AE353: Design Problem 02

Emilio Gordon

March 10, 2017

1 Goal

`DesignProblem02` simulates a "gravity-assisted under actuated robot arm" that is similar to one proposed for wing-box assembly in aircraft manufacturing by Roy Assad. The robotic arm is equipped with two joints however, only one is motorized. The motor applies a torque about the first joint while the second joint is allowed to spin freely. Finally, the robotic arm is equipped with optical encoders to measure joint angles and joint velocities.

The goal is to make for the robot arm to reach a stable equilibrium and then for the tip of the robot arm to move in accordance to a sequence of reachable points in space given by a predefined function. This would simulate the behavior of a robotic arm working inside of an aircraft's wing creating rivets. A requirement for the model is established such that

- The second joint angle shall remain within $\pm 5^\circ$ of its instantaneous reference value throughout the duration of the simulation.

With this requirement in mind, a procedure for verification can be established. Verification ensures that the system meets the established requirements. Verification will be conducted through several methodologies and are as follows

- The simulation will generate an error between the second angle and the instantaneous reference value.
- For every time interval, a steady-state error is computed
- Plot a steady-state error to time curve. On this curve, include a reference line to mark if and when your steady-state error has surpassed the value stated by the requirement.
- Analyze the plot. If values exceed $\pm 5^\circ$, the single requirement has been ignored and the simulation has failed.

2 Model

In order to achieve the task at hand, a linearized state-space model for the system must be created. The sensors on the joints of the robotic arm return information pertaining to the

joint's angle and it's velocity. As a result, q is defined to be a matrix of joint angles while $\dot{q}$ is a matrix of joint velocities. Finally, the matrix τ represents the applied torques of the system.

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \qquad \dot{q} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \qquad \tau = \begin{bmatrix} \tau_1 \\ 0 \end{bmatrix} \tag{1}$$

It is seen in Equation Set (1) that the matrix q takes in the joint angles for both joints one and two. Is is also seen that the matrix $\dot{q}$ takes in the velocities at both joints one and two. Notice however, how τ has τ_1 and zero. This is the consequence of the second joint not having a motor, therefore, being incapable of applying torque.

The motion of the robot is governed by ordinary differential equations with the form

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + N(q, \dot{q}) = \tau, \tag{2}$$

such that

$$M(q) \qquad C(q, \dot{q}) \qquad N(q, \dot{q})$$

are matrix-valued functions of q and/or $\dot{q}$. These functions depend on a number of parameters include mass and moment of inertia. Using the code provided in the DesignProblem02 document, the ordinary differential equations were retrieved. These equations of motion are complex and rigorous to compute. Through the use of Matlab, the equations of motion can be linearized entirely through programming. Figure 1 demonstrates the code necessary to load the Equations of Motion for this system.

```
% Load the equations of motion.
load('DesignProblem02_EOMs.mat');
% Parse the equations of motion.
M = symEOM.M;
C = symEOM.C;
N = symEOM.N;
tau = symEOM.tau;
% Define symbolic variables that appear in the equations of motion.
syms q1 q2 v1 v2 tau1 real
```

Figure 1: Code to retrieve a symbolic description of the equations of motion in MATLAB.

Having established the elements of the state-space model, the nonlinear system must be linearized in order to simplify calculations. An initial equilibrium matrix is utilized to demonstrate the states and input for which the system is at equilibrium. The initial equilibrium is chosen such that

$$x_{eqillibrium} = \begin{bmatrix} q_{1e} \\ q_{2e} \\ v_{1e} \\ v_{2e} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

This establishes that when the robotic arm has both joints at an angle of zero, their velocities will be zero and therefore stationary. The goal wishes to observe the behavior of the arm

as it moves to reach a sequence of reachable points. As a result, by choosing an initial equilibrium of zero, controllability is predicted to be easier since gravity will be acting againt the motor rather than with. The code in Figure 2 shows how easy computing the Jacobian

```
%Establish equations for qdot (velocity) and qdotdot (acceleration)
qd = [v1; v2];
qdd = inv(M)*(-C*qd-N+tau);
%Define an equilibrium point containing input equilibrium [state state state state
    input]
equil = [0 0 0 0 0];
%Linearization takes palce by taking the jacobian of the equations wrt. the
    equilibrium points
Alower = vpa(subs(jacobian(qdd,[q1 q2 v1 v2]),[q1 q2 v1 v2 tau1],equil));
A = [0 0 1 0; 0 0 0 1; Alower];
Blower = vpa(subs(jacobian(qdd,tau1),[q1 q2 v1 v2 tau1],equil));
B = [0; 0; Blower];
C = eye(size(A));
```

Figure 2: Code used to linearize the state-space system about equilibrium

and Linearizing the system can be. The results to the code in Figure 2 is shown in Equation Set (4) producing the A, B and C matrix for the state-space model.

$$A\left(\left[qdd_e\right], x_{eqil}\right) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4.967 & -54.77 & -3.284 & -0.6294 \\ -2.182 & -10.50 & -0.6294 & -0.3085 \end{bmatrix}$$

$$B\left(\left[qdd_e\right], x_{eqil}\right) = \begin{bmatrix} 0 \\ 0 \\ 3.284 \\ 0.6294 \end{bmatrix} \qquad C\left(\left[qdd_e\right], x_{eqil}\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D\left(\left[qdd_e\right], x_{eqil}\right) = \begin{bmatrix} 0 \end{bmatrix} \tag{4}$$

The resulting state-space model is

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

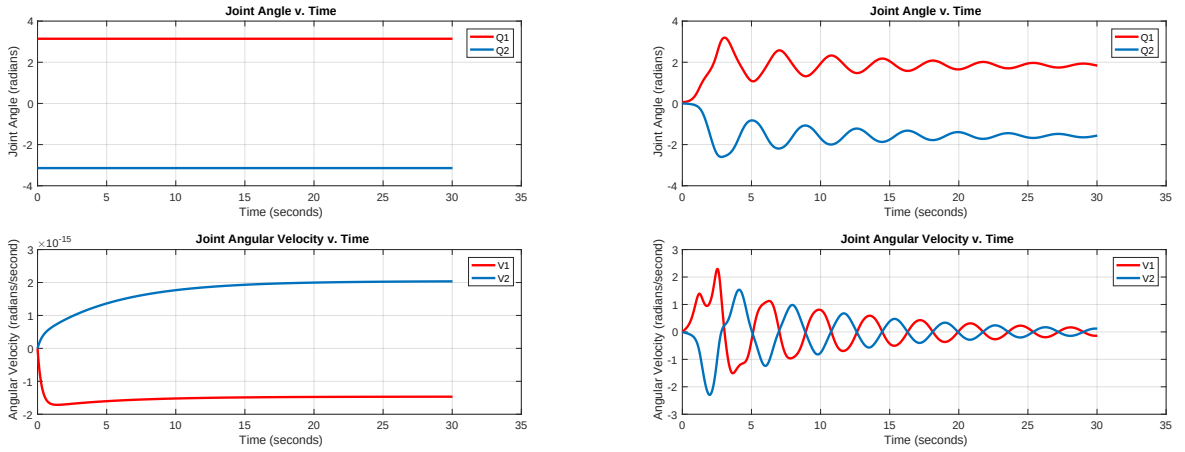
such that the behavior of this linear system and of the original, nonlinear system will be approximately the same so long as x and u are close to equilibrium.

3 Controller

The controller implemented for this design problem is a closed-loop linear system implementing reference tracking which allows the robotic arm to track a desired reference point and remain asymptotically stable so long as that reference point is within the bounds of the physical arm. There are multiple types of controllers, each advancing upon the last.

3.1 Open Loop Linear System

The first controller implemented is one with a zero-input solution. This refers to the systems behavior without any torque being applied to it through joint one. This is useful for understanding the effects the controller will have on the system once an input has been implemented. The system when controlled by a zero-input solution is shown in Figure 3. Represented in Figure 3a is the system initially at a position that equal to its' physical equilibrium (recall that this is a gravity-assisted system) it will remain in that position. However, if this system were to start at a location with a slight offset from equilibrium, it will become unstable as demonstrated in Figure 3b. It is important to note that this system can be in equilibrium at two angle positions to be defined as "natural equilibrium". The first where $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and the second where $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} \pi & -\pi \end{bmatrix}$. Note that the former was the same as the equilibrium set for this problem. Also note that Figure 3a is positioned at $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} \pi & -\pi \end{bmatrix}$



(a) Velocities and joint angles at equilibrium (stable)

(b) Velocities and joint angles with a slight offset from equilibrium (unstable)

Figure 3: Robotic arm during zero input solution

3.1.1 Open Loop Controllability

Not every system is controllable. If one wanted to make the entire state x go to a desired value at some time, a controller must be implemented. By choosing some input $u(t)$, which is a function of time, the desired value should be reached. If it is possible for a system to move

from state x_0 to any final state x_f then the state-space system is considered controllable. Controllability is not always guaranteed so it is important to test for controllability before advancing with design. To solve this, the controllability matrix is implemented. The controllability matrix is associated with the original system so it is easy to compute. Figure 4 shows the calculation necessary to compute the controllability matrix. The system is controllable

```
W = [B A*B A^2*B A^3*B]
```

$$W = \begin{bmatrix} 0 & 3.2840 & -11.1811 & 19.9813 \\ 0 & 0.6294 & -2.2613 & -6.0366 \\ 3.2840 & -11.1811 & 19.9813 & 6.4905 \\ 0.6294 & -2.2613 & -6.0366 & 37.4163 \end{bmatrix}$$

Figure 4: Controllability matrix calculation and result.

if and only if W, the controllability matrix, is full rank. This simply means that the matrix W is invertible. If the controllability matrix is invertible, it is possible to place eigenvalues at desired locations thereby giving the user control over the system. If the controllability matrix W is not invertible, then only some of the system's eigenvalues can be placed at desired locations, and the remaining eigenvalues cannot be moved. In the Figure 3.1.1 the rank between A and W are compared. Since they are the same, $rank(A) == rank(W)$, the open-loop linear system is controllable.

```
rank(A)
rank(W)
```

Figure 5: Controllability rank verification

3.2 Closed Loop Linear System

The second controller to be implemented is the state-feedback controller. State feedback is defined as a system in which the input depends on the state. State feedback has the input

$$u = -Kx \tag{5}$$

In equation 5, K represents a constant matrix whose values can be defined and manipulated to achieve equilibrium. The way in which state feedback,$u = -Kx$, differs from zero input response, $u = 0$, is through the inputs ability to change. Previously, the system would act on its own wether it was stable or unstable due to gravity. However, with a state feedback, this is no longer the case. Torque is now applied to the robotic arm and with a proper formulation of K the system can converge to the desired equilibrium points.

3.2.1 Linear Quadratic Regulator

The K matrix is found by implementing the Linear Quadratic Regulator (LQR), a state feedback controller that connects to the theory of optimal control. LQR is implemented easily through Matlab with the function

```
K = lqr(A,B,Q,R) %Linear Quadratic Regulator
```

The LQR function calculates the optimal gain matrix K such that:

- For a continuous-time state-space model, the state-feedback law $u = -Kx$ minimizes the cost function $J = \int (x'Qx + u'Ru)dt$ subject to the system dynamics $dx/dt = Ax + Bu$
- For a discrete-time state-space model, u[n] = -Kx[n] minimizes $J = \sum x'Qx + u'Ru$ subject to $x[n+1] = Ax[n] + Bu[n]$.

The matrices Q and R are parameters that can be used to trade off error with effort. These matrices must be symmetric and be the right size. It is also important to note that

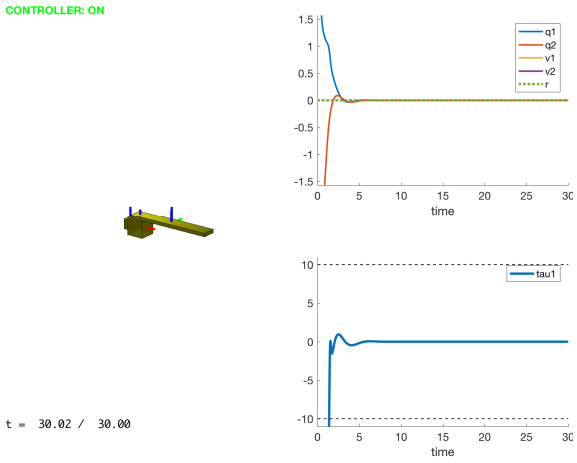
$$Q \geq 0 \quad R > 0$$

As mentioned already, the matrices Q and R are parameters that can be used to trade off error with effort. This can be thought of as penalties with accordance to the states and inputs of the system. The more any of these are penalized, the less willing to change they will be. For example, if v_2, the angle velocity of joint two, had a high penalty, it would be more penalized and therefore, make smaller changes to avoid penalization. It is important to understand that Q refers to penalties acting on the systems states and R refers to penalties acting on the systems inputs. Equation 6 establishes the Q and R matrix chosen to best achieve the goal.

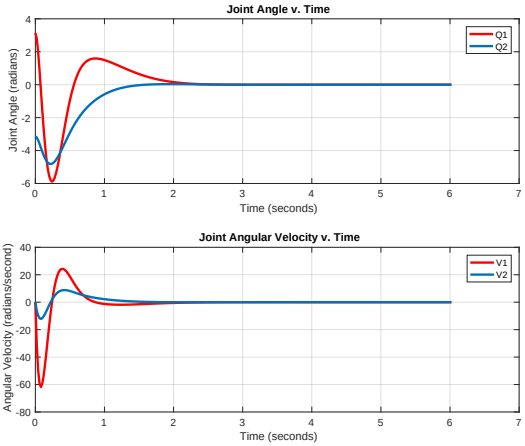
$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 10000 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 \end{bmatrix} \tag{6}$$

It is seen here that highly penalizing the angle of joint one q_1 and joint two q_2 resulted in the most optimal Q value. This will be explained further later in this section.

With the establishment of a Q and R matrix, the Matlab function lqr() resulted in a K which, when implemented into the state-feedback input of $u = -Kx$ allowed the system to reach equilibrium. Figure 6 show the system reaching an equilibrium from the initial position of $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} \pi & -\pi \end{bmatrix}$. This is a complicated maneuver to accomplish since the robotic arm is battling gravity the entire way. Alternatives to this are shown in Figure 6c and Figure 6d where the robotic arm has a random initial value. These initial conditions require much less work to be done and therefore converge faster. Regardless of the initial condition however, the arm converges to the equilibrium.

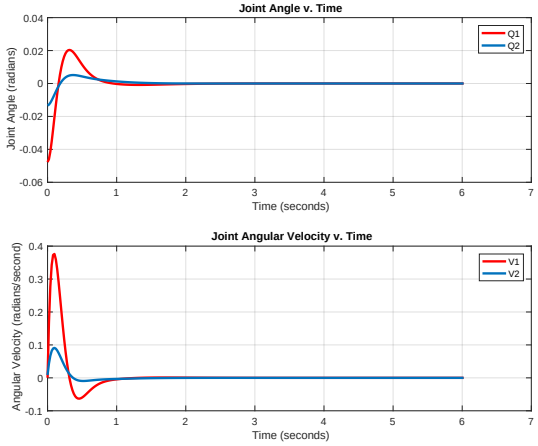


(a) Equillibirum via State Feedback

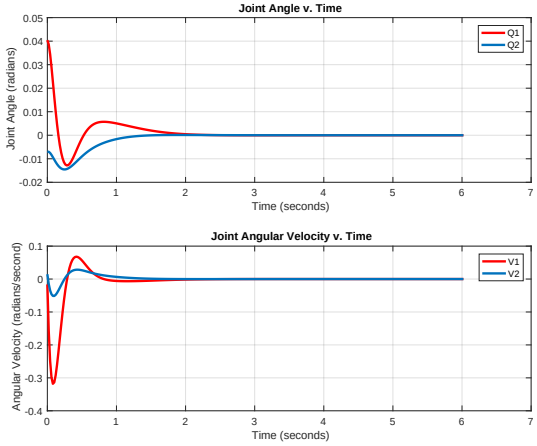


(b) Robotic arm with initial

$$\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} \pi & -\pi \end{bmatrix}$$



(c) Robotic arm with random initial conditions

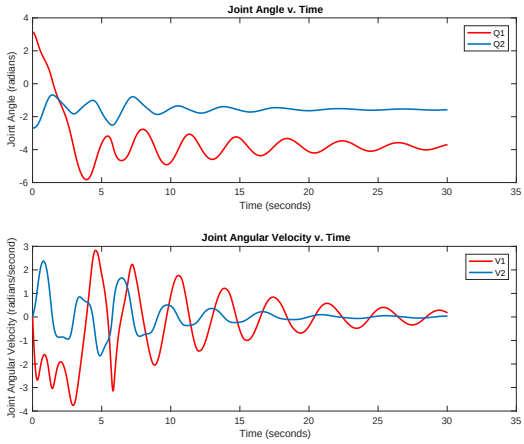


(d) Robotic arm with random initial conditions

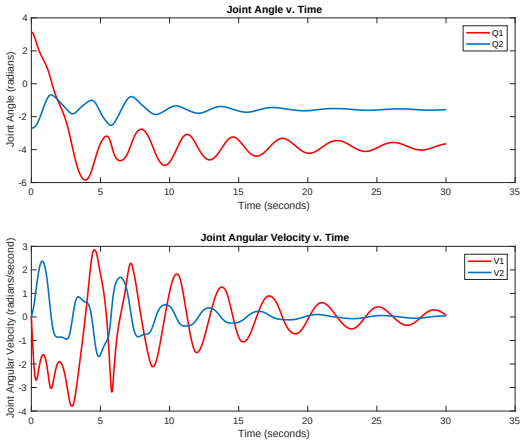
Figure 6: Joint angles and angular velocities of Robotic arm with a state feedback controller going to equilibrium.

The matrices Q and R were used to calculate K. It can be seen in Figure 7 that if the penalties are changed, so will the convergence of the system. Figure 7a shows the system when the penalty on the states is simply the identity matrix. Figure 7b shows the system when the penalties acting on q_1 and v_2 are five while penalties on q_2, v_1 and R are 50. Note the similarities between 7a and 7b. Figure 7c shows the system when the penalties acting on q_1, v_2 and R are five while penalties on q_2 and v_1 are 50. By changing R to five, a clear change occurs in the sytem. The 'cost' that Q and R have on a system is a result of thier ratios. In 7b the ratio of Q to R is the same as 7a (in the places it counted). However, in 7c the ratios are different which result in a stark difference in the system. Finally Figure 7d shows the system when induced to the original Q and R values introduced earlier in this section.

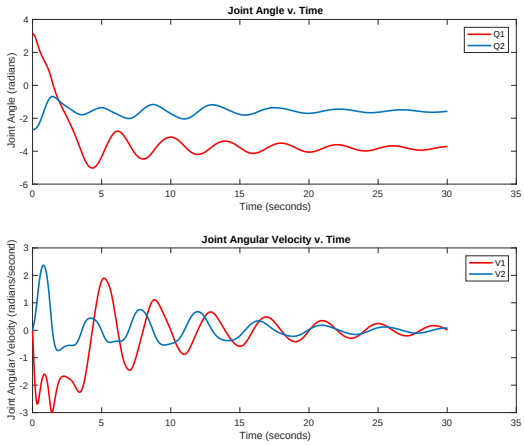
It should be noted that the plots in Figure 7 has the initial starting value of $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} \pi & -\frac{6}{7}\pi \end{bmatrix}$. The initial value for q_2 was chosen as it is slightly off from a "natural" equilibrium point. Recall from the previous section how the natural equilibrium points were $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and $\begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} \pi & -\pi \end{bmatrix}$. By defining q_2 as such, it can be assured that the system will be able to reach some equilibrium in time.



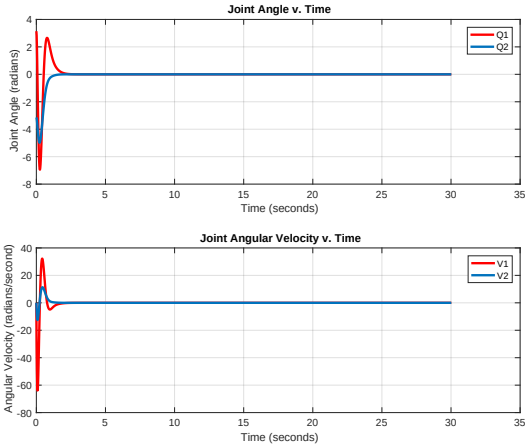
(a) $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $R = \begin{bmatrix} 1 \end{bmatrix}$



(b) $Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$ $R = \begin{bmatrix} 50 \end{bmatrix}$



(c) $Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$ $R = \begin{bmatrix} 5 \end{bmatrix}$



(d) $Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 10000 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $R = \begin{bmatrix} 1 \end{bmatrix}$

Figure 7: System undergoing different values of Q and R, thereby effecting K

3.3 Stability

At this time, both a open-looped zero-input system and a closed-loop state-feedback system have been tested. The stability of these systems can be determined by studying the eigenvalues of these systems.
For the case of a zero-input response, the following input to MATLAB is made for calculating the eigenvalues.

```
[V,F]=eig(A) %Zero Input
```

For the case of state-feedback, the following input to MATLAB is made for calculating the eigenvalues.

```
[V,F]=eig(A-B*K) %State Feedback
```

Where A, B and K are all predefined matrices for the state-space model. The term for which the eigenvector and eigenvalues is taken from is the term dependent on the state of the system as shown in Equation 7 as A or $(A - BK)$

$$\dot{x} = Ax \qquad \dot{x} = (A - BK)x \tag{7}$$

After computing the MATLAB function, the matrices V and F are created. The matrices are structured such that the diagonal entries of F are the eigenvalues of A or $A-BK$. These eigenvalues are denoted by s_1, s_2, ..., s_n, such that

$$F = \begin{bmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & s_n \end{bmatrix}.$$

From this, it is found that the exponential of Ft produces the result

$$e^{Ft} = \begin{bmatrix} e^{s_1 t} & 0 & \dots & 0 \\ 0 & e^{s_2 t} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & e^{s_n t} \end{bmatrix}.$$

The exponential establishes that the behavior of $x(t)$ is dependent on the entries in the matrix e^{Ft}. Two relations can be seen as a result

- If s_1 is a positive real number, then $e^{s_1 t}$ grows exponentially as t increases meaning the system is unstable.
- If s_1 is a negative real number, then $e^{s_1 t}$ decays exponentially to zero as t increases meaning the system is stable.

With this information in mind, the stability for the zero-input open loop controller and the state-feedback closed-loop controller is verified. It is seen that the zero-input controller

produces a positive eigenvalue verifying it is unstable. Opposing this, the state-feedback closed-loop controller has all negative real parts, verifying its stability.

$$F_{ZeroInput} = \begin{bmatrix} 2.7562 & 0 & 0 & 0 \\ 0 & -4.2366 & 0 & 0 \\ 0 & 0 & -1.0561 + 3.6854i & 0 \\ 0 & 0 & 0 & -1.0561 - 3.6854i \end{bmatrix}$$

$$F_{StateFeedback} = \begin{bmatrix} -32.8790 & 0 & 0 & 0 \\ 0 & -1.2210 + 1.8225i & 0 & 0 \\ 0 & 0 & -1.2210 - 1.8225i & 0 \\ 0 & 0 & 0 & -1.9273 \end{bmatrix}$$

3.4 Reference Tracking

The final controller to be added is reference tracking. With the implementation of reference tracking, the input is made to be linear to the reference point r.

$$u = -Kx + k_{ref}r \tag{8}$$

Reference tracking introduces two new terms into the input equation, k_{ref} and r. The value for r represents the "reference." It chosen such that as $t \to \infty$ the value $y \to r$. For the purposes of achieving a robotic arm, this reference value is subject to change through the duration of the simulation. Despite this, the initial r value shall be set equal to equilibrium. The value for k_{ref} is calculated such that the system reaches steady state. The definition of these two knew terms is expressed in Equation 10.

$$r = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \qquad k_{ref} = \frac{1}{-C(A - BK)^{-1}B} \tag{9}$$

It is important to note that the output of k_{ref} is a 1x4 matrix whose entries are all zero except for the value which corresponds to q_2, the angle of the second joint. With this information, the two entries can be simplified to scalars 10.

$$r = f(t) \qquad k_{ref} = \begin{bmatrix} 0 & -316.6672 & 0 & 0 \end{bmatrix} = -316.6672 \tag{10}$$

acknowledging that k_{ref} and r only apply to q_2. Since the aforementioned goal for this system was to be able to track a sequence of reference points given by a function, r is represented as a function of time. Any given function would be a possible reference value. However, given the constraints of the model and it's physical limitations, several factors should be kept in mind when choosing a formula for r such that the system does not loose stability and to reduce the state error.

- The function should start at the systems equilibrium at the time it is called.
- The function should be continuouse

- If the system were to not be continuous and contain steps, the step intervals of the function should consider being less than one.
- The function should restrain from steep slopes.

These rules are not required in order for reference tracking to be successful however they greatly reduce the state error which is critical when considering the aforementioned goal. With these rules in mind and a certain personal vision for how the arm should behave, the following function for $f(t)$, the reference trajectory, was contrived.

$$f(t) = \frac{1}{5}\sin(0.1t - 0.6)\cos(0.01\pi t^{1.6} - 0.6)$$

With an understanding of waveforms, this function was created so have a reference path of a 'beating' wave with low amplitude and a large wavelength so to satisfy the rules above.

To implement reference tracking into the controller, a separate MATLAB script had to of been created. The function 'GenerateReferenceTrajectory' completed this task and is shown in Figure 8

```
%Function GenerateReferenceTrajectory.m
function q2 = GenerateReferenceTrajectory(t)
    if t < 6
        %Reference Value from t=0 to t=6
        q2=0
    else
        %Reference Value for t > 6
        q2 = SomeFunction f(t)
    end
end
```

Figure 8: Code to define and implement reference values for controller in MATLAB.

The code from 'GenerateReferenceTrajectory' tells the system that for the first six seconds of the system, the reference point will be 0. There are several reasons why this was done. First, the goal is to have the robot arm stabilize to a predefined equilibrium value. This was stated to be zero. Finally, it allows the system to be at a recognized location once the reference trajectory comes into play. The function used was designed such that at $t = 6$, $ref = \mathrm{f(t)} = 0$ and therefore would not be a shock to the system, potentially saving it from falling out of stability. It is clear in Figure 9 when reference tracking switches from being a constant value to the function $f(t)$. The next section will analyze the steady-state error between the reference trajectory and instantaneous q_2 values. In Figure 10 the duration of the simulation was increased to 120 seconds. This better illustrates the desired shape of the function $f(t)$ and adds additional reasoning to the stability of this system. Had the time been extended even more, the 'beating' look that was desired would be even more apparent.

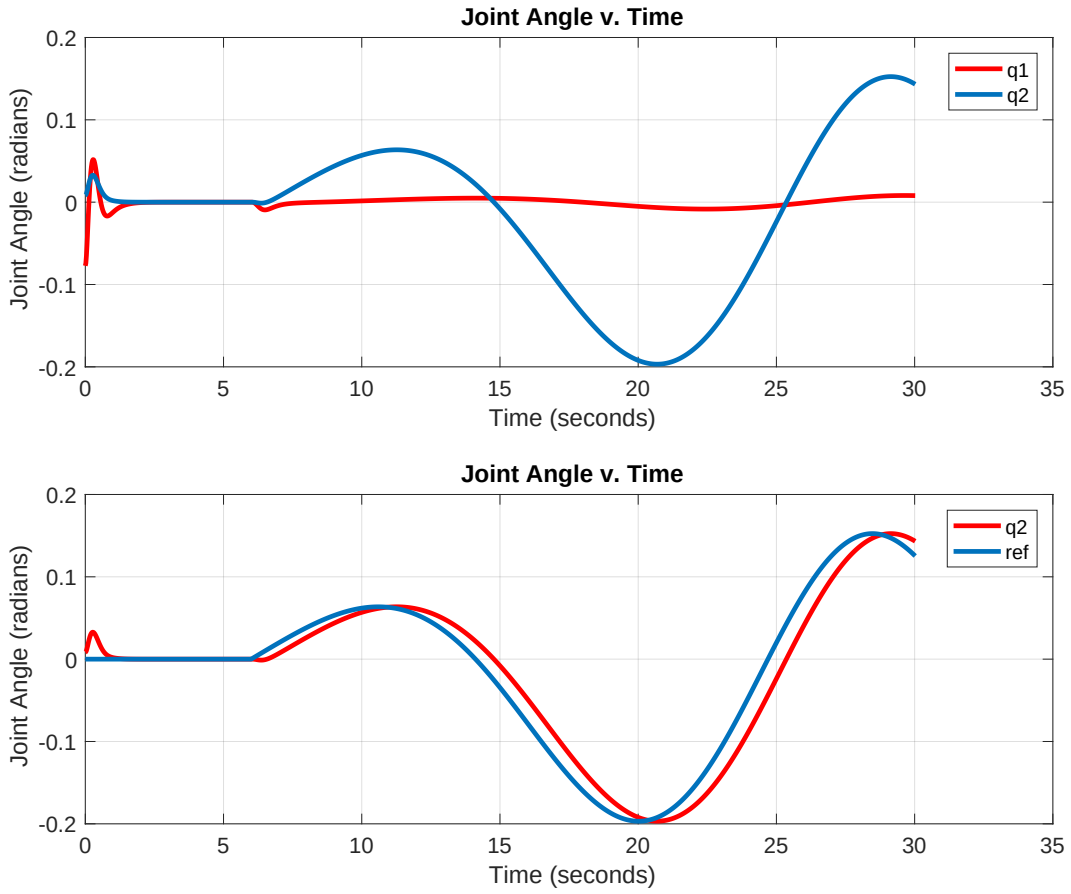


Figure 9: (top) Joint angles when reference tracking is applied. (bottom) Difference between reference and q_2 in time (30sec).

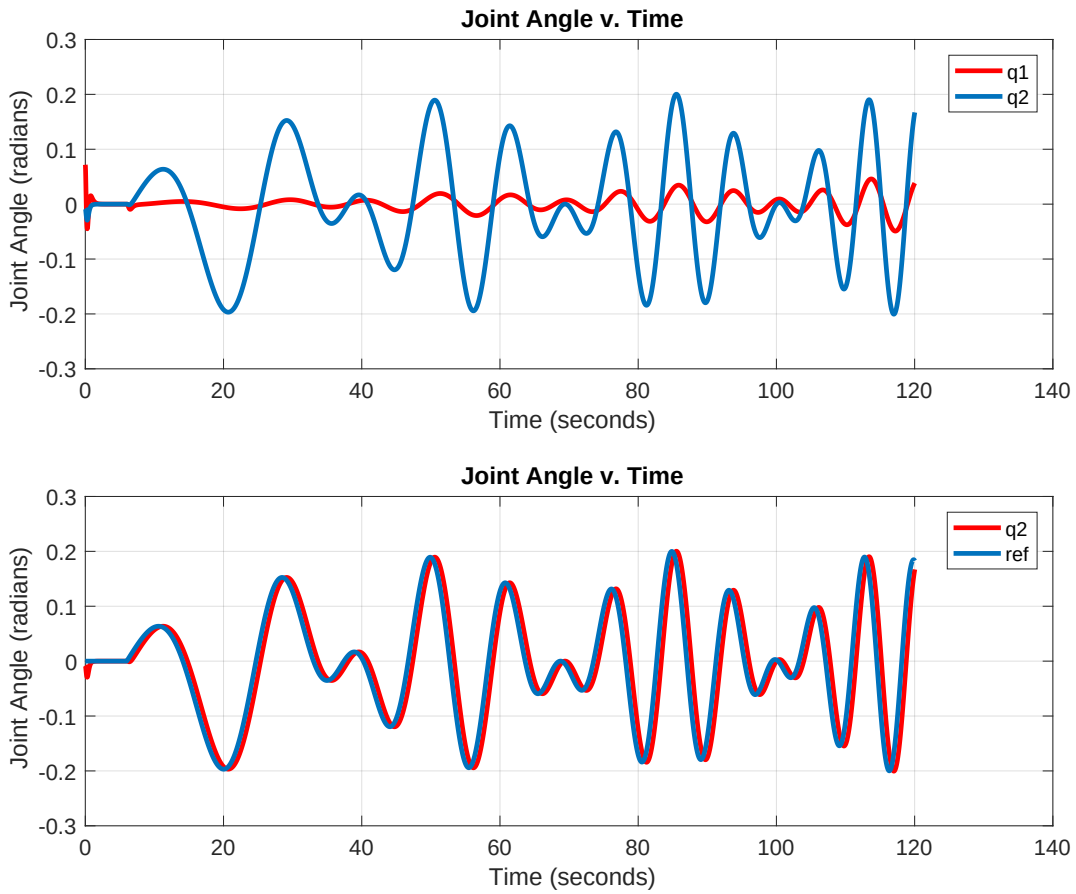


Figure 10: (top) Joint angles when reference tracking is applied. (bottom) Difference between reference and q_2 in time (120sec).

3.4.1 Steady State Error

The controller section concludes with a discussion on steady-state error. Steady-state error is the difference between the steady-state output and the reference point. Since disturbance is simply the differnce between steady-state output and the reference point, it's units will be in radians. However, the goal is established in degrees so a simple conversion must be made. It can be observed in Figure 11 that the steady-state error never exceeds a value greater than five.

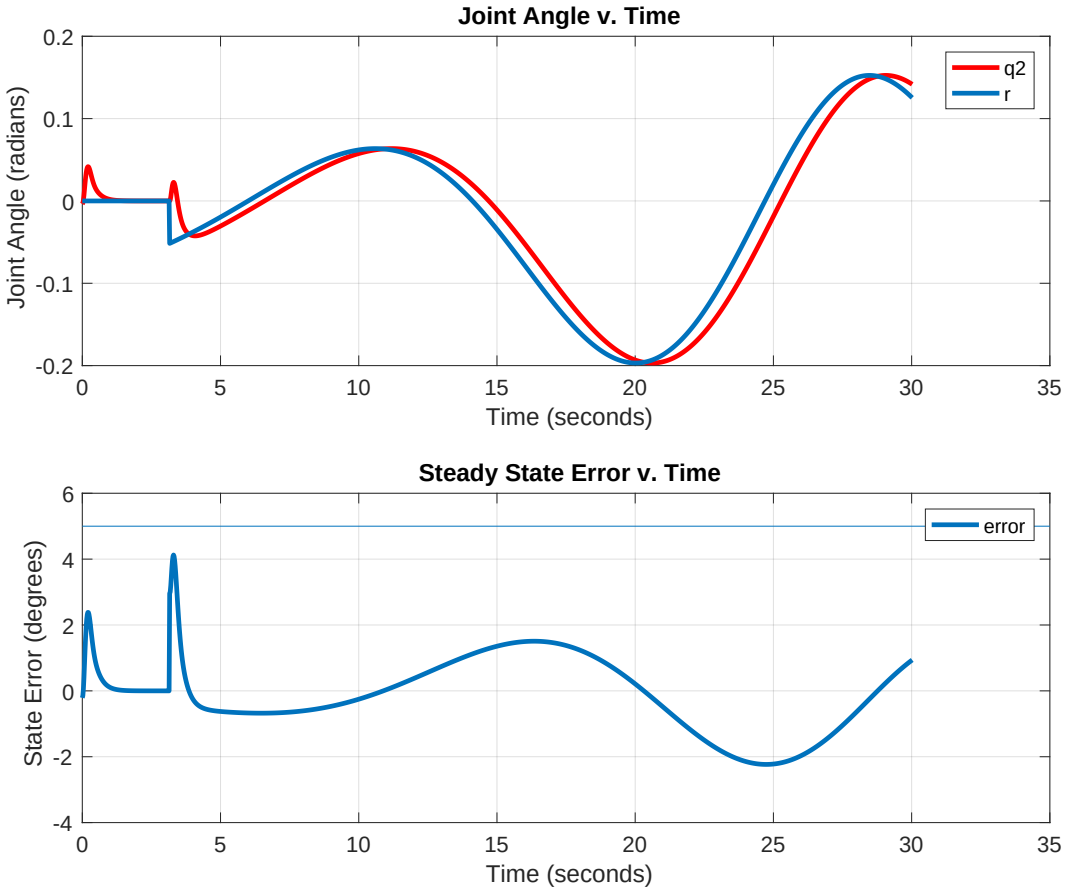


Figure 11: Steady-State error without disturbance

A factor that can be introduced into the system is disturbance. Disturbances are forces external to the system which are unknown and have an effect on the systems output. Disturbances can be accounted for by implementing integral action however, such steps were not taken for this system. By performing simulations with and without disturbance, an obvious increase in steady-state error depicted. It can be seen in Figure 12 that the introduction of disturbance drastically influences the systems ability to reach the desired reference value, thereby increasing the steady-state error.

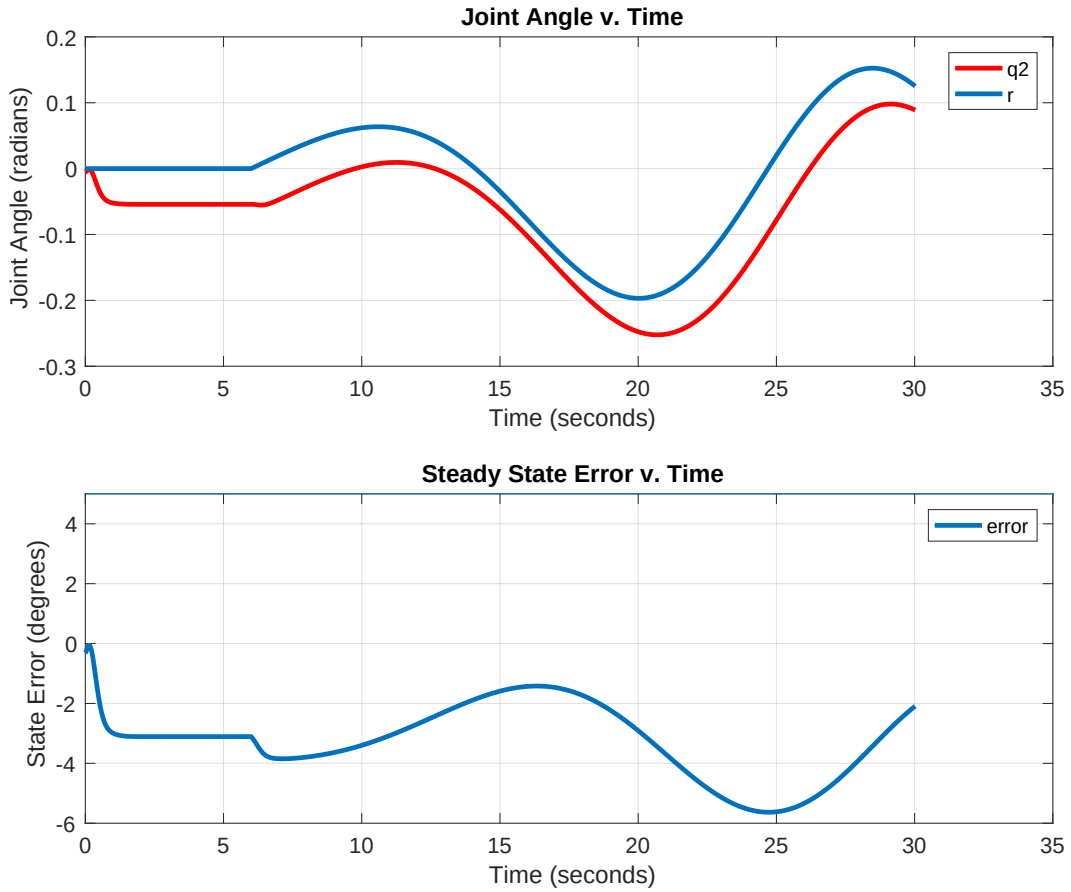


Figure 12: Steady-State error with disturbance

Prior to introducing disturbance, the maximum instance of state-error was less than $\pm 5°$. After introducing disturbance, the maximum instance of steady-state error is greater than $\pm 5°$.

4 Simulation Results

In the first section of this report, a set of instructions was listed for verifying the requibments. To reiterate, these step are as follows

- Devise a function to compute state error at every time interval.
- Plot a steady-state error to time curve. On this curve, include a reference line to mark when your steady-state error has surpassed the value stated by the requirment.
- Analyze the plot. If values exceed $\pm 5°$, the single requirement has been ignored and the simulation has failed.

This procedure was conducted and the results are shown in Figure 13. By observing the plot, it is clear that the since the steady-space error does not surpass $\pm 5°$, the requirement is satisfied.

However, if the timeframe of the simulation is extended as shown in Figure 14, the requirement ends up not being satisfied. This is clearly a result of reference equation chosen since it is expected to get oscillate more rapidly as time advances. It was explained earlier how the structure of the waveform can have effects on the state-error of the function and how the speed of oscillations prevent the system from reaching the reference value fast enough.

An alternative equation using a more simple reference trajectory is shown in Figure 15. It is apparent by looking at the plot that the steady-state error never exceeds $\pm 5°$ thus satisfying the established requirement and proving that the function chosen to be the reference trajectory has an effect on steady-state error.

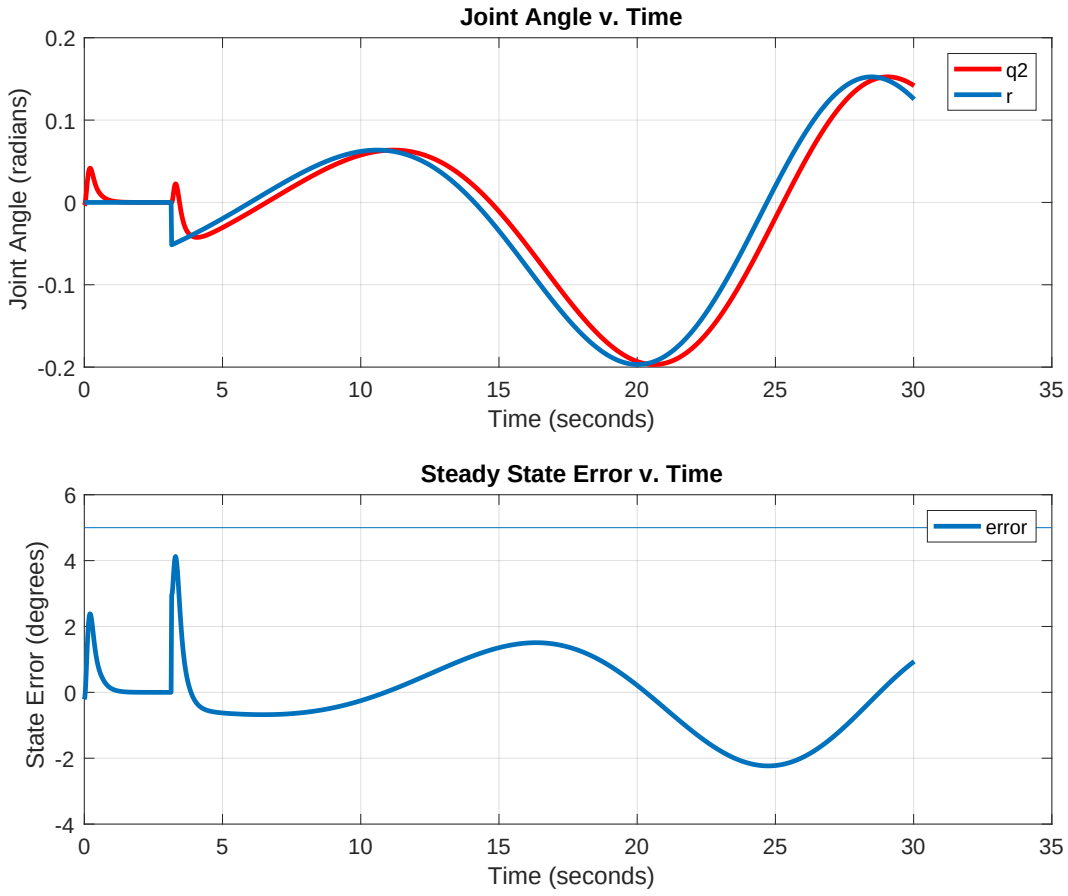


Figure 13: Steady-State Error for 30 seconds. Requirement satisfied (pending)

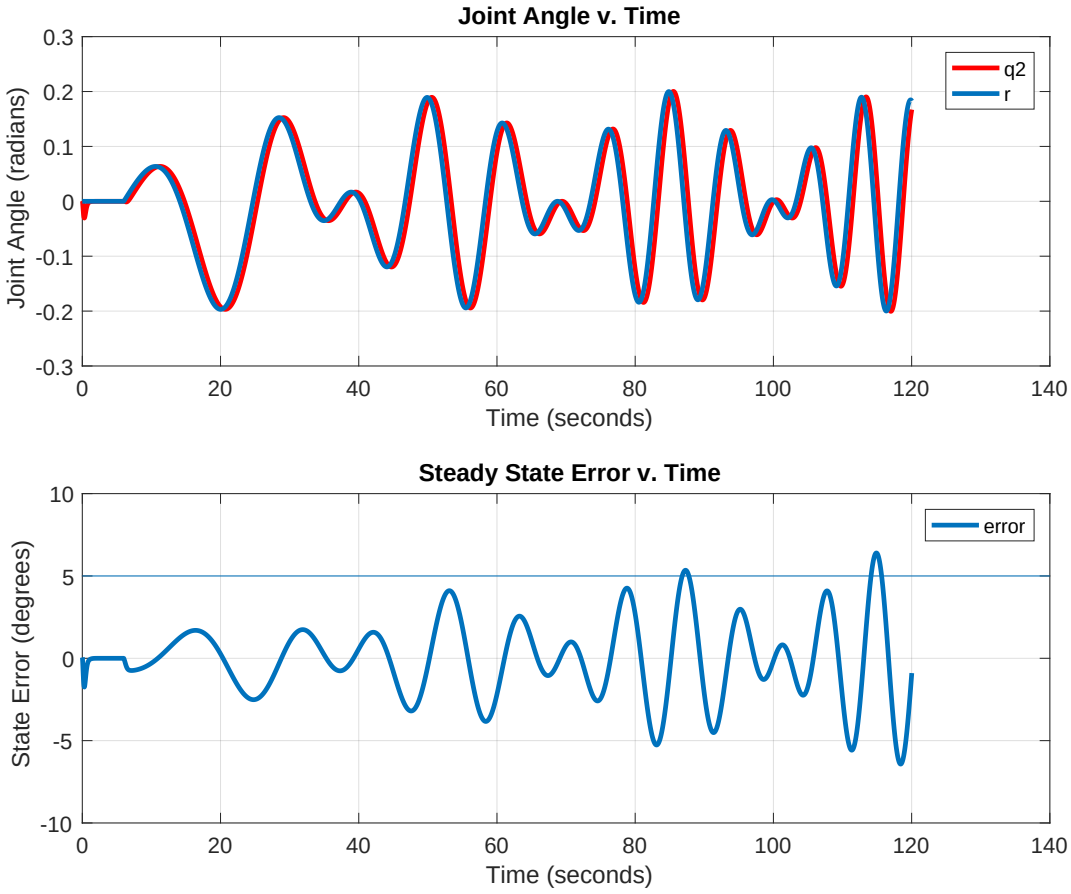


Figure 14: Steady-State Error for 120 seconds. Requirement unsatisfied

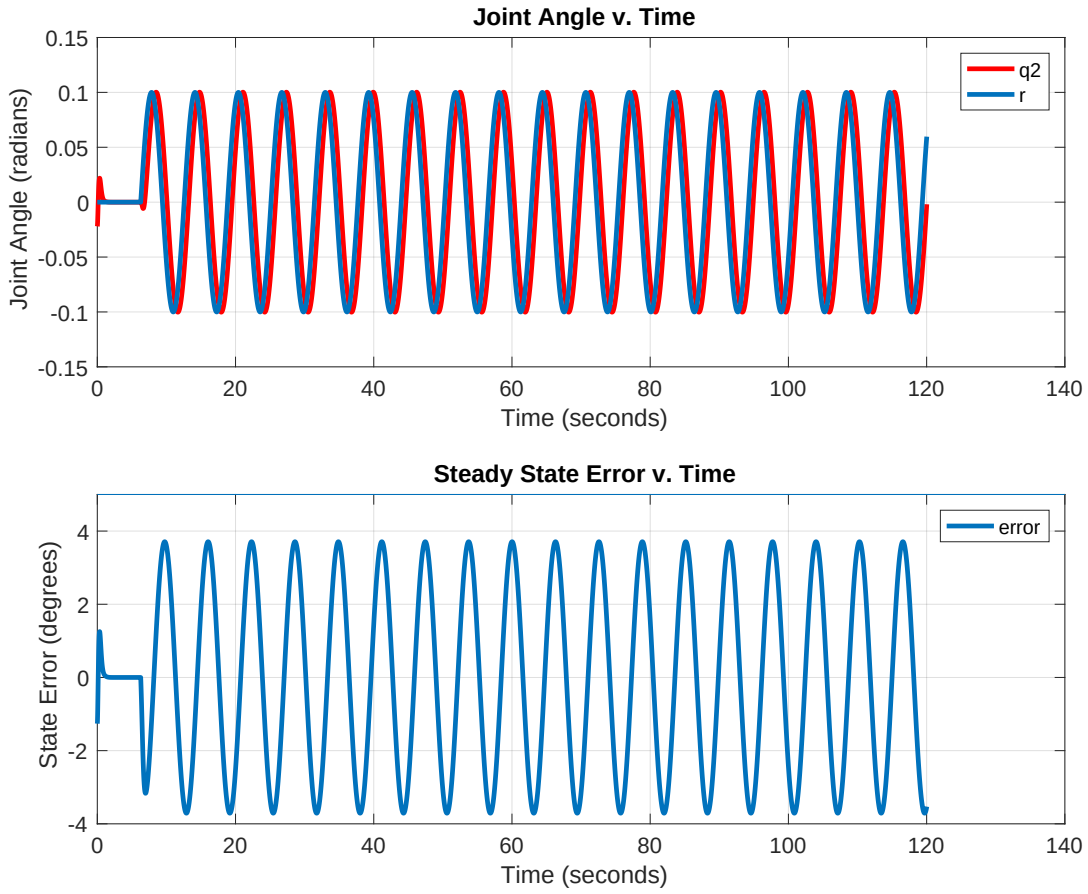


Figure 15: Steady-state error with a reference trajectory of $r(t) = 0.1\sin(t)$. Requirement satisfied (always)

5 Conclusion

In this report, a gravity-assisted under actuated robotic arm was tasked to track a given reference trajectory within a $\pm 5^\circ$ margin of error. A simulation was conducted to determine if this goal was achievable. The requirements set for the model were that the second joint angle shall remain within $\pm 5^\circ$ of its instantaneous reference through-out the duration of the simulation. This requirement was then verified by computing the error at every instance of time and through repeated simulation. The error was then plotted and any points exceeding $\pm 5^\circ$ were noted and signified the requirement was not satisfied. If no point exceeded $\pm 5^\circ$ then the requirement had been satisfied.

The results conclude that this goal is feasible. The robotic arm can remain within a $\pm 5^\circ$ margin of error given certain conditions for the reference trajectory. Achieving this goal required the system to first be linearized. This was made simple through Matlab. Following that, the open-loop linear system was tested and confirmed for controllability using the controllability matrix and rank comparison. The open-loop linear system was used to compute the eigenvalues to determine if it were stable. After the open-loop linear system proved to have been unstable, a closed-loop state feedback controller was implemented. This system used a linear quadratic regulator to determine an optimal K value which in turn produced an efficient way of reaching equilibrium. Once proven to have been stable, reference tracking was implemented which allowed for the continuous tracking of a reference trajectory which is user controlled. Finally, a function for steady-state error was used to verify the results and confirm that the goal is obtainable since the requirements were satisfied (with some exceptions).