Introduction

This report focuses on the development of a simulation that represents a Robotic Arm Elbow Control System. This system changes the voltage applied to the actuator, the elbow motor, in order to produce the required rotational motion and change the deflection angle of the forearm (θ_F). By comparing the actuator deflection angle (θ_m) with our provided θ_{ref} (55°) the system produces the required actuator rotation that moves the forearm to θ_{ref}, this provides indirect control to θ_F, the forearms deflection angle. Using simulations to model such systems helps identify problems early in the development cycle, this enables engineers to deliver higher quality solutions faster and with lower risk.

1.1 State Space Model Derivation

The system equations provided are the following:

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

$$J_M\frac{d^2\theta_M}{dt^2} = K_T i - BSM\Delta\omega$$

$$J_{G1}\frac{d^2\theta_{G1}}{dt^2} = BSM\Delta\omega$$

$$T_F = J_F\frac{d^2\theta_F}{dt^2} + BSF\frac{d\theta_F}{dt} + \frac{m_f l_f}{2}g\sin(\theta_u + \theta_F)$$

In reduced form respectively:

$$\frac{di}{dt} = \frac{1}{L}\left(V_a - Ri - K_E\frac{d\theta_M}{dt}\right)$$

$$\frac{d^2\theta_M}{dt^2} = \frac{1}{J_M}\left(K_T i - BSM\Delta\omega\right)$$

$$\frac{d^2\theta_{G1}}{dt^2} = \frac{1}{J_{G1}}\left(BSM\Delta\omega\right)$$

$$\frac{d^2\theta_F}{dt^2} = \frac{1}{J_F}\left(T_F - BSF\frac{d\theta_F}{dt} - \frac{m_f l_f}{2}g\sin(\theta_u + \theta_F)\right)$$

The State Variables based on the reduced form of the provided equations.

$x_1 = i,$	Current through motor's circuit
$x_2 = \theta_m,$	Actuator deflection angle
$x_3 = \dot{\theta}_m,$	Angular velocity of the actuator
$x_4 = \theta_{G1},$	Angular deflection of Gear 1
$x_5 = \dot{\theta}_{G1},$	Angular velocity of Gear 1
$x_6 = \theta_F,$	Forearm's deflection angle
$x_7 = \dot{\theta}_F,$	Angular velocity of the forearm's deflection angle

Differentiating these state variables and substitution of states:

$$\dot{x_1} = \frac{di}{dt} = \frac{1}{L}\left(V_a - Rx_1 - K_E x_3\right)$$

$$\dot{x_2} = \dot{\theta_m} = x_3$$

$$\dot{x_3} = \ddot{\theta_m} = \frac{1}{J_M}\left(K_T x_1 - BSM(x_3 - x_5)\right)$$

$$\dot{x_4} = \dot{\theta_{G1}} = x_5$$

$$\dot{x_5} = \ddot{\theta_{G1}} = \frac{1}{J_{G1}}\left(BSM(x_3 - x_5)\right)$$

$$\dot{x_6} = \dot{\theta_F} = x_7$$

$$\dot{x_7} = \ddot{\theta_F} = \frac{1}{J_F}\left(T_F - BSF\frac{d\theta_F}{dt} - \frac{m_f l_f}{2} g\sin(\theta_u + x_6)\right)$$

The variable T_F is defined as the Gear rotation multiplied by the torque of the motor:

$$T_F = GR \cdot T_M = GR \cdot K_F \theta_{G1}$$

$$T_F = GR \cdot K_F x(4)$$

Therefore, $\dot{x_7} = \frac{1}{J_F}\left((GR\ K_F\ x(4)) - BSF\frac{d\theta_F}{dt} - \frac{m_f l_f}{2} g\sin(\theta_u + x_6)\right)$

This is the State Space Representation, the set of equations that describe the Robotic Arm System.

MATLAB - Script Approach

1.2 Matlab Model Function

```
function xdot = robotic_arm_model(x,V_A)
    % constants are in a separate file constants.m which runs when we run main.m
    global B_SM B_SF J_M J_G1 J_F g GR K_E K_F K_T L l_F m_F R theta_U;

    xdot = zeros(7,1);

    xdot(1,1) = (1 / L) * ((Va) - (R * x(1)) - (K_E * x(3)));
    xdot(2,1) = x(3);
    xdot(3,1) = (K_T / J_M) * x(1) - (B_SM / J_M) * (x(3) - x(5));
    xdot(4,1) = x(5);
    xdot(5,1) = (1 / J_G1) * (B_SM * (x(3) - x(5)));
    xdot(6,1) = x(7);
    xdot(7,1) = (1 / J_F) * (GR * K_F * x(4) - B_SF * x(7) - ...
                (m_F * l_F / 2) * g * sin(theta_U + x(6))); %theta_U = deg2rad(3)
end
```

Above is the model function of my MATLAB simulation code. The model is initially declared as robotic_arm_model(x, VA) which simulates the behavior of a robotic arm. The derivatives of my states are defined in this code as derived above in Section 1.1.

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

The initial state of the deflection angle of the forearm is given at 7° which was converted into radians in my constants.m file using deg2rad(7) and initiated in the main MATLAB script as:

$$x = \begin{bmatrix} 0; & 0; & 0; & 0; & 0; & \theta_{F0}; & 0 \end{bmatrix};$$

The control system itself is a proportional controller of the following form:

$$V_E = G_C \ \Delta\Theta$$

Where G_C is the gain of the controller and $\Delta\theta$ is the difference between the reference angle and the actuator deflection angle which is measured by the Actuator Sensor represented by a simple gain K_S. The reference deflection passes through the Reference Amp which is represented by a simple gain K_R:

$$\theta_{ref} = K_R * deg2rad(55) \tag{1}$$

$$V_E = G_C(\theta_{ref} - K_S \ x(2)) \tag{2}$$

The resulting commanded voltage V_E (volts) then passes through the Gear Compensator, which is simply a gain, K_G. The compensated voltage, V_A (volts), is used to control the elbow actuator to drive the gears and thus indirectly generate an appropriate forearm deflection:

$$V_A = V_E \ K_G \tag{3}$$

The equations (1), (2) and (3) can be seen in the Control Section of the main.m file inside the Dynamic Section of the code.

The Numerical Integration method chosen for this simulation is the The Fourth-Order Runge-Kutta Method (RK4), using exact code taken from laboratory 2. This integration method was used due to its reliable capabilities in simulations where the aim is to maintain numerical stability. In comparison to other methods, RK4 is described by accuracy and better proximation. Compared to other methods, RK4 had much lower error, as h gets smaller the higher order method gets better.

The step-size chosen for this simulation is 0.001, this choice considered time resolution, accuracy and stability. Due to the given Gains, a lower step size does not work for the simulation, when using 0.01 the simulations did not capture the correct behavior. The chosen step size captures finer details of the system's behavior and it reduces truncation errors. Even if using 0.001 means increased computational load, this step size ensures stability in a system with high-frequency dynamics that is very sensitive like ours.

1.4 Simulation Responses and Analysis

Fig. 1 below depicts the response of all seven states over time, this simulation was run over 25 seconds in order to get an accurate representation of the behavior over time and using a 0.001 step-size.

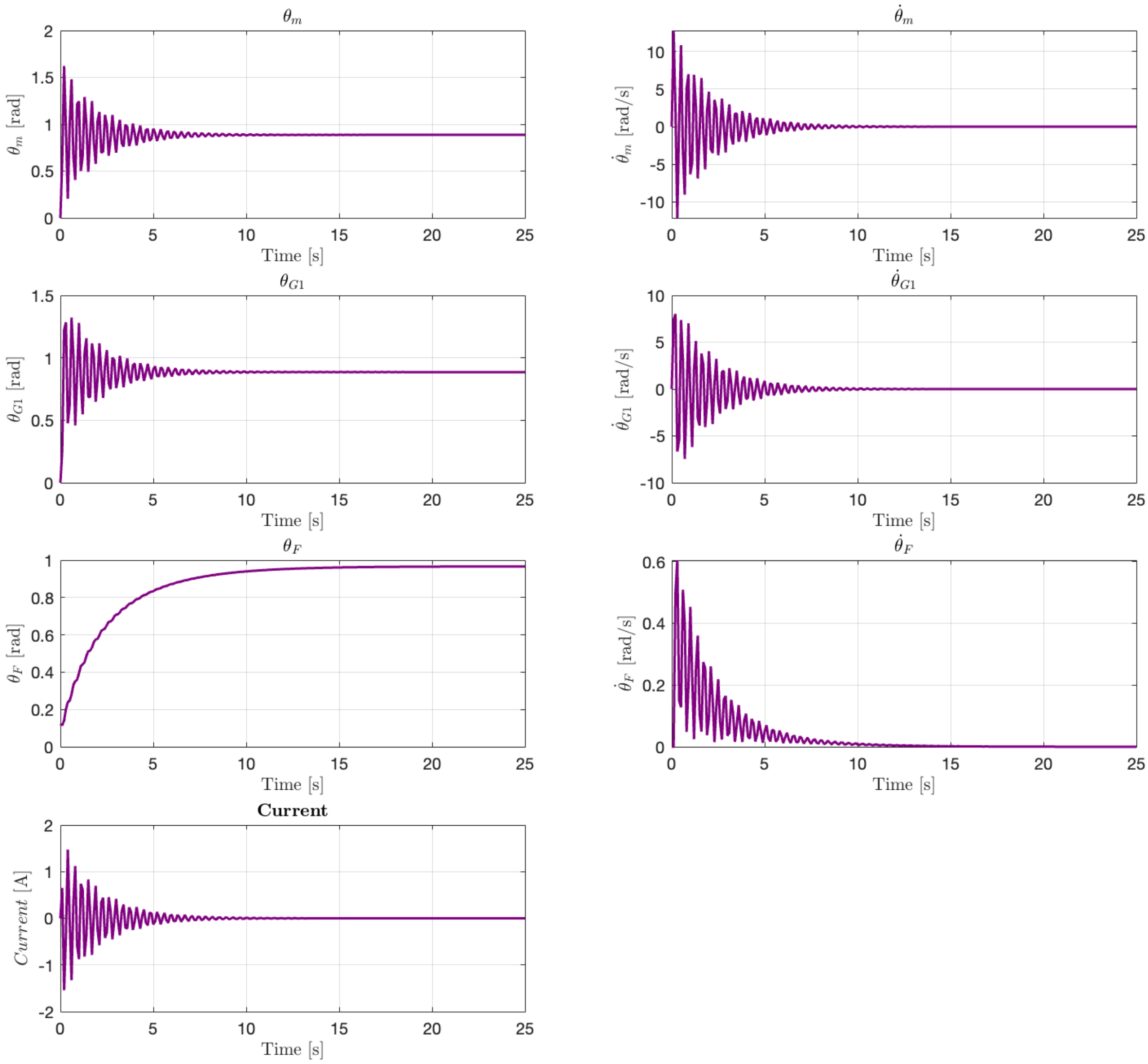

Figure 1: MATLAB Responses of States Over Time

The graphs depicting θ_M and θ_{G1} start from zero and oscillate with damping before stabilizing at 0.88 rad, which is around 50°. This is due to the reference angle being 55°, the system does not reach exactly 55° due to multiple factors such as damping and the given gains; this error is reflected as $\Delta\theta$. The use of a proportional control system does not eliminate steady-state error ($\Delta\theta$) entirely, hence this is the natural response of using this type of controller without an integral component. The addition of the integral term into the controller would remove the steady state error. The graph indicates that θ_M and θ_{G1} undergo transient oscillations but converge to a steady-state value over time, the length of the oscillations is also determined by the Gains chosen. The graph of θ_F starts at 0.12rad (θ_{F0}), this value slowly increases until it reaches steady state on 0.96 rad, matching θ_{ref}. The response of θ_F differs from the rest of the responses, this state has stronger damping effects, BSF which highly dampens its behavior. Additionally, compared to θ_M and θ_{G1}, the inertia of the forearm is much higher, J_F is 0.0204 (J_M was 0.002), this leads to a more composed response when it comes to rapid changes in angular acceleration, hence a more gradual response. The current which drives the motor oscillates reflecting it's ability to dynamically adjust the motor torque in response to control inputs. Finally, it comes to a steady-state at 0° when no further input is required to achieve the final position.

All three graphs of $\dot{\theta}_M$, $\dot{\theta}_{G1}$ and $\dot{\theta}_F$ oscillate and come to rest at 0°. $\dot{\theta}_M$ oscillates with high initial peaks that eventually stabilize due to damping. $\dot{\theta}_{G1}$ does not have as sharp peaks as $\dot{\theta}_M$, however, it comes at a steady state around the same time. Finally, $\dot{\theta}_{G1}$, the forearm's angular velocity decays quicker compared to the rest of the states, this is due to the inertia and damping that filter out higher frequency oscillations as mentioned above. Additionally, it includes a gravitational force term that provides a restoring force which stabilizes the response.

MATLAB - Simulink Approach

1.5 Simulink Block Diagram

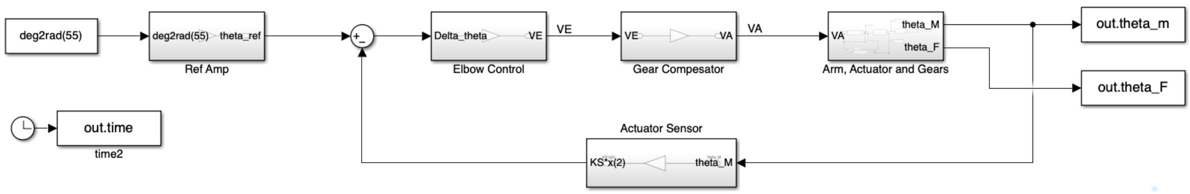

Figure 2: Simulink Block Diagram using Subsystems

Above is the Simulink representation of the Robotic Arm System. The system is separated into the main subsystems and in turn as seen in Fig. 3 the "Arm, Actuator and Gears" subsystem is composed out of three further subsystems. The rest of the subsystems in Fig. 2, the Elbow Control (G_C), Gear Compensator (K_G), Actuator Sensor(K_S) and the Ref Amp (K_R) are all simple gains as described in the specification.

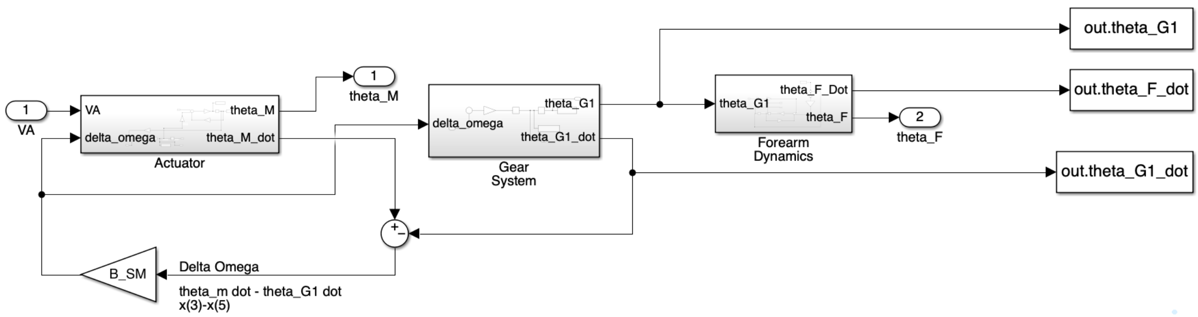

Figure 3: Arm, Actuator and Gears Subsystem

The inside of each subsystem can be found in the Appendix A.

As shown in Fig. 3, the Actuator takes as input VA, this actuator voltage is used to drive the actuator to deflection θ_M by means of its generated torque, TM. The Actuator and the Gear System share $\Delta\omega$, which is derived as the difference between $\dot{\theta}_M$ (from the Actuator output) and $\dot{\theta}_{G1}$ (from the Gear System output). This shared $\Delta\omega$ serves as an input to both subsystems and represents the relative angular velocity difference. This difference plays a role in connecting the two system through the drive shaft of the actuator which is physically connected to Gear 1 and acts as a load on the motor. The Gear System calculates θ_{G1}, the angular deflection of Gear 1, along with $\dot{\theta}_{G1}$ that is being fed back to compute $\Delta\omega$. Subsequently, the Forearm Dynamics subsystem uses θ_{G1} as input to calculate T_F, the torque acting on the forearm. This torque drives the forearm dynamics and therefore leads to derive the angle of the forearm, θ_F and $\dot{\theta}_F$.

1.6 Validation Using Simulink Responses and Analysis

Fig. 4 below depicts the responses of the system when modeled through Simulink in comparison to MATLAB script code. In general the two methods agree very closely which indicates that the system is modeled correctly in both. There are some noticeable differences in some of the states, while in θ_F it is evident that the two responses are identical. Both of the simulation methods used the same solver, RK4 and step size of 0.001, hence the discrepancies between them are most likely due to other factors. These factors might include floating point precision errors or additional Simulink methods designed to handle stiff systems.

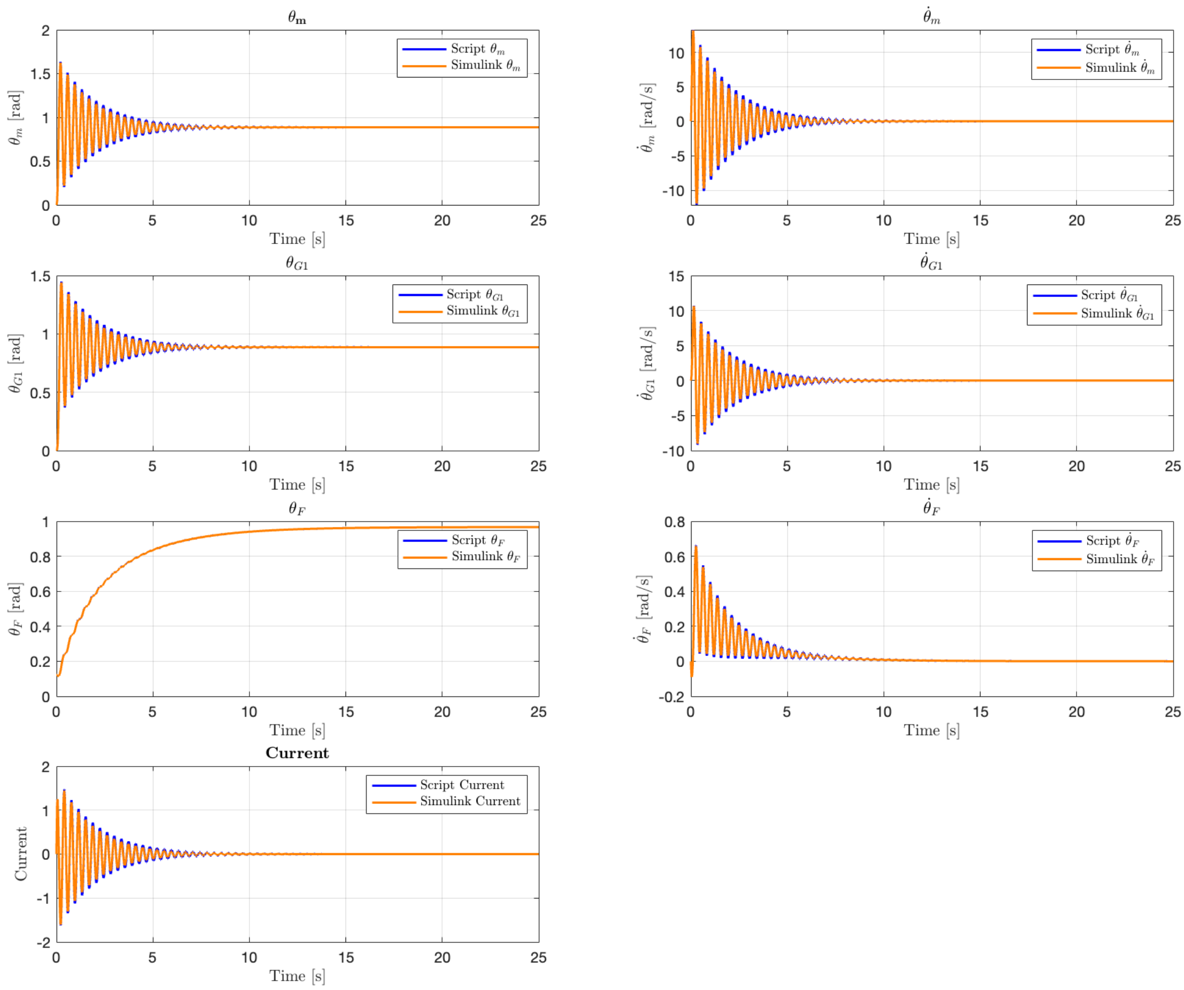

Figure 4: MATLAB over Simulink Responses

Conclusion

This simulation successfully captured the dynamic response of the Robotic Arm Model. The behavior of the states were observed through both MATLAB script code and Simulink Block Diagram methods, these agreed with only minor discrepancies. However, refining Gain values (such as Gc) is required, this would allow the tuning of the system and in turn a faster steady-state response without as many oscillations. Finally, this design uses a Proportional Controller, adding an integral term and creating a Proportional-Integral Controller would eliminate the steady-state error and would deliver more accurate results with reduced oscillations.

Appendix A - Simulink

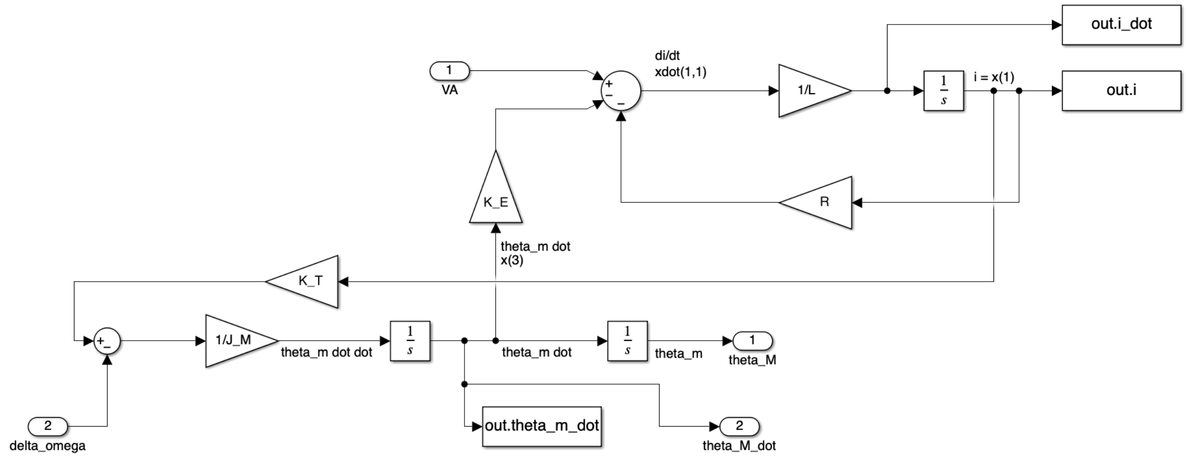

Figure 5: Actuator Subsystem

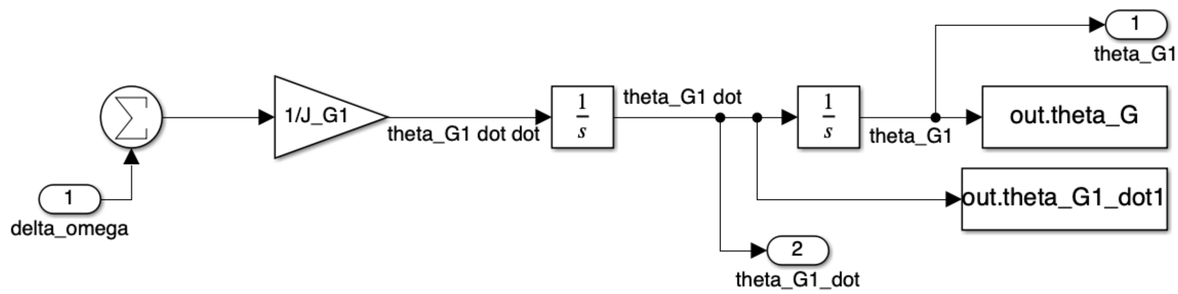

Figure 6: Gear Subsystem

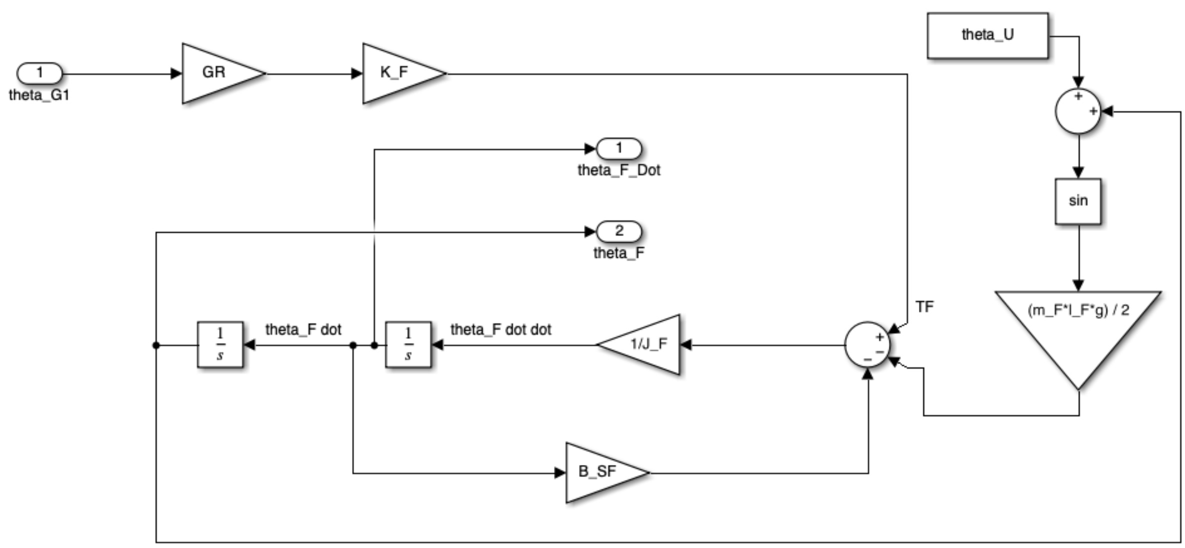

Figure 7: Forearm Dynamics Subsystem