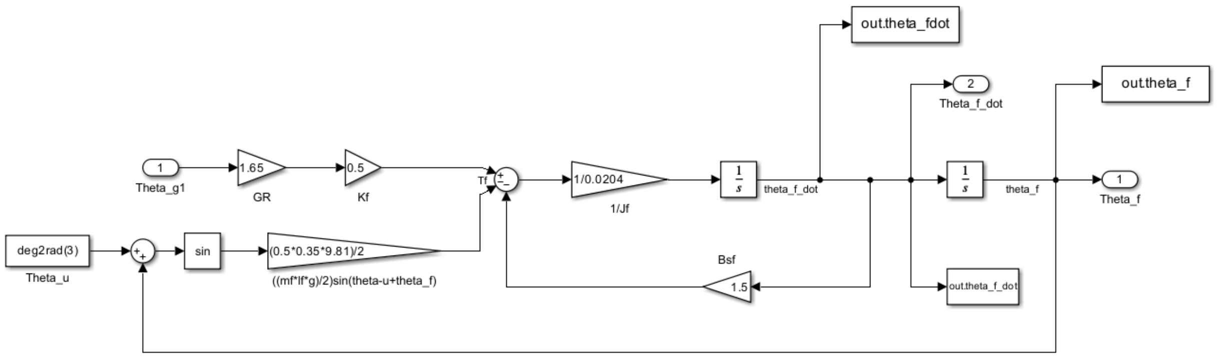


Figure 14 - Forearm dynamics

1.6 Validation Using Simulink Responses and Analysis

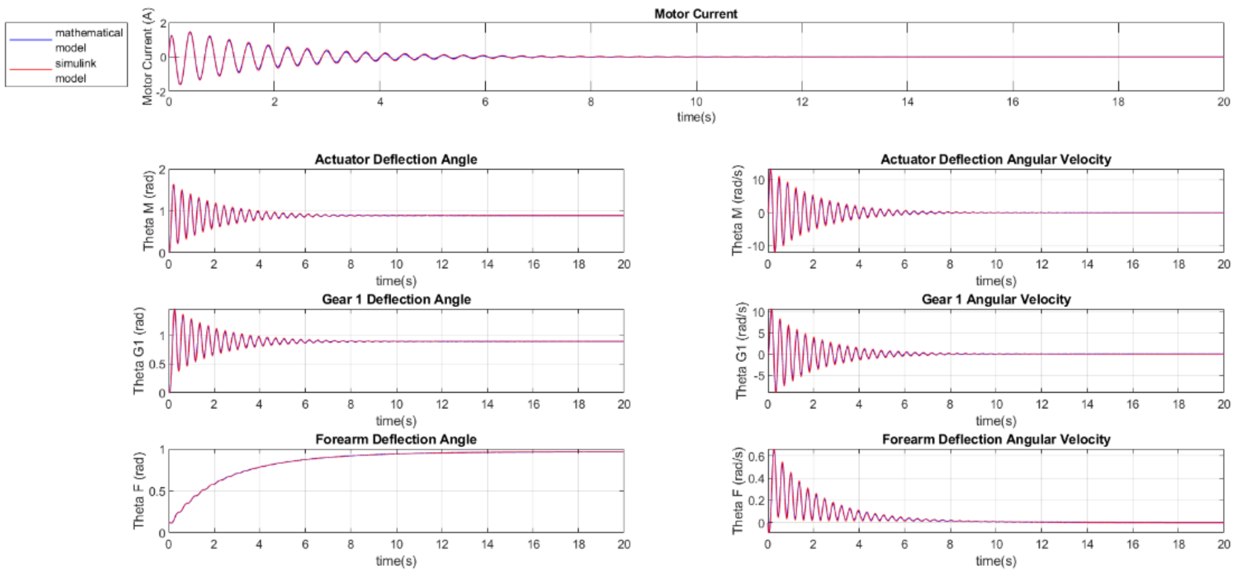


Figure 15 - Comparison of MATLAB and Simulink results

The results from both simulation models give the same shape but there are small discrepancies in amplitude between the two sets of results. The Simulink model produces slightly higher amplitudes. Since the gains are the same across both models, the error is likely due to rounding errors or differences in numerical solvers between the two simulation methods.

1.7 Conclusions for Part 1

The mathematical model developed from the state space of the robot arm system accurately captures the behaviour of the system including the motor dynamics, gear interactions, and forearm deflection. The Simulink model of the system produced results which were consistent with those of the MATLAB model and from both simulations we can see that the control system successfully reduces oscillations an motor current and angular velocity to settle around zero. Although the control system successfully achieved a forearm deflection angle of 55°, the gear controller used does not fully correct the oscillations in the response. Further tuning the value of G_C could eliminate this.