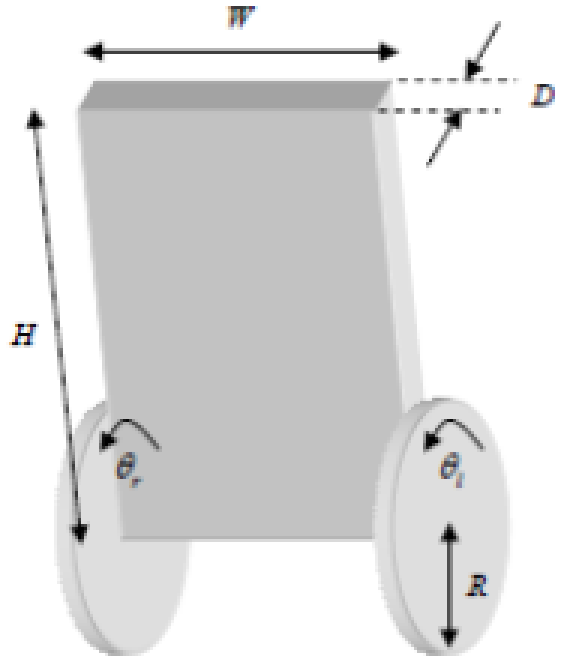
MSE 483

Modern Controls Systems

Group 1: Two-Wheel Self-Balancing Robot



Tony John Kooliyath, 301410719
Artur Shadnik, 301382061
Dave Taye, 301405435

Table of Contents

Project Timeframe

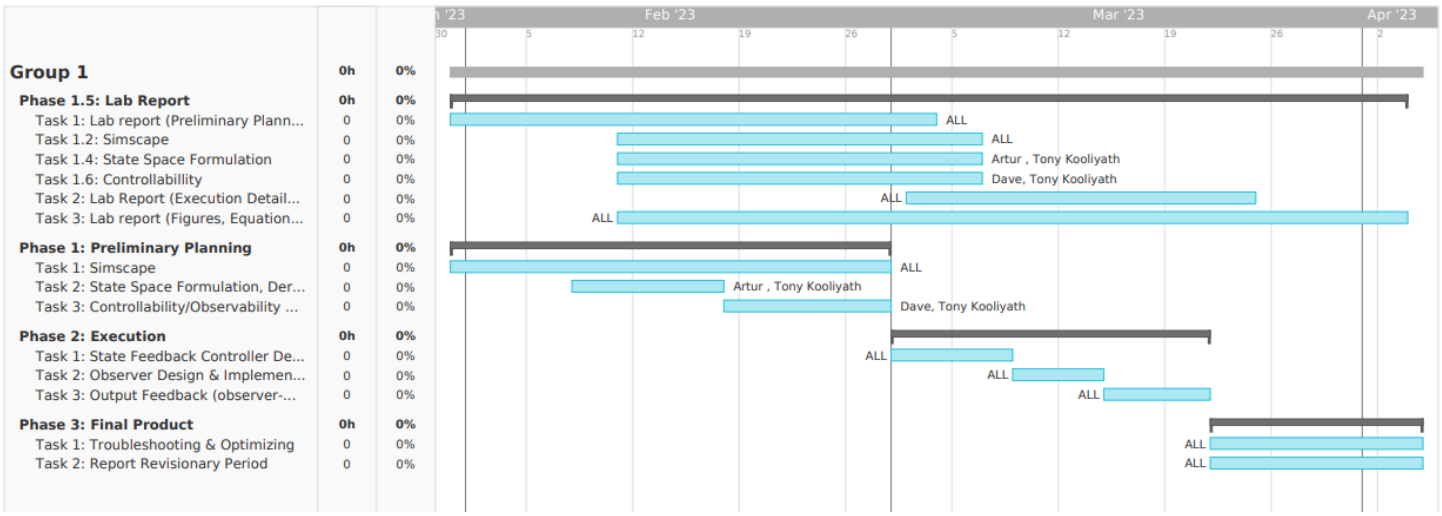


Figure 1: Gantt chart illustrating the division of work and work scheduling

Introduction

The Two Wheeled Self Balancing Robot (TWSBR) operates by utilizing the inverted pendulum principle, which involves maintaining the pendulum in an upright or vertical position relative to the floor. This conventional system is illustrated in Figure 2-1, where the motorized cart moves forward when the pendulum falls forward, and vice versa. To accomplish this, the speed and acceleration of the motorized cart are controlled. As shown in Figure 2-1, the tilt angle, which is the angle formed by the pendulum in relation to the vertical axis, is used as a reference for maintaining the inverted pendulum in its upright position.

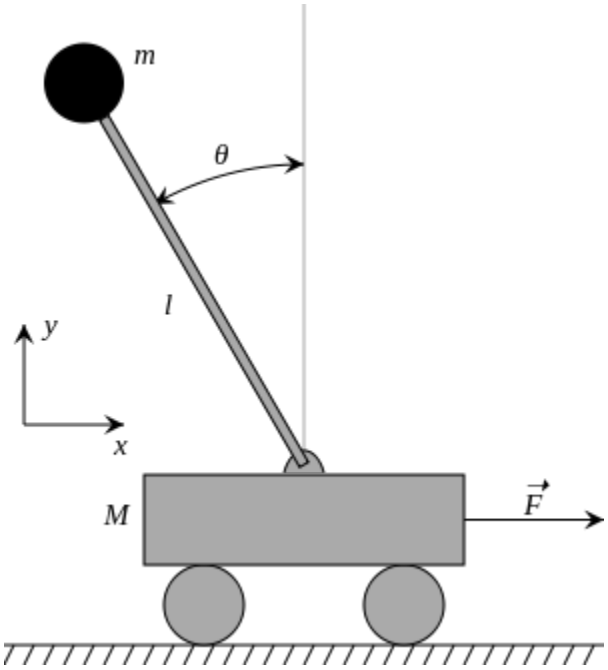


Figure 2: Inverted Pendulum on a cart

The actual design of the TWSBR removes the platform entirely and places the “pendulum” on two wheels. The pendulum can then be replaced by either a platform to carry cargo or humans, or any other number of designs. The system can be controller by applying a torque via the electric motors in each wheel. By applying a very specific amount of torque at the right time, the system can be held upright, or steered to a different linear position as desired by the operator. For the purposes of this project, the system is simplified. Motor are considered as “black box” and motor control is not implemented. The controllable input to the system is taken as torque. The wheels are assumed to be bonded by a massless shaft. The body of the robot is assumed to be a solid block. Figure 2-2 shows the simplified model.

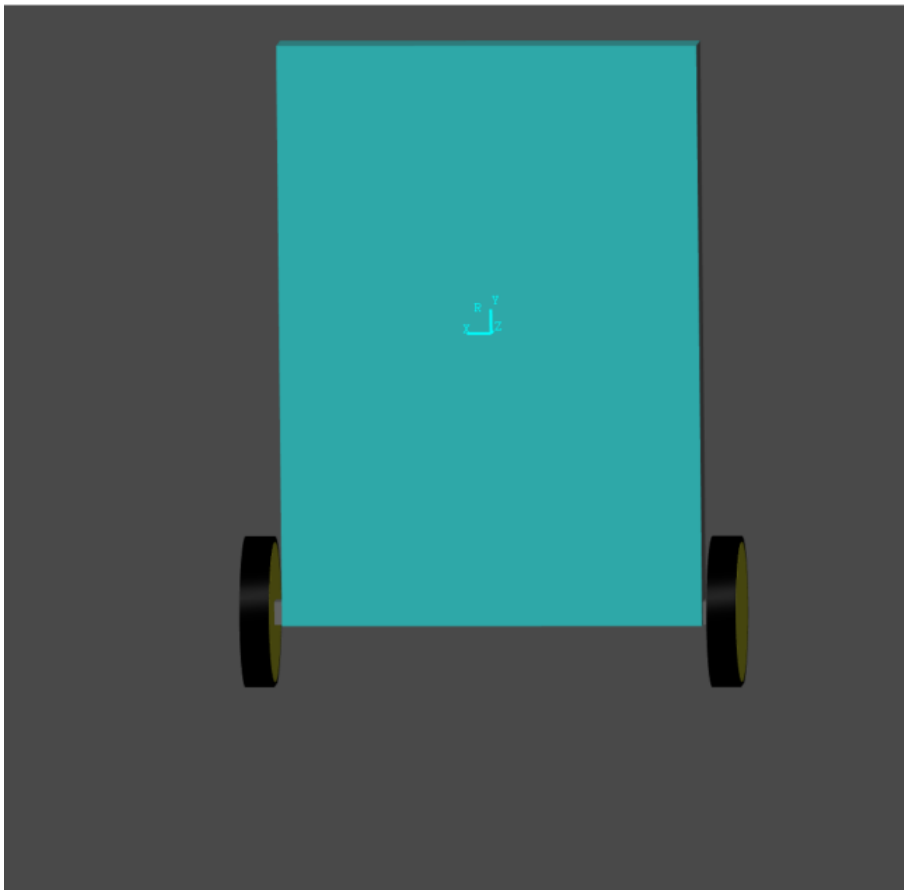


Figure 2-2: Inverted Pendulum on a cart simulation

This report will detail the steps in designing a controller as well as state estimator for the TWSBR. The steps include formulating equations of motion baked on the physics of the system, finding a reasonable operating point for the system, linearizing the system about the operating point to remove any non-linear terms and aid in designing the controller. Once the system is linearized, open-loop stability analysis is performed to analyse the system behaviour and determine what the controller needs to do. Simulink and MATLAB are used to simulate the non-linear and linear systems. Then, desired system performance is determined, and based on this performance, closed-loop controller poles are calculated. The desired poles are used to design a controller by deriving controller gain using Ackermann's formula. The full-state feedback controller is tested on the linear and non-linear system in MATLAB and Simulink. Lastly, since it is impractical to directly measure all the states of a real system, we implement a Luenberger state estimator to observe the internal states of the system. These observed states are then used in conjunction with the previously designed control. The performance of the system is tested on both the linear and non-linear model. A random disturbance is applied to the system to ensure complete stability. Lastly, a Simscape multibody model is developed and tested witht the designed controller to provide visualization of the system and ensure desired performance is being achieved.

Equations Of Motion Formulation

The equations of motion were derived using Newtonian mechanics (Newton's 2nd law). The following system consists of an arbitrary mass (assumed to be rectangular in geometry), as well as two wheels to agitate some form of rotation within the mass. Another assumption made is that the motors act synchronously to eachother, and are electro-mechanically the same. To start, the force equations of a wheel need to be determined. The input into the system will be a supplied torque, whereas the output will be displacement, velocity, angular displacement, and angular velocity.

Wheel:

With regards to the wheel, the following variables are used within the force equation:

m_w	Mass of the wheel(s)
J_w	Moment of inertia acting on wheel
τ_w	Torque of the wheel
$\theta_w, \dot{\theta}_w, \ddot{\theta}_w$	Angular displacement, velocity, and acceleration of the wheel respectively
F_f	Force of friction
F_N	Normal force
F_R	Horizontal reactionary force (e.g. coupling of beam, motor bearrings)
g	Gravitational acceleration (9.981 m/s^2)
$x, \dot{x}, \ddot{x}$	Linear displacement, velocity, and acceleration
r	Radius of the wheel

Resolving the FBD of the wheel into it's respective force components, two equations of motion can be derived. One of these equations correlates to the sum of linear forces in the x direction of the wheel, and the other is the sum of torques acting on the wheel. The beginning step is to formulate Newton's second law. The assumption is that the wheel is rotating clockwise.

$$EQ\ 1.\ \Sigma F_x = ma_x$$

$$EQ\ 2.\ \Sigma\tau = I\alpha$$

Subsitituting the appropriate variables, and account for the sum of force vectors (friction and reactionary) in the x direction into ***EQ 1***, one gets:

$$EQ\ 3.\ F_f - F_R = m_w \ddot{x}$$

$$EQ\ 4.\ x = \theta_w r,\ \dot{x} = \dot{\theta}_w r,\ \text{and}\ \ddot{x} = \ddot{\theta}_w r$$

Subsitituting the appropriate variables, and account for the sum of torques (wheel and rotational friction) in the clockwise direction into ***EQ 2***, one gets:

$$EQ\ 5.\ \tau_w - F_f r = J_w \ddot{\theta}_w$$

Afterwards, the variables F_f and θ_w can be eliminated by substituting ***EQ 3 and EQ 4*** into ***EQ 5*** to get:

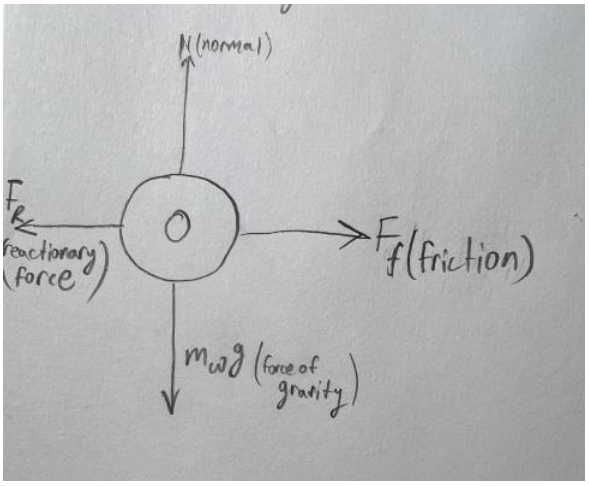
$$\Rightarrow \tau_w - (m_w \ddot{x} + F_R) r = J_w \frac{\ddot{x}}{r}$$

$$\Rightarrow \tau_w - m_w \ddot{x}\, r - F_R r = J_w \frac{\ddot{x}}{r}$$

$$\Rightarrow \tau_w - F_R r = J_w \frac{\ddot{x}}{r} + m_w \ddot{x}\, r$$

$$\Rightarrow \tau_w - F_R r = J_w \frac{\ddot{x}}{r} + m_w \ddot{x}\, r$$

$$\Rightarrow \tau_w - F_R r = \left(\frac{J_w}{r^2} + m_w\right) r \ddot{x}$$



$$\Rightarrow \frac{\tau_w}{r} - F_R = (\frac{J_w}{r^2} + m_w)\ddot{x}$$

Isolating for $\ddot{x}$ yields:

$$\ddot{x} = \frac{(\frac{\tau_w}{r} - F_R)}{(\frac{J_w}{r^2} + m_w)}$$

Body:

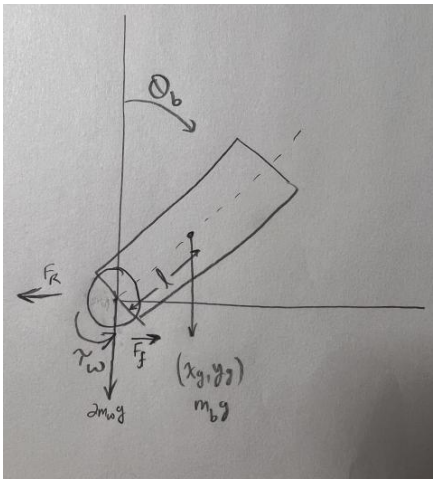
With regards to the arbitrary mass body, the following variables are used within the final dynamic equations:

m_b	Mass of the body
J_b	Moment of inertia acting on body
τ_w	Torque of the wheels
θ_b, $\dot{\theta}_b$, $\ddot{\theta}_b$	Angular displacement, velocity, and acceleration of the body respectively
F_C	Vertical reaction force (e.g. coupling of beam, motor bearrings)
F_N	Normal force
F_R	Horizontal reactionary force (e.g. coupling of beam, motor bearrings)
g	Gravitational acceleration (9.981 m/s^2)
x, $\dot{x}$, $\ddot{x}$	Linear displacement, velocity, and acceleration
r	Radius of the wheel
x_g, y_g	X and Y coordinates of COM of body
l	Length of the body to the COM (A.k.a half the

	length of the body)

Using Newton's law to solve for COM - x component:

$$\Sigma F_x = m_b \ddot{x}_g = 2F_R$$



Solving for $\ddot{x}_g$:

$$x_g = x + l\sin(\theta_b)$$

$$\dot{x}_g = \dot{x} + l\dot{\theta}_b\cos(\theta_b)$$

$$\ddot{x}_g = \ddot{x} + l\ddot{\theta}_b\cos(\theta_b) - l\dot{\theta}_b^{\,2}\sin(\theta_b)$$

$$EQ\ 1 \Rightarrow m_b(\ddot{x} + l\ddot{\theta}_b\cos(\theta_b) - l\dot{\theta}_b^{\,2}\sin(\theta_b)) = 2R$$

Using Newton's law to solve for COM - y component:

$$\Sigma F_y = m_b \ddot{y}_g = m_b g - 2F_C$$

Solving for $\ddot{y}_g$:

$$y_g = l\cos(\theta_b)$$

$$\dot{y}_g = -l\dot{\theta}_b\sin(\theta_b)$$

$$\ddot{y}_g = -l\ddot{\theta}_b\sin(\theta_b) - l\dot{\theta}_b^{\,2}\cos(\theta_b)$$

$$EQ\ 2 \Rightarrow m_b(-l\ddot{\theta}_b\sin(\theta_b) - l\dot{\theta}_b^{\,2}\cos(\theta_b)) = m_b g - 2F_C$$

Solving for the sum of torques:

$$\Sigma\tau = I\alpha$$

$$EQ\ 3 \Rightarrow J_b\ddot{\theta}_b = 2lsin(\theta_b)F_C - 2lcos(\theta_b)F_R - 2\tau_w$$

Substituting $EQ\ 1$ and $EQ\ 2$ into $EQ\ 3$, and using the previously solved wheel equations when isolating for variables $\ddot{\theta}_b$ and $\ddot{x}$ yields the following equations of motion:

$$(J_b + m_b l^2)\ddot{\theta}_b = m_b g l sin(\theta_b) - m_b l cos(\theta_b)\ddot{x} - 2\tau_w$$

$$(m_b + 2(m_w + \frac{J_w}{r^2}))\ddot{x} = -m_b l cos(\theta_b)\ddot{\theta}_b + m_b g l sin(\theta_b)\dot{\theta}_b^2 - \frac{2}{r}\tau_w$$

Linearization of Equations of Motion

In order to develop a state space model, the dynamic equations of the system must be linear. This is because non-linear equations prove to be harder to stabilize due to the presence of oscilliatory functions and square terms, hence the equations needs to be linearized about an operating point. First, this point must be determined. From the problem definition, the initial state $\dot{x}(0)$ is known to be $[0\ 0\ 0\ 0]^T$; the upright position of the robot. At this point since the angle Θ (in this case θ_b) is small, the following approximations can be made:

$$sin(\Theta) \simeq \Theta,\ cos(\Theta) \simeq 1,\ \omega = 0$$

Linearizing the equations and isolating with the previous oscillatory approximations yields the following:

$$\ddot{x} = (m_b + 2(m_w + \frac{J_w}{r^2}) - \frac{(m_b l)^2}{m_b l^2 + J_b})^{-1} \bullet (-\frac{(m_b l)^2 g\theta_b}{m_b l^2 + J_b} + 2(\frac{1}{r} + \frac{m_b l}{m_b l^2 + J_b})\tau_w)$$

$$\ddot{\theta}_b = (J_b + m_b l^2 - \frac{(m_b l)^2}{m_b + 2(m_w + \frac{J_w}{r^2})})^{-1} \bullet (m_b l g\theta_b - 2(r + \frac{m_b l}{m_b + 2(m_w + \frac{J_w}{r^2})})\frac{\tau_w}{r})$$

State Space Model

In order to design a proper feedback control system, a a sate space model representation of the linearized system is required. This is because it's a useful mathematical framework describe systems variables and their changes over time. It will allow for model predictions, estimations, controllability, and obsersavility. The standard notation of the state space model will be:

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

Where the state variables will be:

$$[x_{body},\ \dot{x}_{body},\ \Theta_{body},\ \dot{\Theta}_{body}]^T$$
$$where\ d_{body} = r * \Theta_{wheel}\ \ and\ \ \omega_{body} = \dot{\Theta}_{body} = r * \omega_{body}$$

Furthermore, the linearized state space can be described in matrices:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -c1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & c3 & 0 \end{pmatrix} = A \qquad \begin{pmatrix} 0 \\ c2 \\ 0 \\ -c4 \end{pmatrix} = B \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = C \qquad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = D$$

Where c1, c2, c3, and c4 are:

$$c1 = \frac{(m_b l)^2 g}{(m_b l)^2 + J_b} \bullet \frac{1}{m_b + 2(m_w + \frac{J_w}{r^2}) - \frac{(m_b l)^2}{m_b l^2 + J_b}}$$

$$c2 = (\frac{1}{r} + \frac{m_b l}{(m_b l)^2 + J_b}) \bullet \frac{2}{m_b + 2(m_w + \frac{J_w}{r^2}) - \frac{(m_b l)^2}{m_b l^2 + J_b}}$$

```
% System constants
mb = 0.2445;
mw = 0.0343;
R = 0.035;
l = 0.035;
Jb = 0.5 * mb * l^2;
Jw = 0.5 * mw * R^2;
g = 9.81;
```

$$c3 = \frac{m_b l g}{J_b + m_b l^2 - \frac{(m_b l)^2}{m_b + 2(m_w + \frac{J_w}{r^2})}}$$

$$c4 = \frac{2}{r(J_b + m_b l^2 - \frac{(m_b l)^2}{m_b + 2(m_w + \frac{J_w}{r^2})})} \bullet (r + \frac{m_b l}{m_b + 2(m_w + \frac{J_w}{r^2})})$$

- The linearized state space system can be expressed as:

$$\ddot{x} = -c1(\theta_b) + c2(\tau_w) \;\&\; \ddot{\theta}_b = c3(\theta_b) - c4(\tau_w)$$

The following matlab code snippet and simulink model illustrates the state space model as an open-loop system:

```
% Linearized state space system definition

c1 = (mb^2 * l^2 * g)/((mb + 2 * (mw + (Jw/R^2) - (mb^2 * l^2)/(mb * l^2 + Jb))*(mb^2 * l^2 * g)/(mb * l^2 + Jb)));
c2 = (2/(mb + 2 * (mw + Jw/R^2) - (mb^2 * l^2)/(mb * l^2 + Jb))) * (1/R + (mb * l)/(mb * l^2 + Jb));
c3 = (mb * g * l) * (1/Jb + 1/(mb * l^2) - ((mb + 2 * (mw + Jw/R^2)/(mb^2 * l^2))));
c4 = (2/R) * (1/Jb + 1/(mb * l^2) - ((mb + 2 * (mw + Jw/R^2)/(mb^2 * l^2)))) * (R + ((mb * l)/(mb + 2 * (mw + Jw/R^2))));

A = [0 1 0 0; 0 0 -c1 0; 0 0 0 1; 0 0 c3 0];
B = [0; c2; 0; -c4];
C = eye(4);
D = [0; 0; 0; 0];

sys_ol = ss(A,B,C,D);
```

Figure 3: Matlab open-loop linearized code snippet

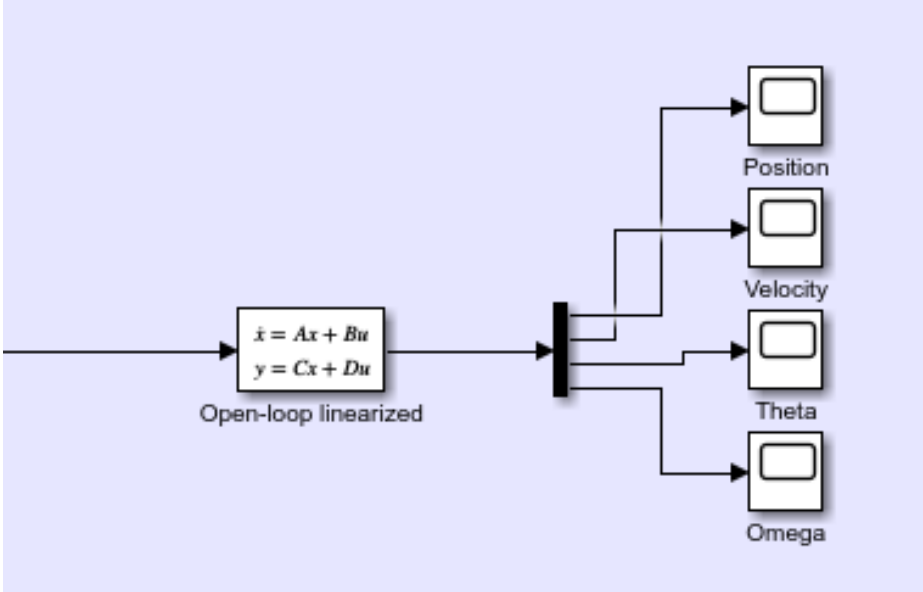


Figure 4: Simulink open-loop linearized model

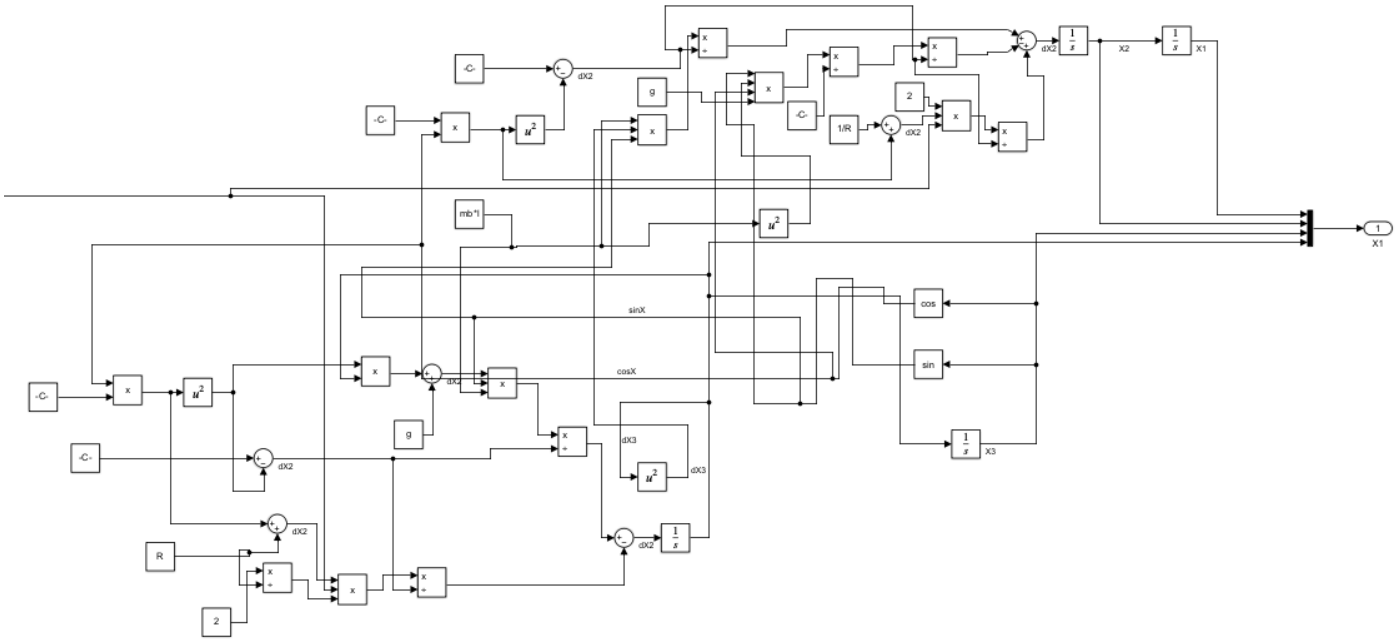


Figure 5: Non-linear system definition in Simulink

Open-loop Stability Analysis

It is important to evaluate the stability of the open-loop system to determine a baseline before beginning the design of the control system. To analyse the stability of the open-loop linearized system, the poles of the system are examined. The MATLAB functions `pole` and `pzmap` are used to calculate the open loop poles and plot a pole-zero map for the system. Figure 6 shows a pole-zero map for the open-loop system. The open-loop poles are found to be [-26.8864, 0, 0, 26.8864].

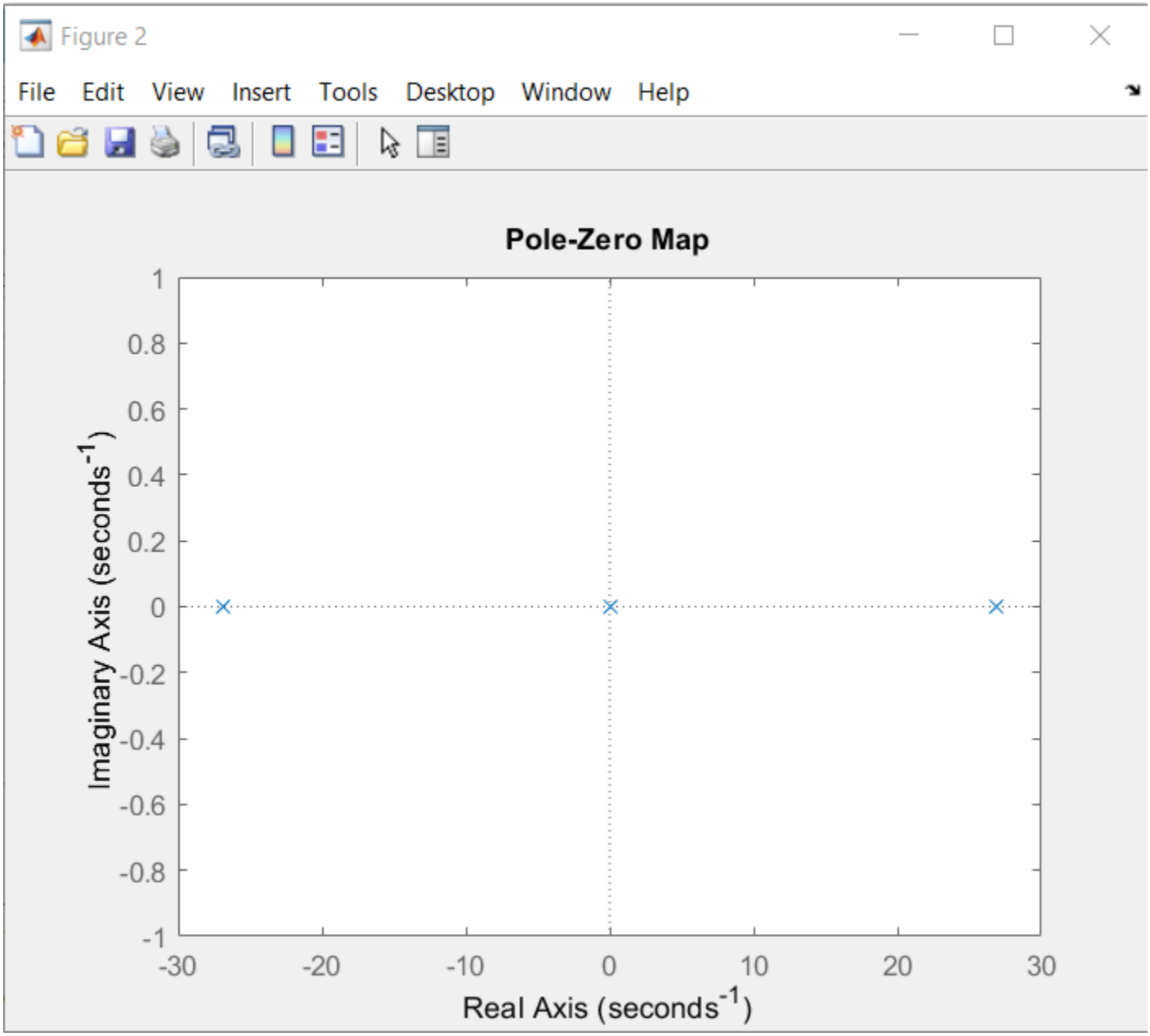


Figure 6: Pole-Zero mapping of open-loop system

Due to the presence of a pole in the positive real quadrant (s = 26.8864), the open-loop system is unstable. We can intuitively verify this, since we know a body on 2 wheels is not stable without outside support. This justifies the development of a control system to balance the 2 wheeled robot when an outside disturbance is applied.

The system response to a step input is simulated using both Simulink and MATLAB `step` function. As expected based on the stability analysis, all 4 states either exponentially grow or decay to + or - ∞. Figures 7, 8 and 9 below show the linearized MATLAB simulation, the linearized Simulink simulation and the non-linearized Simulink simulation respectively. In the simulink plots, the yellow line shows the trajectory of position, the blue line shows the trajectory of velocity, the red line shows the trajectory of angular displacement and the green line shows the trajectory of angular velocity.

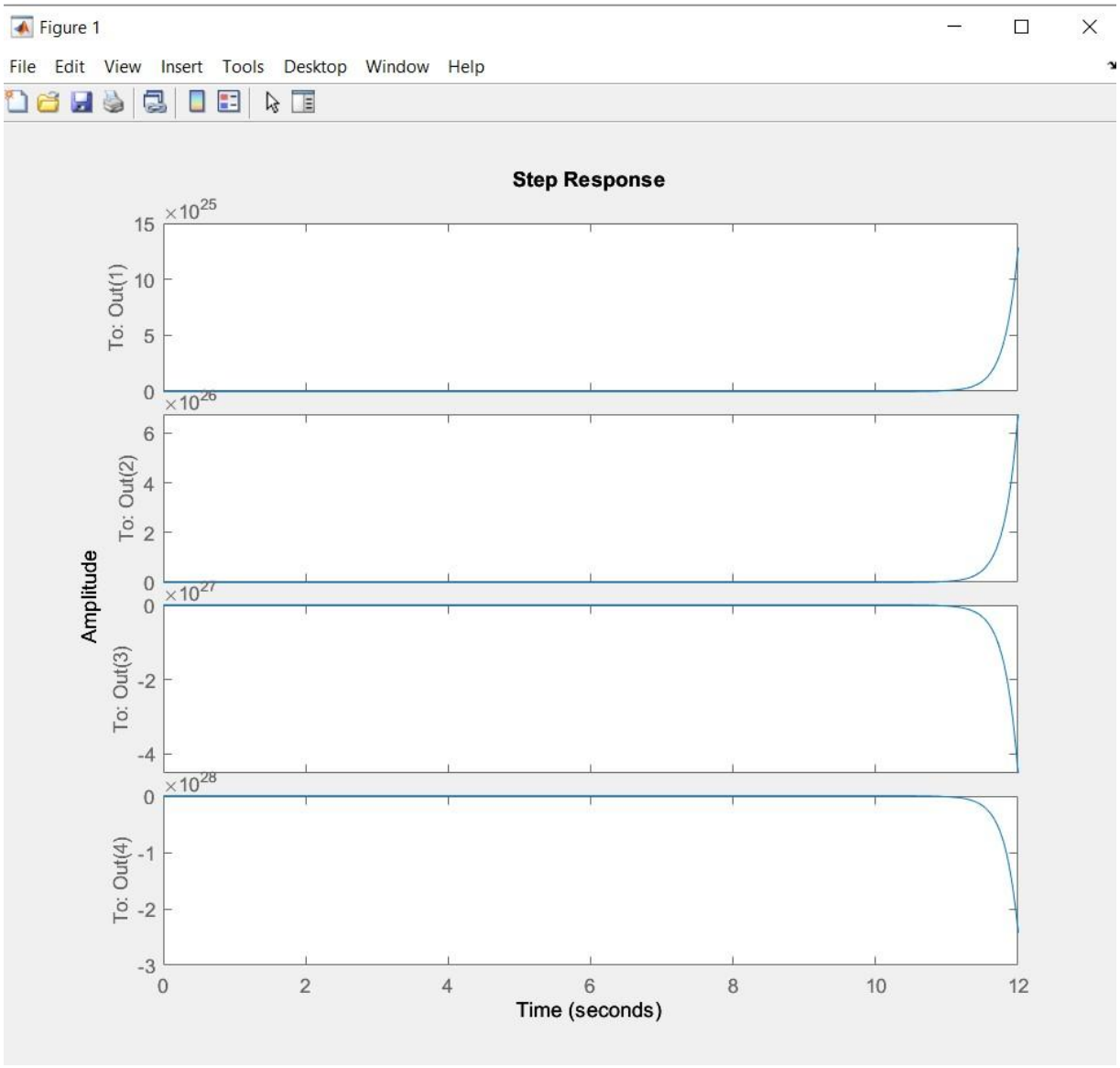


Figure 7: Open-loop response of linearized system

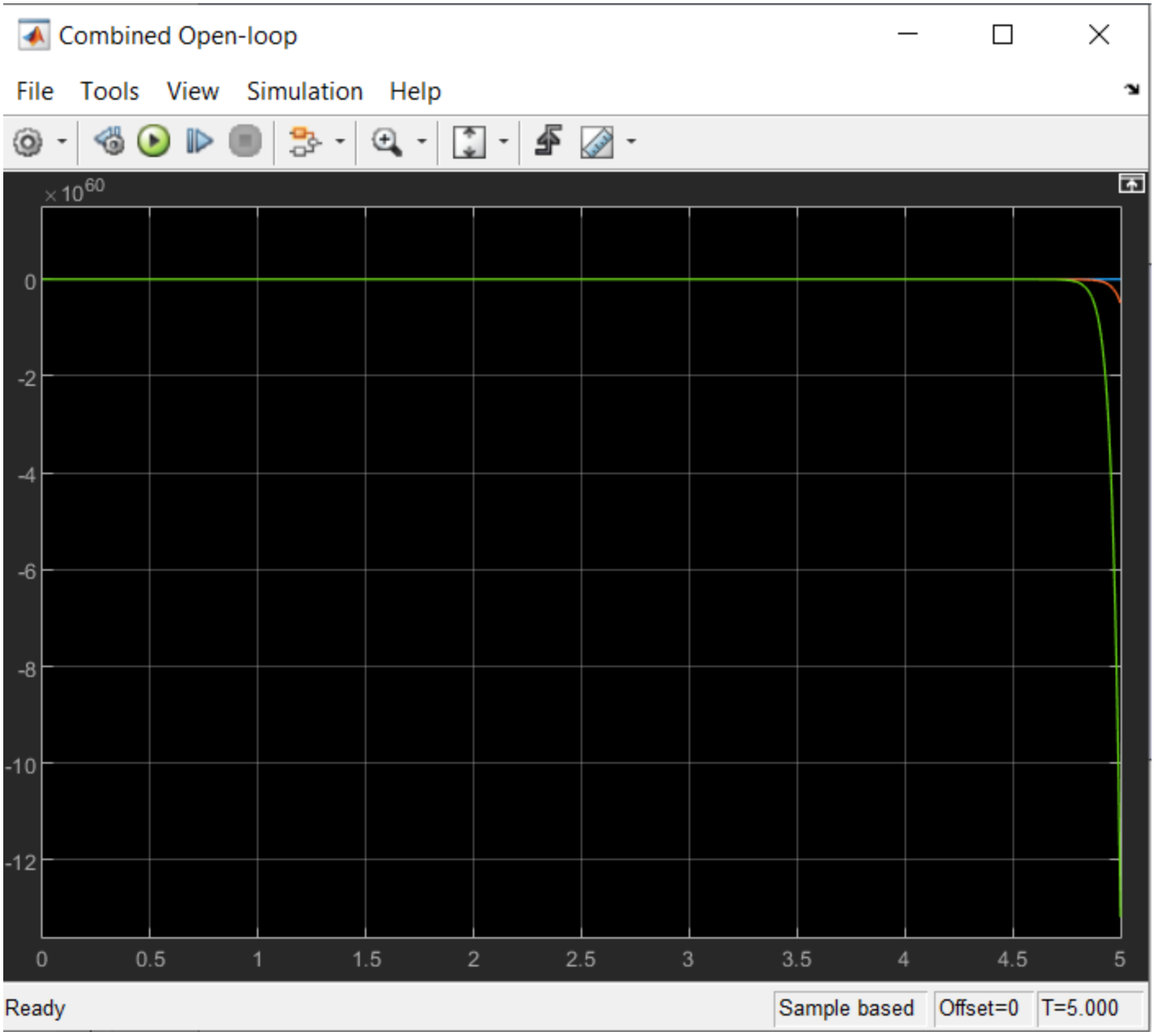


Figure 8: Simulink open-loop response of linearized system

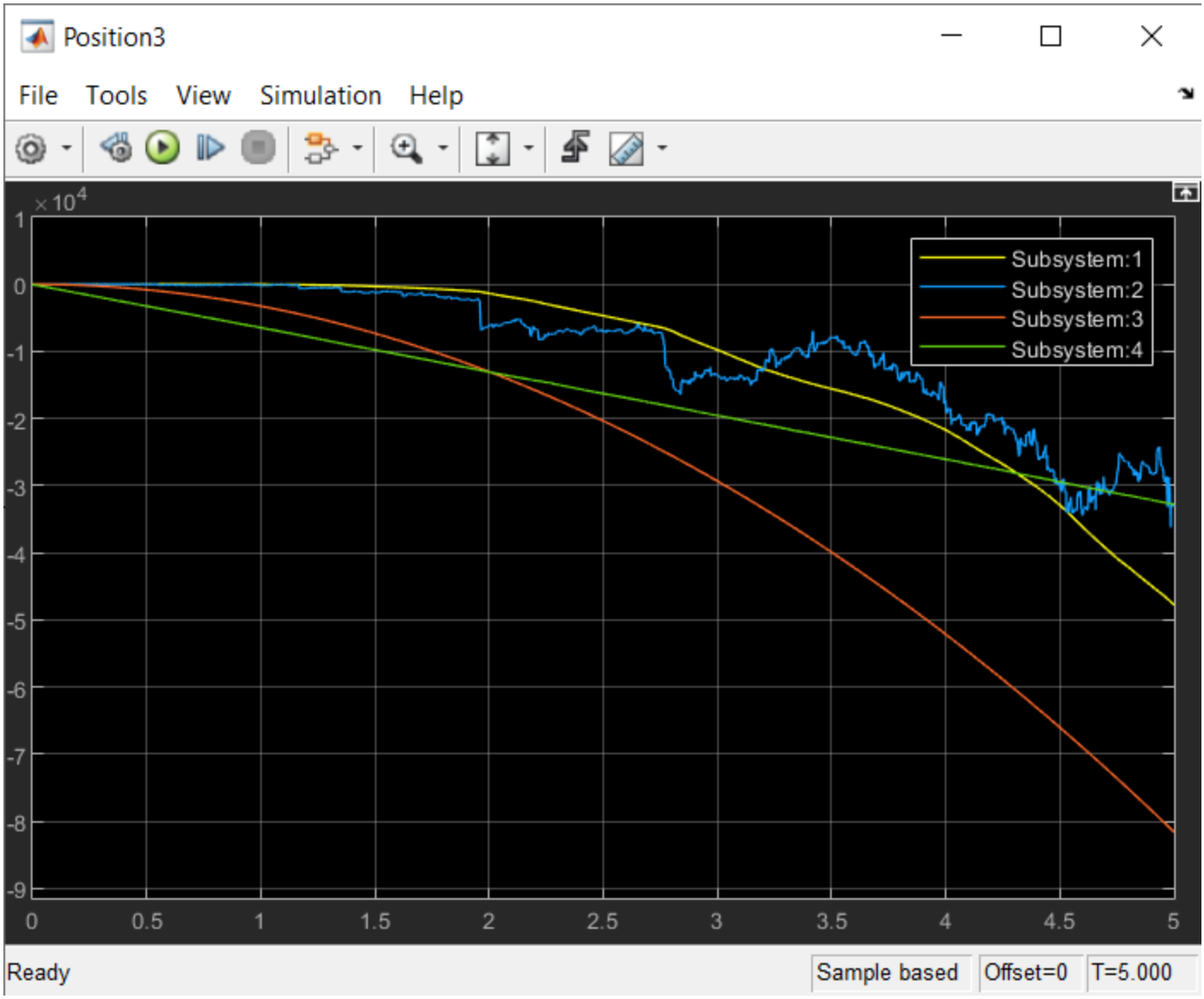


Figure 9: Simulink open-loop response of non-linearized system

The time scales on the MATLAB and Simulink are slightly different which explains the slight variation in the curves. We can see that the non-linear system becomes much more unpredictable as the simulation time increases. Furthermore, it appears to be stochastic in nature. This is because of higher order and oscillating terms within the state space, which affects the response of the system.

Controllability and Observability Analysis

Having established the need for a controller to stabilize the system, we must now determine whether the system is controllable as well as observable. If the system is not controllable, we will not be able to successfully apply a controller to achieve a desired final state, and if the system is not observable we will not be able to design a state estimator to observe states which we cannot directly measure.

First, we check the controllability of the system. Controllability of a 4th order system is given by the formula P = [B AB A^2B A^3B]. If the rank (maximum number of linearly independent column/row vectors) of this matrix is equal to the number of state variables $\boldsymbol{x}$, then it is said to be controllable. Using MATLAB function `ctrb` we find the controllability matrix to be:

P = [

```
    1.0e+07 *

         0    0.0001         0   -0.0000
    0.0001         0   -0.0000         0
         0   -0.0029         0   -2.1211
   -0.0029         0   -2.1211         0
```

]

When examining the rank of the P it's obvious to see that all 4 rows are independent of one another, so the rank of the matrix P is 4. Therefore it is concluded that the system is controllable.

Next we check the observability of the system. The observability of a 4th order system is given by the formula Q = [C CA CA^2 CA^3]^T. Similar to controllability, if the rank of matrix Q is equal to the order of the system, it is said to be observable. Using MATLAB function `obsv` we find the observeability matrix to be:

Q = [

```
1.0e+05 *

 0.0000         0         0         0
      0    0.0000         0         0
      0         0    0.0000         0
      0         0         0    0.0000
      0    0.0000         0         0
      0         0    0.0000         0
      0         0         0    0.0000
      0         0    0.0072         0
      0         0    0.0000         0
      0         0         0    0.0000
      0         0    0.0072         0
      0         0         0    0.0072
      0         0         0    0.0000
      0         0    0.0000         0
      0         0         0    0.0072
      0         0    5.2255         0
```

]

By using the MATLAB `rank` function, we see that the rank of the observability matrix is 4, the same as the order of the system. This means the system is also observable.

Since the system is both controllable and observable, we can proceed with designing a controller.

State Feedback Control

In order to design a suitable feedback control system for the system, it's important to identify the necessary performance criteria to achieve the task at hand. Likewise, the most important key factors are:

- Steady state error: The error difference of target position and actual position of the robot given that the robot is in a steady-state (no inputs/disturbances after initial perturbation). It's important to minimize this so that the robot will maintain the upright position reliably and effectively. $e(\infty) \leq 5\% \, of \, target \, value$

- Overshoot: The amount of corrective behaviour the system employs in order to converge to a desired state/value. Minimizing this is key to ensure that oscillations are kept to a minimum and the system doesn't become unstable. $\%O.S. \leq 5\%$
- Settling time: The amount of time is takes to settle down to a steady state/value. This is important to minimize so that the robot can apply corrective measures immediately, without compromising it's balance in the upright position. $T_s \leq 2\ seconds$
- Control effort: The amount of energy required (torque) to keep the robot upright in the system. This should be minimized to improve reliability and efficiency of the design. $Don't\ care\ in\ this\ case\ since\ torque\ is\ already\ relatively\ minimal$

Minimizing each of these attributes simultaneously results in trade-offs for the system (e.g. reduce overshoot can result in higher settling time, etc.). Likewise, we've established less conservative requirements for improved flexibility so that all attributes can be optimized (ignoring control effort). The first method used to design the controller is state feedback. The 4 states of the system are measured directly and the sensors are assumed to have no noise. A controller gain matrix is applied to the measured states, and this is subtracted from the reference signal to obtain a new signal u(t), which is then given as input into the plant. The resulting closed-loop dynamics are given by

$$\dot{x} = (A - B * K)x$$

Where K is the controller gain matrix.

One method for finding the gain values for the K matrix is to first determine the desired location of the closed loop poles of the system which will provide the desired control. After the poles are chosen, they are used to find the gain using Ackermann's formula

$$K = [0\ 0\ 0\ 1]C^{-1}\Delta_{new}(A)$$

Where

$$\Delta_{new}(A) = det(SI - (A - B * K))$$

Which is the new desired characteristic polynomial. The desired poles are found by examining the system response, settling time and overshoot with a variety of different poles. Ultimately, it is found that [-3, -6, -30 -50] provide acceptable settling time as well as overshoot. With these poles, the K gain matrix is computed to be [-0.0723, -0.0400, -0.1022, -0.0037]. The new closed loop system dynamics are found to be

A_{CL} = [

```
  1.0e+03 *

         0    0.0010         0         0
    0.0373    0.0207    0.0528    0.0019
         0         0         0    0.0010
   -2.1209   -1.1736   -2.2753   -0.1097
                                         ]
```

The closed loop state feedback system is implemented in Simulink as follows.

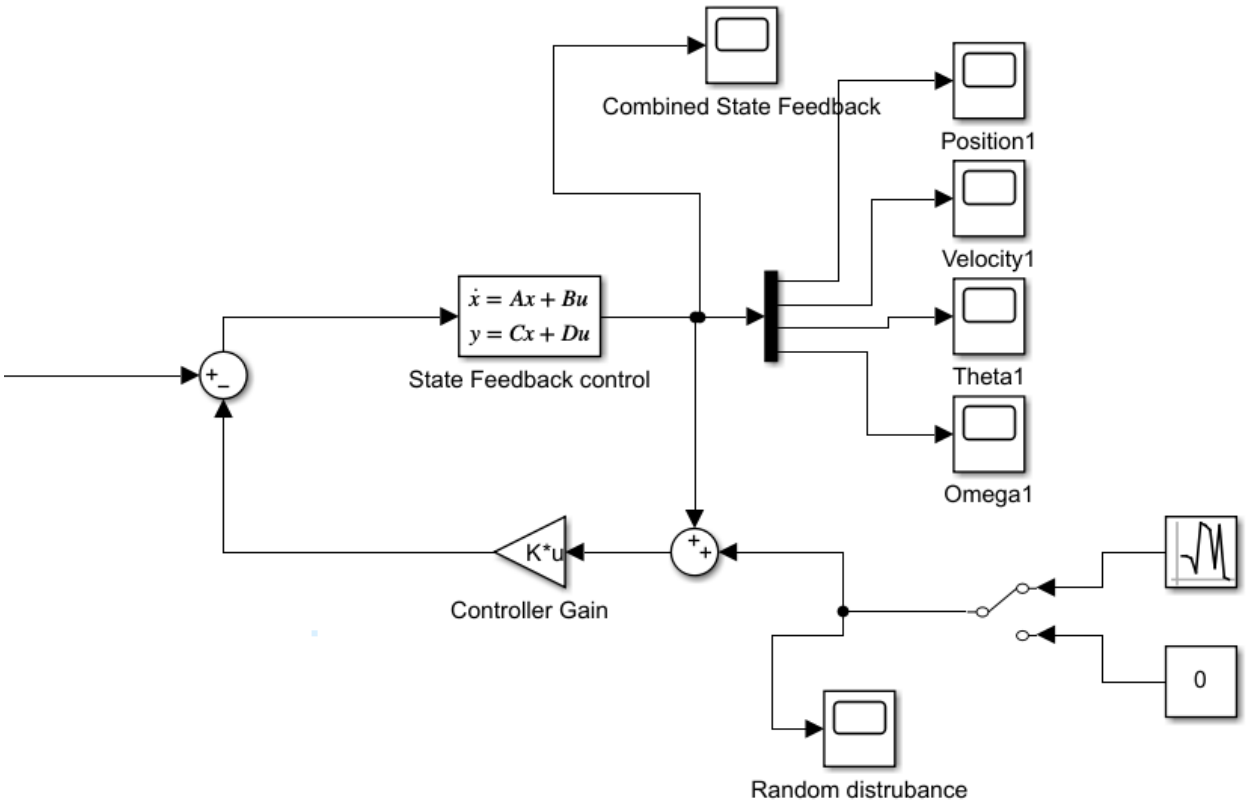


Figure 10: Closed loop state feedback system of linearized model in Simulink

The state we are most interested in is x3 = θ. We want this value to settle to 0 as quickly as possible. With the chosen poles, x3 can be controlled to 0 in 1.58 seconds.

The MATLAB and Simulink plots of the controlled system response to a step input are shown in Figures 11 and 12 below.

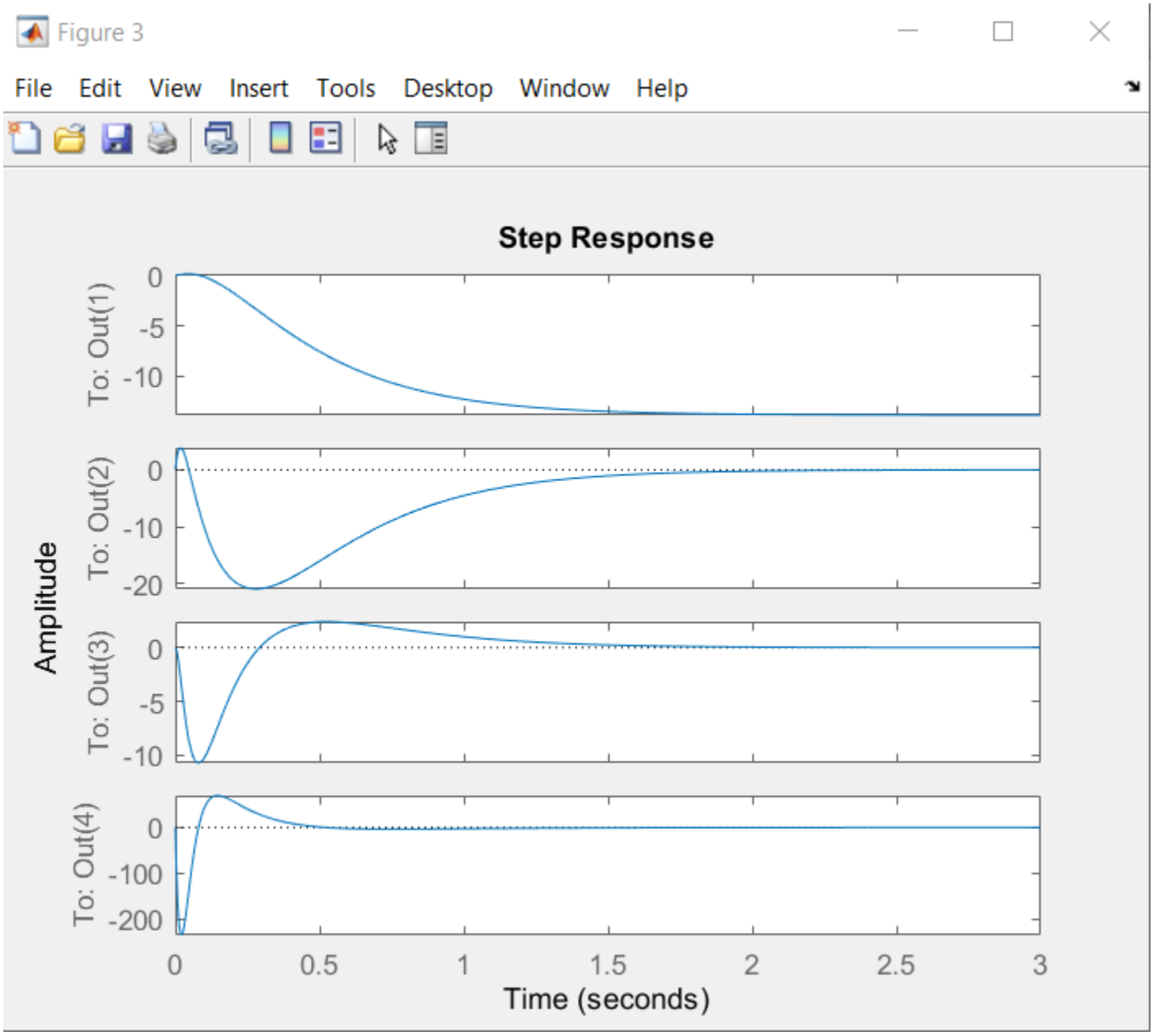


Figure 11: Step response of Closed loop state feedback system of linearized model

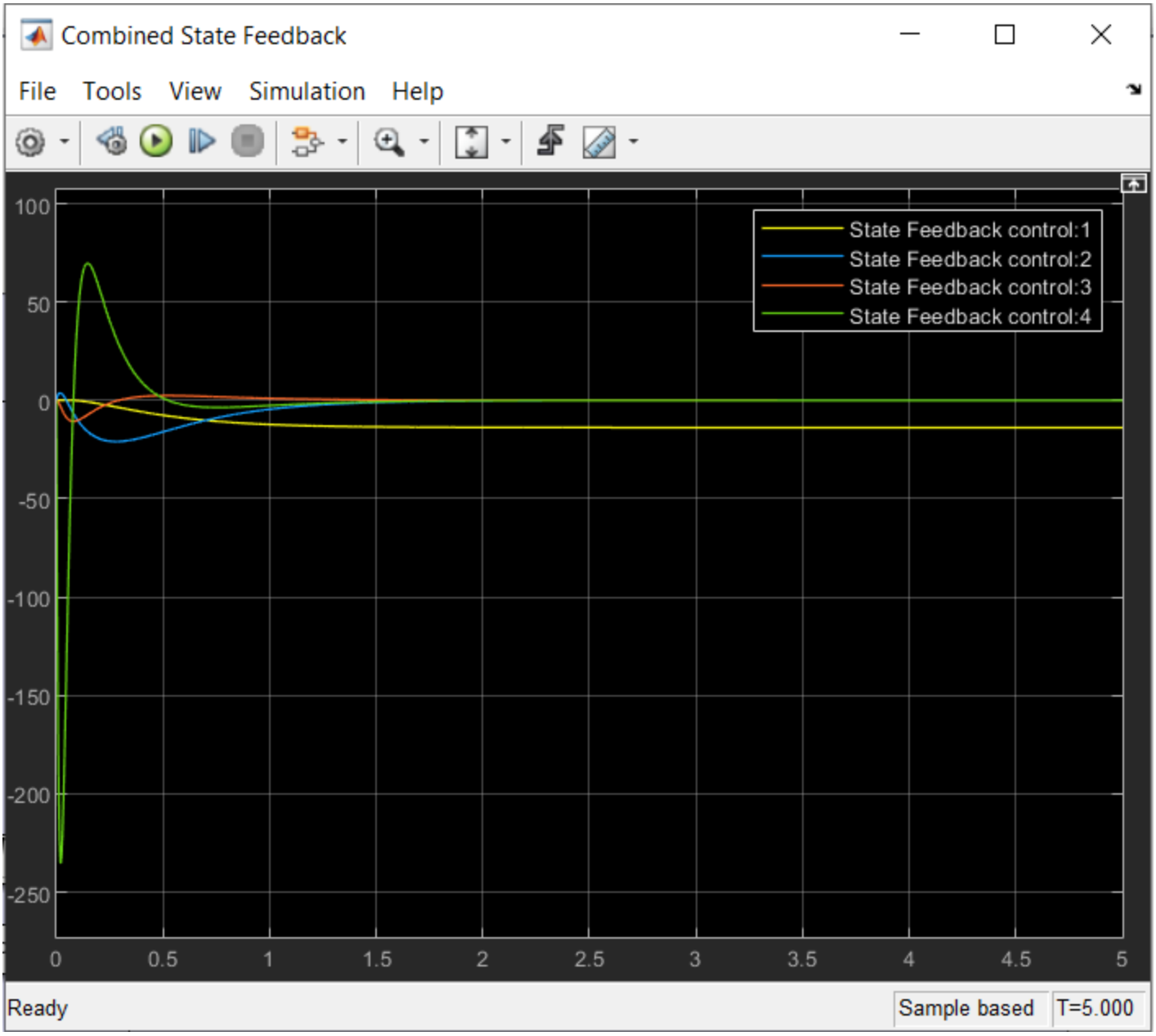


Figure 12: Step response of Closed loop state feedback system of linearized model in Simulink

When compared to the open-loop response presented earlier, we can see clear improvements in the system response. Rather than growing or collapsing exponentially, all 4 states settle to steady state within 3 seconds. We can see that the only state which does not reach 0 is x1, position. Rather, it settles at a slightly negative number, when the other 3 states reach 0. Intuitively this makes sense, as the robot will need to move around slightly in order to regain balance after the disturbance, and in doing so it will have changed position slightly.

To ensure the system would be able to control itself back to an upright position from any potential outside disturbance, we simulated a random disturbance to the states of the system. This was done by adding a random zero mean signal that changes every 1.5 seconds. The disturbance signal is shown in Figure 13 below.

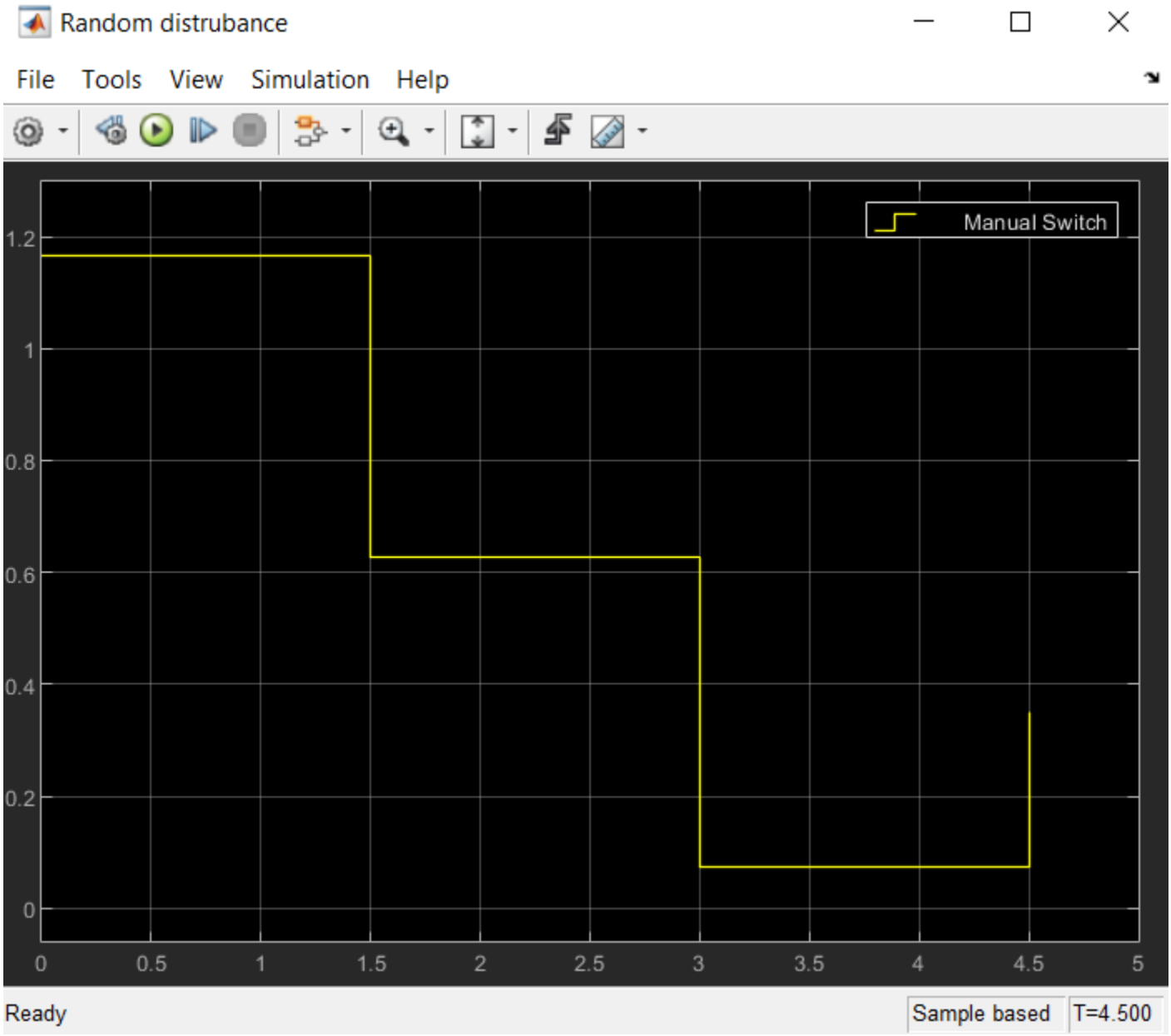


Figure 13: Random step disturbances applied to the system

The closed loop, state feedback based control response of the system is shown in Figure 14 below.

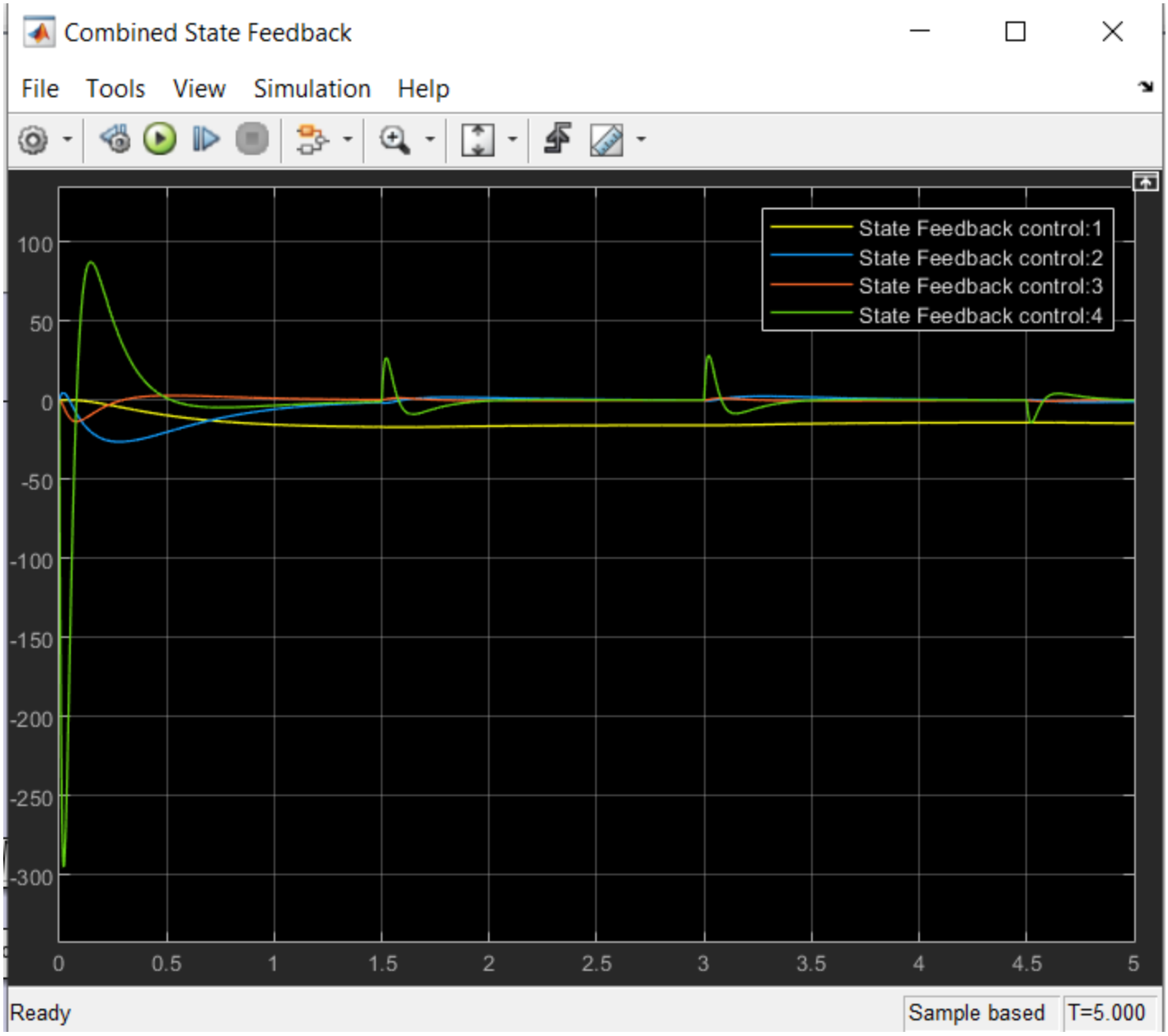


Figure 14: Corrective behaviour in simulink to random distrubances

As we can see, the system rapidly recovers back to steady state. Figure 15 below shows the displacement and angular displacement of the system with the random disturbance applied.

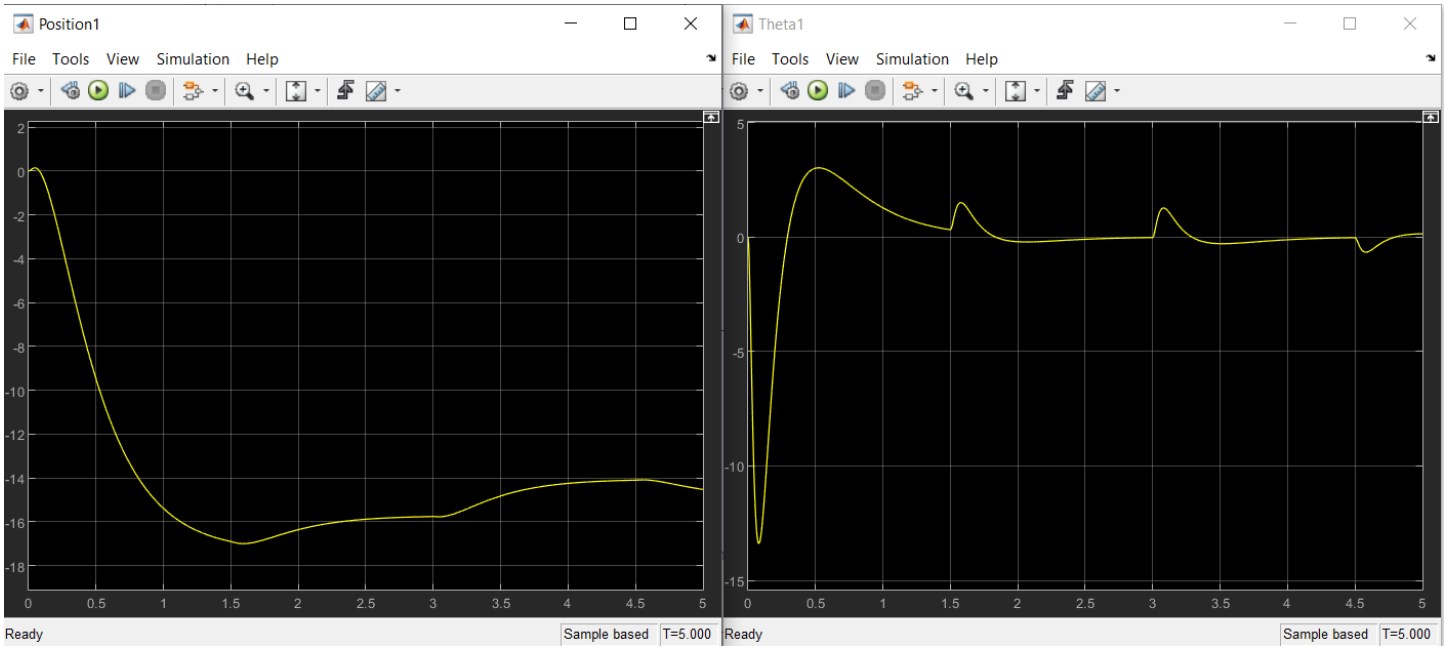


Figure 15: Position and angle corrective behaviour

As we observed earlier, the settling time of the system was approximately 1.5 seconds. We can see that here, with the theta value returning to nearly 0 within 1.5 seconds, before being disturbed again as illustrated more clearly in Figure 15.

Next, we wish to examine the controller performance on the non-linear system. For this, the controller is connected to the non-linear system built in simulink. Similar to the linearized system, a random disturbance is applied.

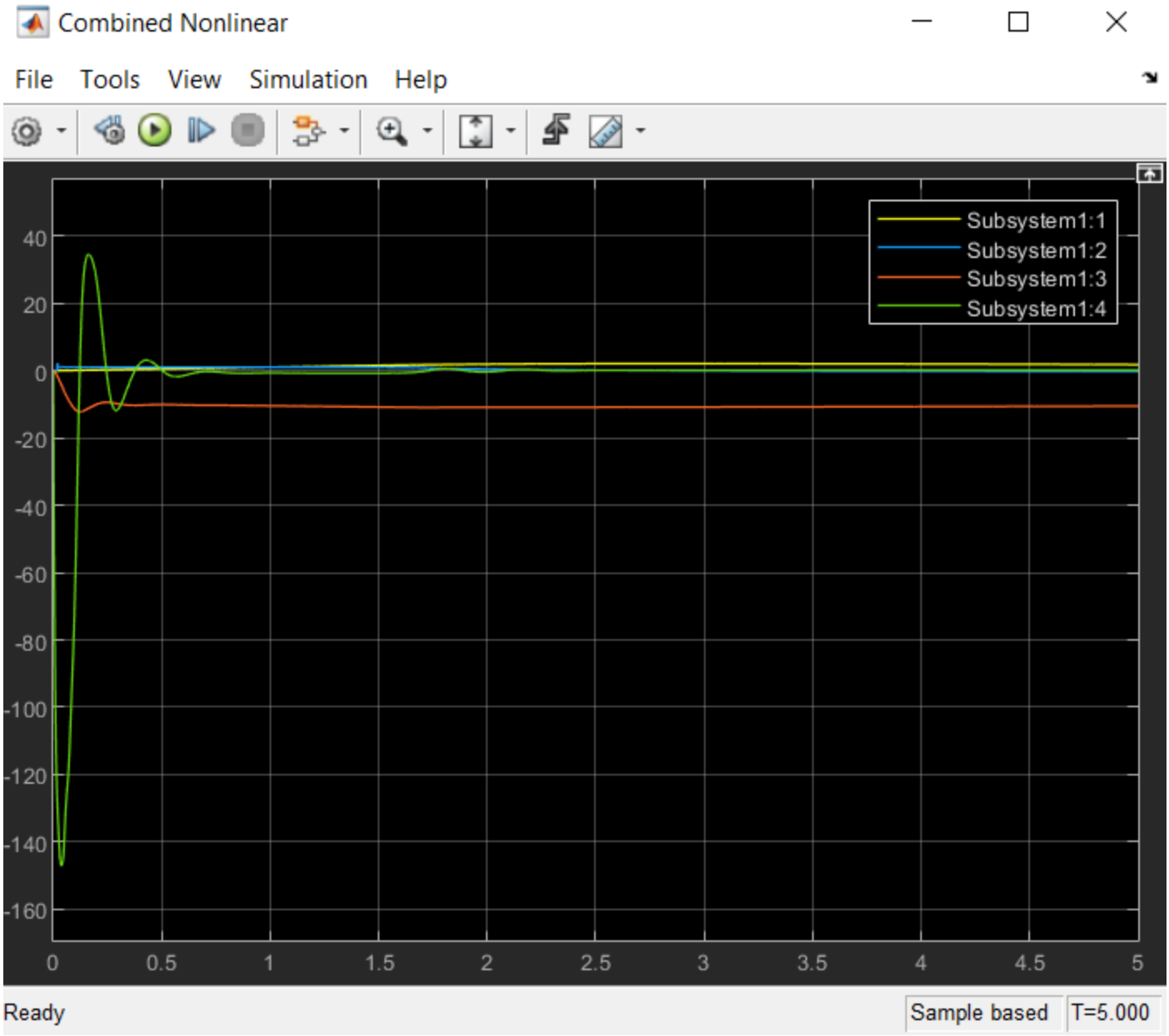


Figure 16: Displacement, velocity, angular displacement, and angular velocity non-linear response given controller feedback design

As we can see in figure 16, the response is very similar to the linearized response, with one major difference. We can see that the value of theta reaches a steady state, but the steady state value is not zero. Rather, it settles at around 15 degrees. This is a shortcoming of the controller. Since it was designed to be used at small values of theta, whenever a larger value is reached for that state, we begin to see error in the state feedback controller.

Through trial and error, a better controller could potentially be designed. We may consider using a Linear Quadratic Regulator to design the controller. By assigning higher weight to the theta state, we should in theory get a controller that is optimized to control theta to a desired value. In practice, however, we found that LQR did not yield an improvement over our own controller. With state wights set as
Q = [

```
     1     0     0     0
     0     1     0     0
     0     0     3     0
     0     0     0     1
```

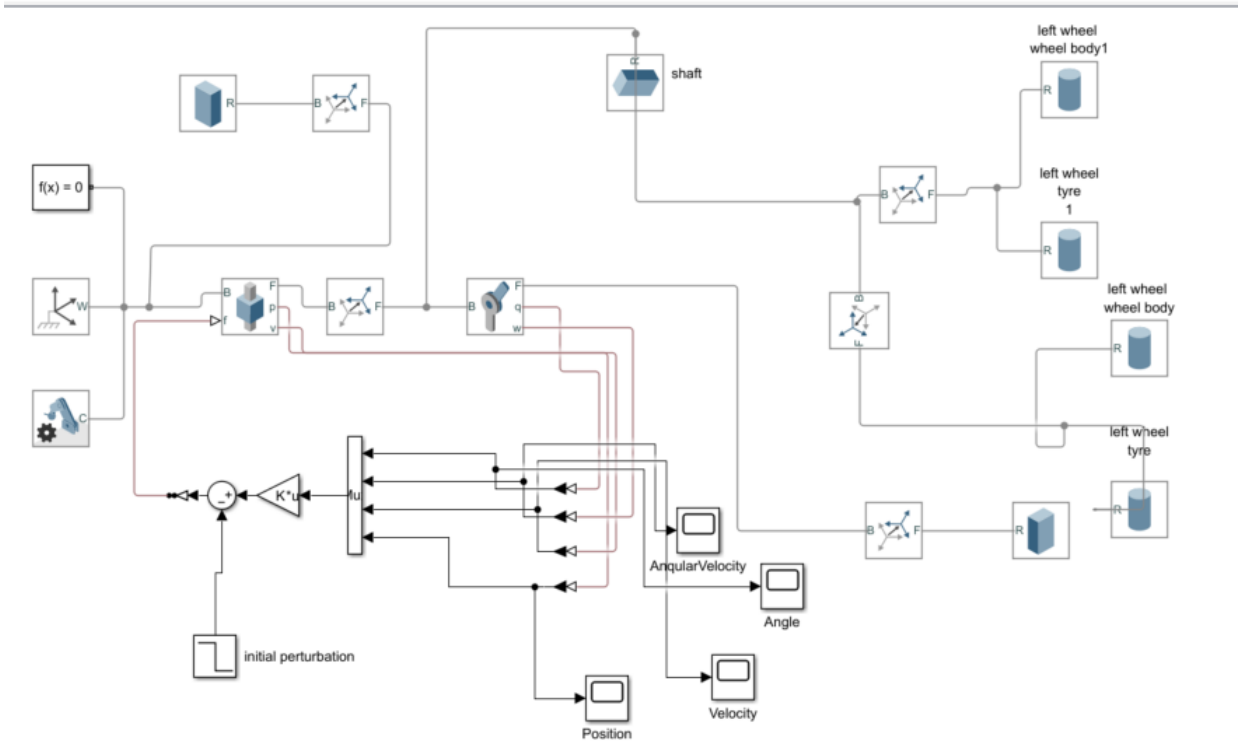
]
And the input weight set to R = 2, we obtain poles at

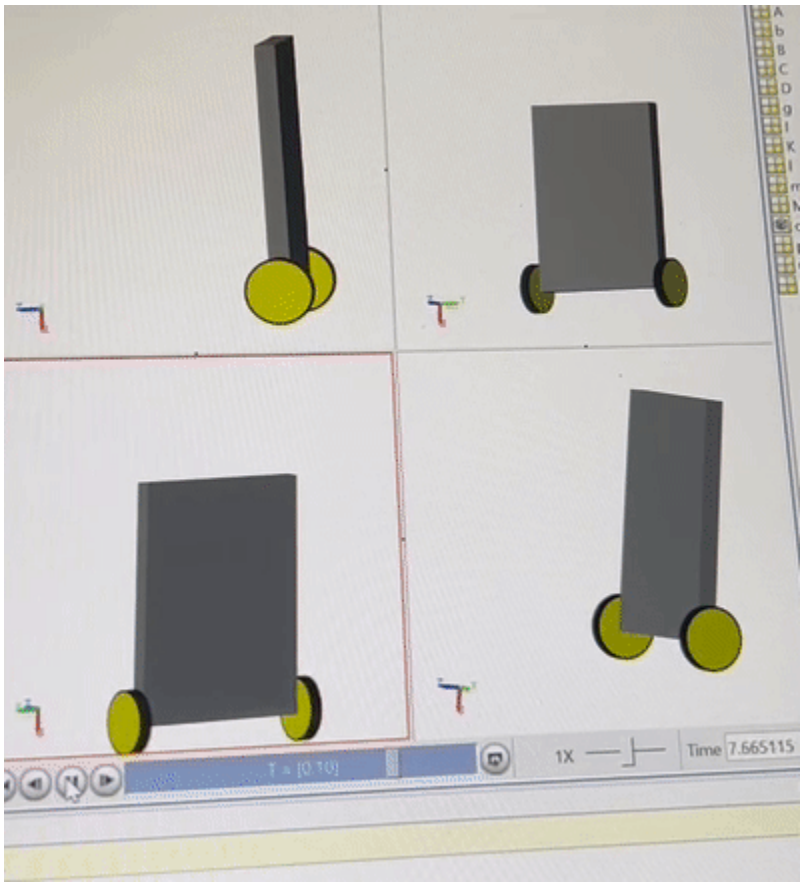
```
  1.0e+04 *

-0.0001 + 0.0000i
-0.0003 + 0.0002i
-0.0003 - 0.0002i
-2.0751 + 0.0000i
```

With controller gain values K = [-0.7071, -0.9971, -4.4737, -0.7250]. This controller leads to inferior performance, with a settling time of 4.3 seconds. It also causes the non-linear system to crash. Due to its very poor performance when compared to the controller designed using Ackermann's formula, we do not consider the LQR based controller as acceptable option for this system.

Furthermore, simulations were made to illustrate the feedback control of the robot correcting its angle upright. The simulations can be found in separate files within this submission. Figures 16-2 and 16-3 illustrates the Simscape multibody model, whereas 16-4 illustrates the Simscape response of the system:





Figures 16-2 and 16-3: Simscape multibody design and simulation

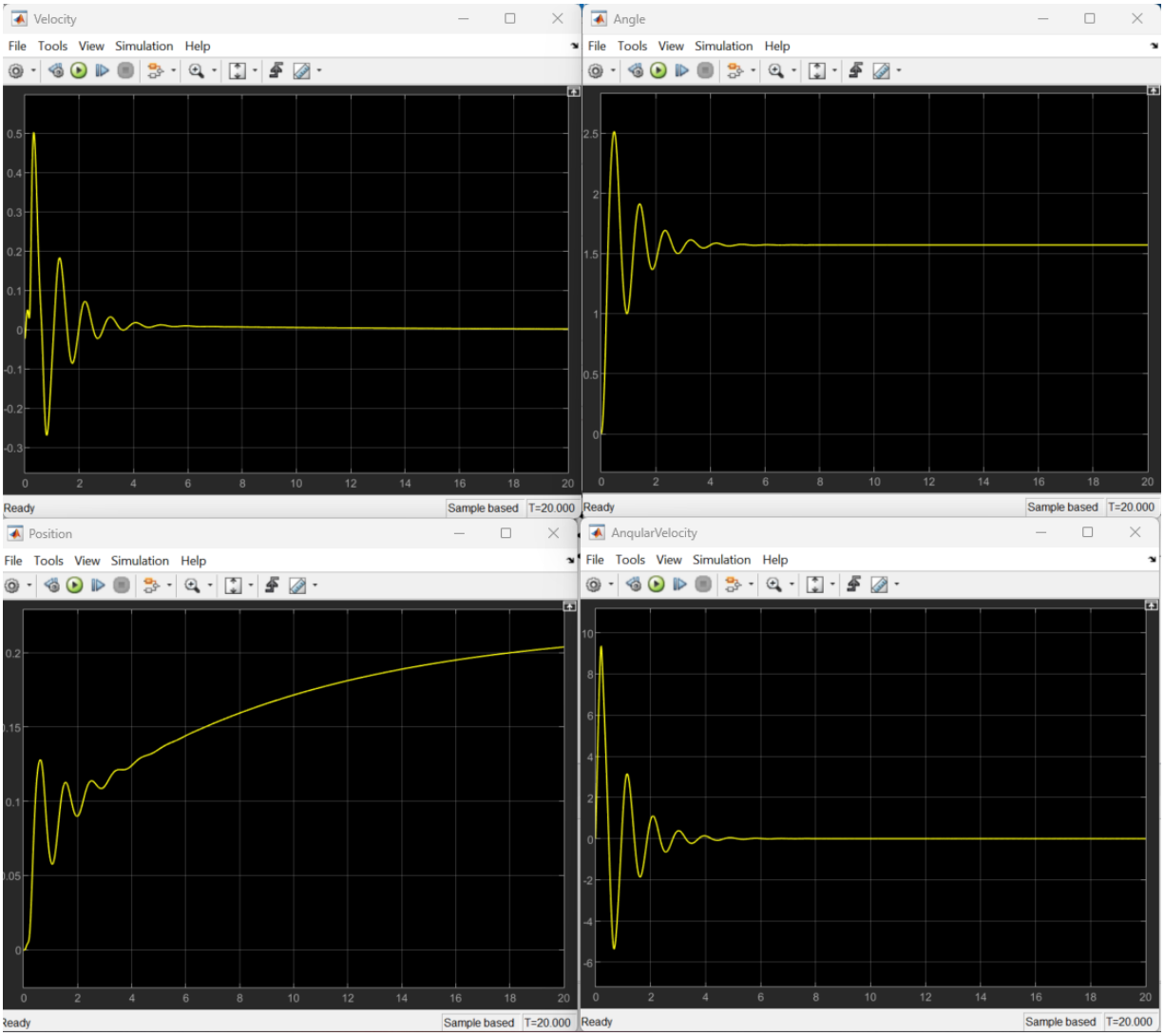


Figure 16-4: Simscape multibody simulation system response

Upon inspection of the simulation, the model experiences a disturbance, and oscilates back and forth in order to correct itself - as desired by our specifications and requirement within a reasonable amount of time. The simulation represents the ideal feedback control design of investigation.

Observer Design and Observer-based Control

One of the potential design constraints of such a system in real life is the inability to reliably measure all 4 states at any given time. However, for a state feedback controller to be implemented, all states of the system need to me observed. To do this, we design a state estimator or observer. The state estimator will predict all states of the system at all times, without the need to directly measure them. In this case, the Luenberger observer will be used. This is because it is designed to be robust and

provides good estimates for internal states, even in the presence of noise and disturbances. The Luenberger observer dynamics will be given by:

$$\hat{x}' = A\hat{x} + Bu + L(y - \hat{y})$$
$$\hat{y} = C\hat{x} + Du$$

Where $\hat{x}$ is the state estimate. L is the observer gain matrix. It is found in a similar manner to the K controller matrix. The first step in designing the observer is to pick the locations for the observer poles. There are no definitive mathematical processes to follow when designating observer poles, but in general the larger the poles are the faster the observer will converge to a good estimate of the system states. For our system, we have chosen the pole locations as [-15, -30, -140, -250]. This follows the general guideline of placing observer poles at 5x the controller poles. The poles at -15 and -30 are the dominant poles of the system. The resulting L matrix is found to be

```
L = [
  1.0e+03 *

   0.0010    0.0010         0         0
        0    0.0100    0.0000         0
        0         0    0.1000    0.0010
        0         0    0.7229    1.0000
                                       ]
```

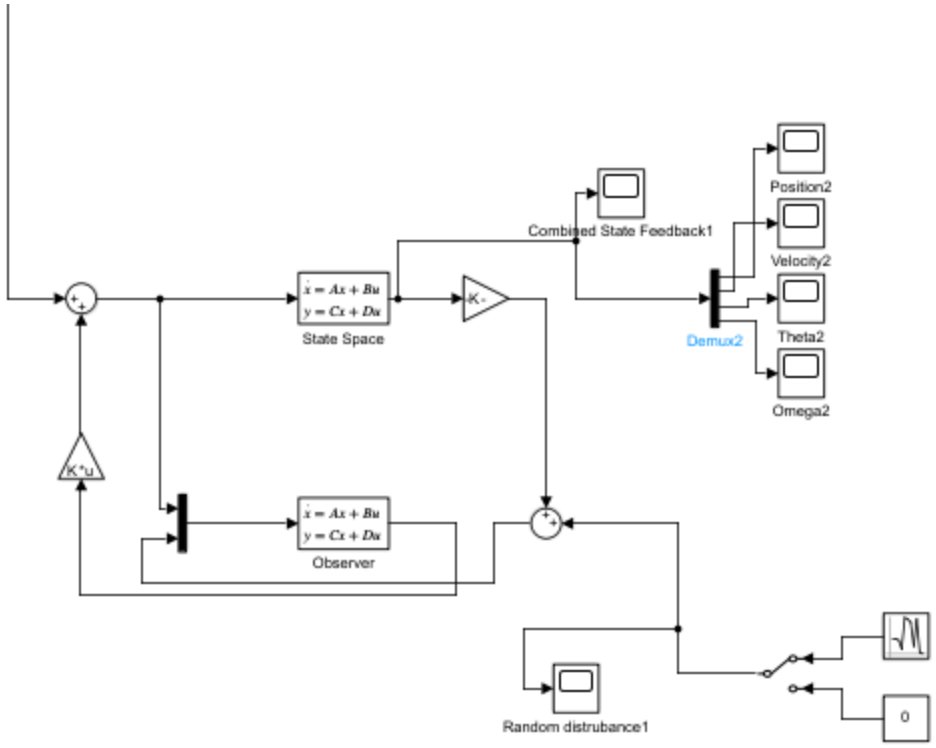


Figure 17: Simulink model of state observer controller for linearized system

With the observer implemented to estimate the states of the system, we can then use the observed states as input to the controller. If the observer design is good, we expect the system to behave similarly as it did with direct state feedback, with some error expected. Figure 17 shows exactly the expected behaviour. The directly measured states of displacement and angular displacement settle quickly and without any major oscillations. The estimated acceleration and angular acceleration states oscillate slightly before also settling in a little bit more time than the direct state feedback. The settling time of the system increases to 3.9 seconds. Althout this is more than double the previous result, it is not fully reflective of the system response. We can see from the response charts that all states settle to about 95% of steady state within less than 2 seconds, which is still good performance. Further improvements to the observer design (optimization of pole locations) could lead to a lower estimate error, and therefore better control.

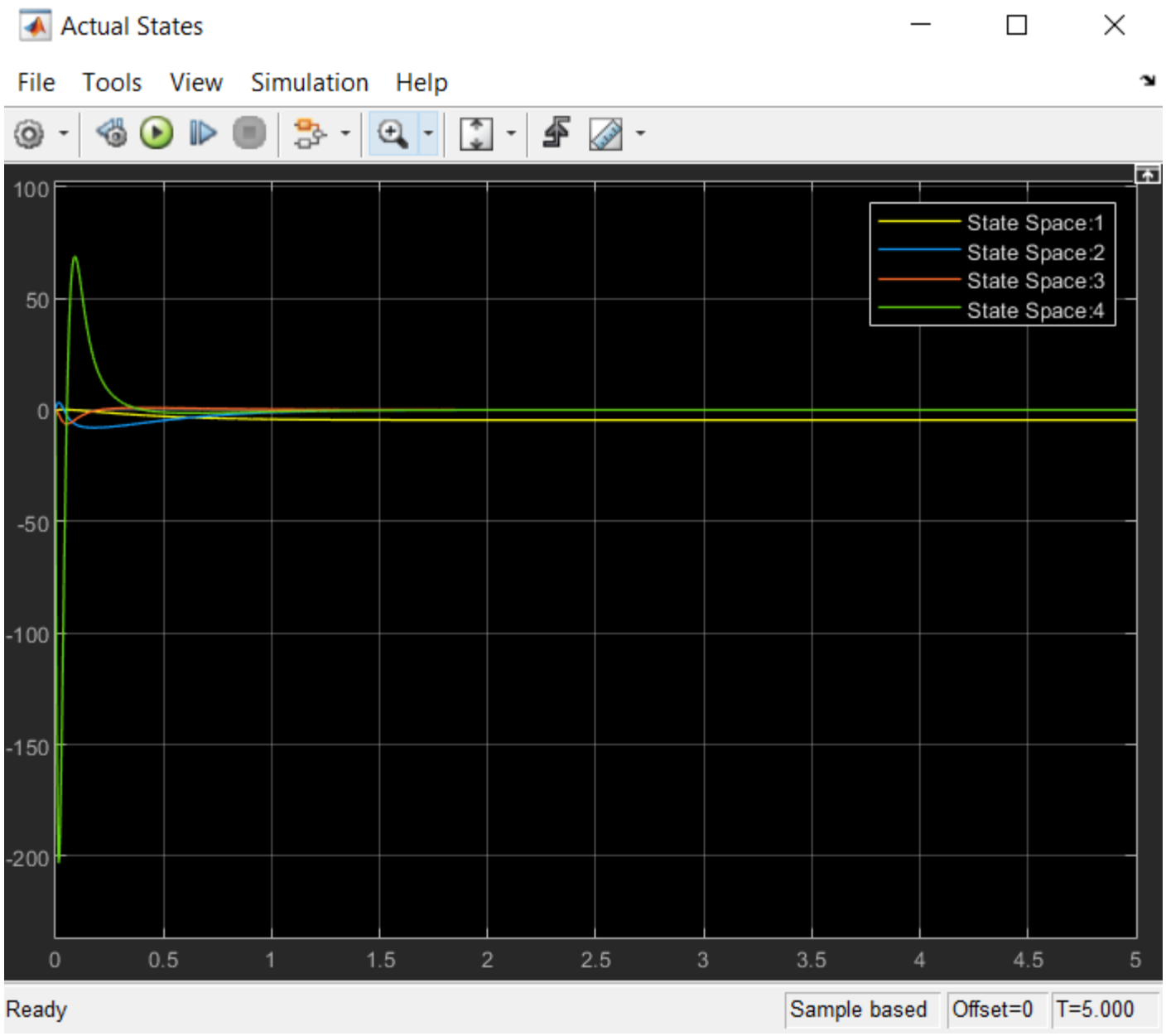


Figure 18: State feedback observer of linearized system

As with the state feedback controller, we also simulated continuous random disturbances to determine whether the slower settling time would affect the system. We chose 2 seconds for the signal period, which is theoretically faster than the settling time of the system. If the system is properly controlled, we expect it to not become unstable despite the disturbances coming before the system settles to steady state. Figure 19 below shows the waveform of the random disturbance signal applied to the system. Figure 20 shows the combined state responses. Figure 21 shows the displacement and angular displacement responses of the system.

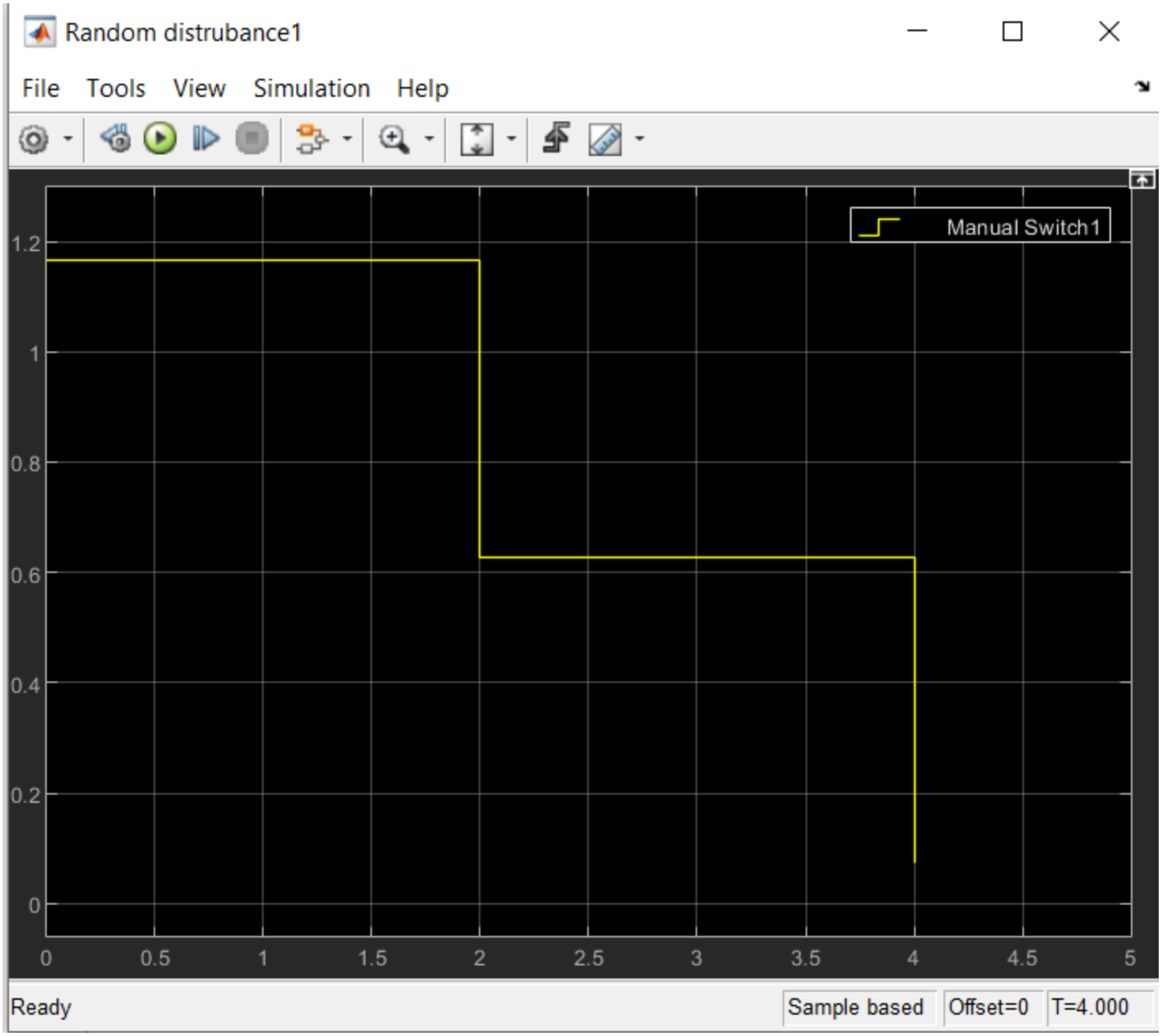


Figure 19: Sample disturbance

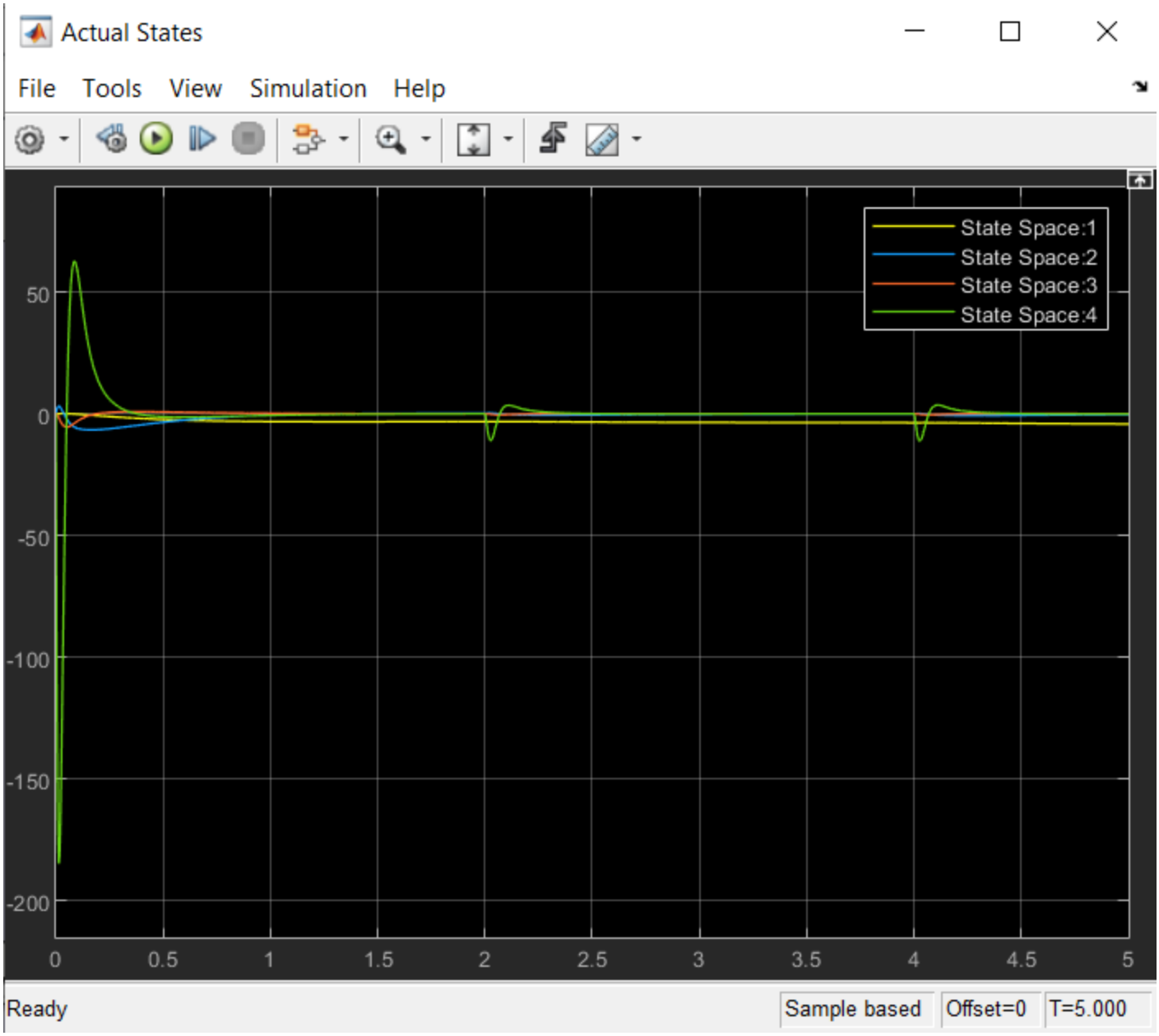


Figure 20: State feedback given disturbances of linearized system

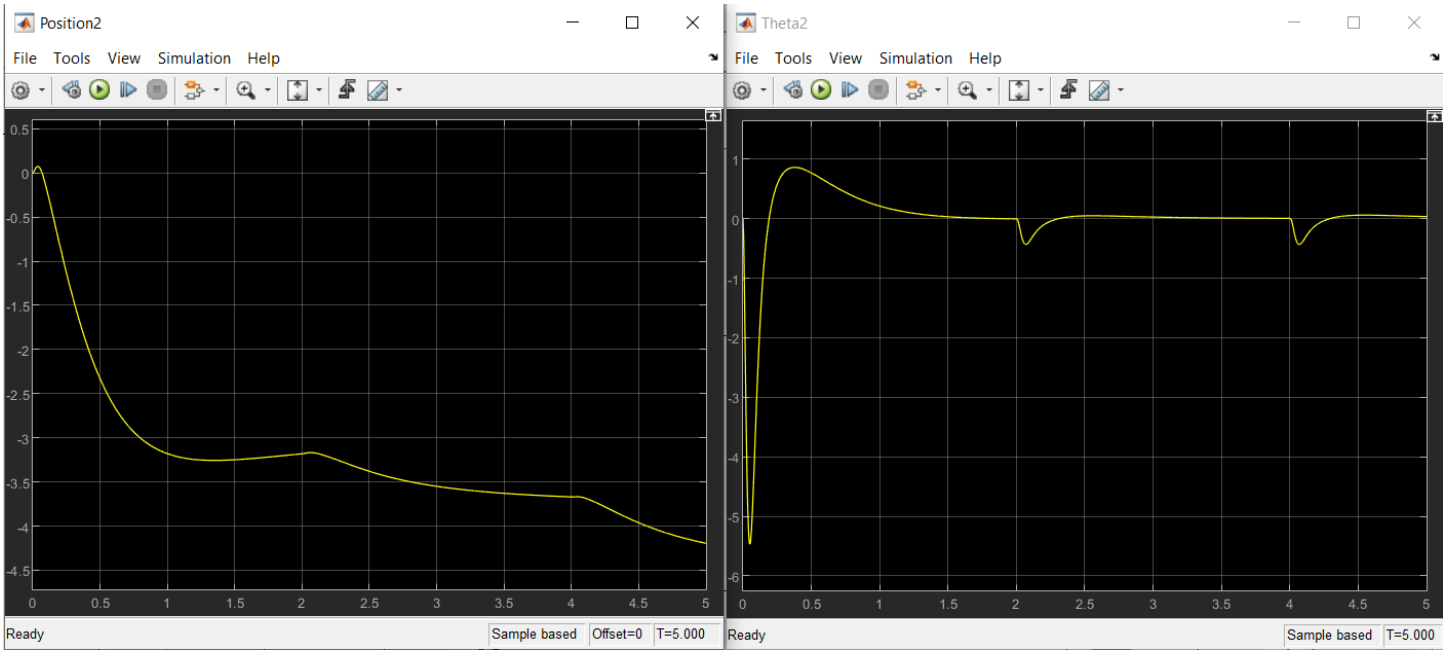


Figure 21: State feedback observer response of disturbances within linearized system

Lastly, we wish to examine the performance of the observer based feedback controller on the non-linear system. This is an important test because it will determine whether this controller is suitable for use in real world applications. This assumes that the system is in its true, non-linearized form, and only some of the states are able to be measured. If the controller can reliably steer the value of theta back to the desired 0 degrees, we can assume the controller is functioning well.

First, we test it with no disturbance. The response is demonstrated in Figure 22 below.

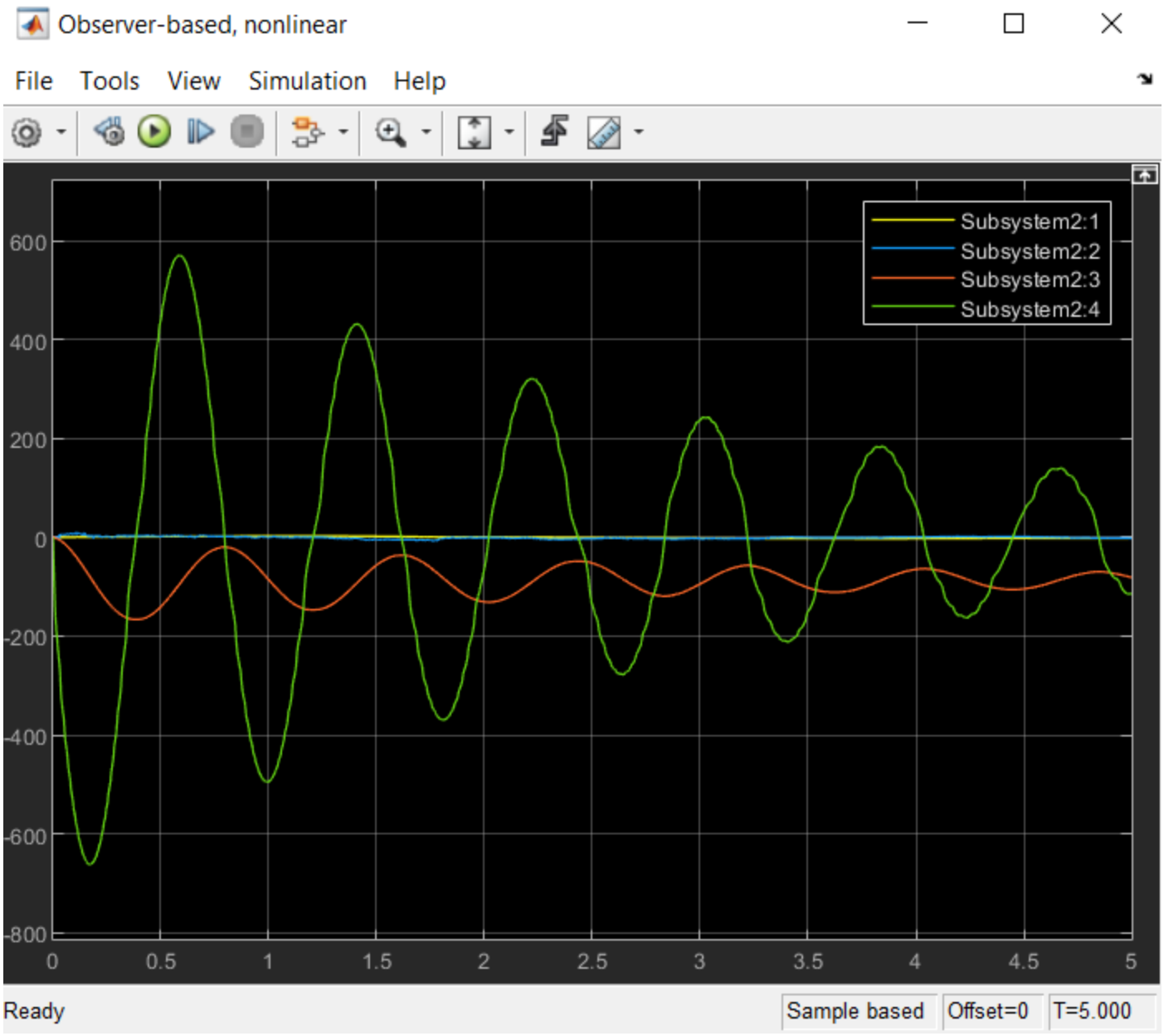


Figure 22: State feedback observer response of non-linearized system

As we can see, the value of theta is oscillating but is being steered toward zero. Same as with the state feedback controller, the stead state value seems to be slightly off zero. However it is important to note that the system is being stabilized. Next, we examine the system response to random disturbances. As we can see in the Figure 23 below, the response looks similar, but now the value of displacement is changing faster, which can be explained by the robot needing to move around to readjust its angle of tilt.

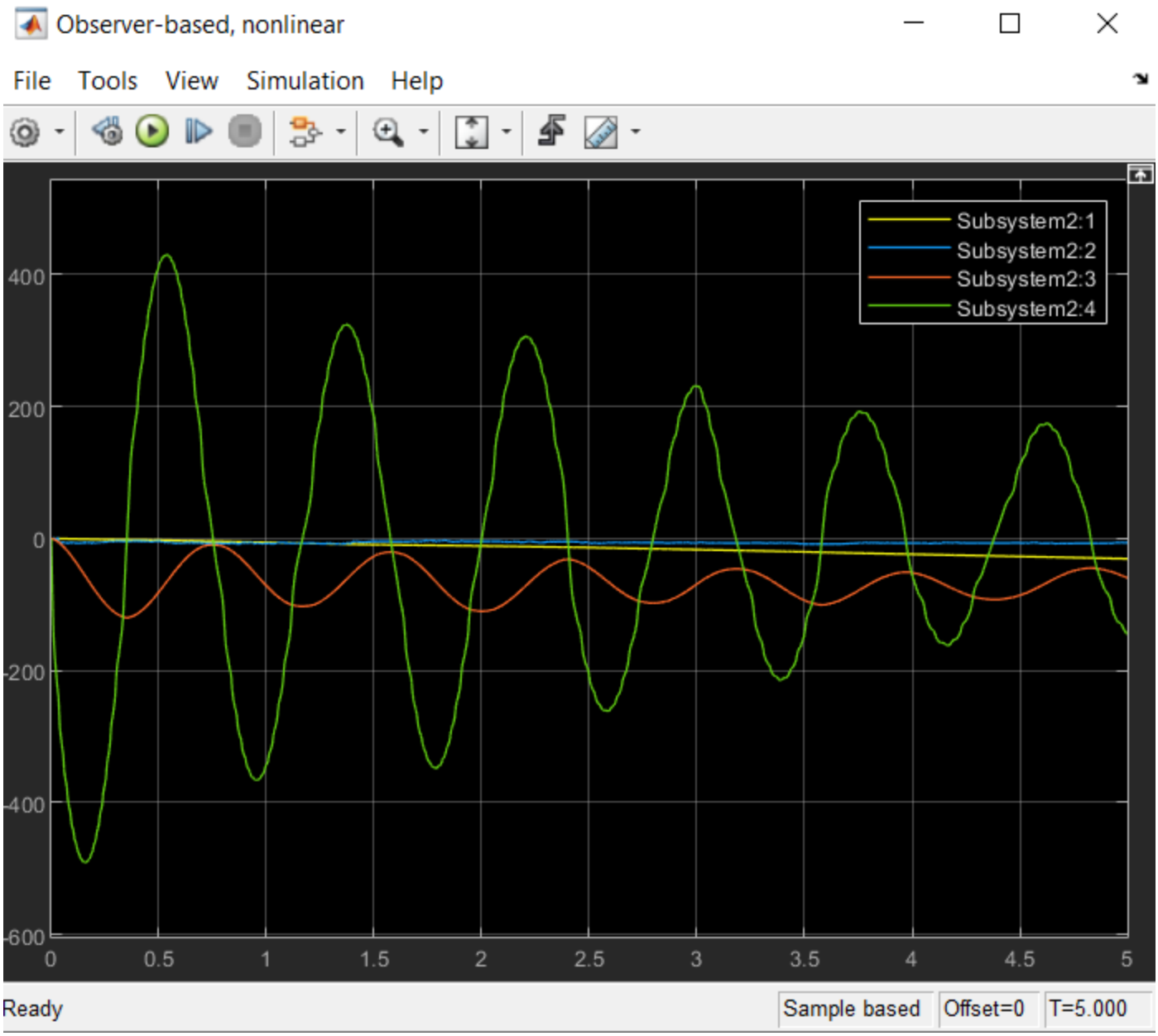


Figure 23: State feedback observer response of disturbances within non-linear system

This controller does stabilize the non-linear system based on state estimates, however the system is underdamped and should be further refined.

Conclusion

This report detailed the process of designing a control system for a Two Wheeled Self Balancing Robot. A detailed explanation was given into the formulation of the dynamic equations of the system. The operating point was determined and the system was linearized around the operating point. The open-loop characteristics of the system were examined. It was determined that a controller to modulate the input torque was necessary to ensure desired performance. The deisred performance was assessed, and based on this a controller was designed by using Ackermann's formula to place closed-loop poles at desired locations. Then, a full state observer was designed to estimate all states of the system. This allows for full state feedback control even if all states of the system cannot be reliably measured. The linear and non-linear system were modelled in MATLAB and Simulink, and the various controller and observer combinations were simulated to assess system performance. It was found that with the controller and observer, the linearized system achieved steady state in under 2 seconds. There was some steady state error found in the non-linear system, as well as underdamping of the non-linear system. Potential changes to the controller were suggested to fix these.

Sources

[1] Fk, "Modeling and control of two-wheels self balancing (TWSB) robot (updated 8 Feb 2023)," *Modeling and Control of Two-Wheels Self Balancing (TWSB) Robot (Updated 8 Feb 2023)*, 01-Jan-1970. [Online]. Available: https://fkeng.blogspot.com/2019/03/theory-and-design-of-two-wheels-self.html.

[2] Park J-H, Cho B-K. Development of a self-balancing robot with a control moment gyroscope. International Journal of Advanced Robotic Systems. 2018;15(2). doi:10.1177/1729881418770865

[3] "Inverted pendulum," *Wikipedia*, 26-Jan-2023. [Online]. Available: https://en.wikipedia.org/wiki/Inverted_pendulum. [Accessed: 14-Apr-2023].

[4]"Classic inverted pendulum - equations of motion," *YouTube*, 18-Mar-2015. [Online]. Available: https://www.youtube.com/watch?v=5qJY-ZaKSic&ab_channel=BriannoColler. [Accessed: 14-Apr-2023].

[5]"Classic inverted pendulum - equations of motion," *YouTube*, 18-Mar-2015. [Online]. Available: https://www.youtube.com/watch?v=5qJY-ZaKSic&ab_channel=BriannoColler. [Accessed: 14-Apr-2023].

[6]“Implementation of a two wheel self-balanced robot using ... - IEEE xplore.” [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8883007. [Accessed: 15-Apr-2023].

[7] L. Pupek and R. Dubay, “Velocity and position trajectory tracking through sliding mode control of two-wheeled self-balancing mobile robot,” *2018 Annual IEEE International Systems Conference (SysCon)*, 2018.

[8] “Home Page,” *UNItesi*. [Online]. Available: https://thesis.unipd.it/retrieve/5b660ab3-8204-4d2f-8700-002b2aa04d7a/Spiller_Dino_1084324.pdf. [Accessed: 14-Apr-2023].