Middle East Technical University

Department of Electrical & Electronics Engineering

EE 498 Special Topics: Control System Design and Simulation

Term Project Report

Mehmet KILIÇ

2232262

Table of Contents

Introduction

In this project, an inverted pendulum on a cart system is modeled. After transforming the nonlinear system to a linearized discrete time state space model, plant properties and possible control strategies are discussed. Then, a controller is implemented and required simulations are conducted. Finally, results are discussed.

After modeling and controller design part is completed, a simulation task is completed. Given satellite trajectory model is simulated using built-in solvers in Simulink. Then, a Runge-Kutta method with order 4 is implemented with fixed step size. Finally, this method is adapted into a variable step size one and performances of these methods are compared.

Part 1 - Control

1.a. System Model

In this section, mathematical model of the inverted pendulum on a cart system is derived. A simplified version of the system is given below. In this system, a pendulum having the mass m is placed on a cart and its position (i.e., the angle Θ) is controlled by using an actuator which drives the cart. The system is unstable without control. This means that the pendulum will fall if not controlled. When the position of the pendulum starts to deviate from the reference (let us say the point θ=π), actuator moves the cart so that the pendulum keeps its position. In the figure below, the force F shows the resulting force that actuator applies to the cart. In this derivation, actuator side is omitted, and system dynamic equations are derived in terms of the force F.
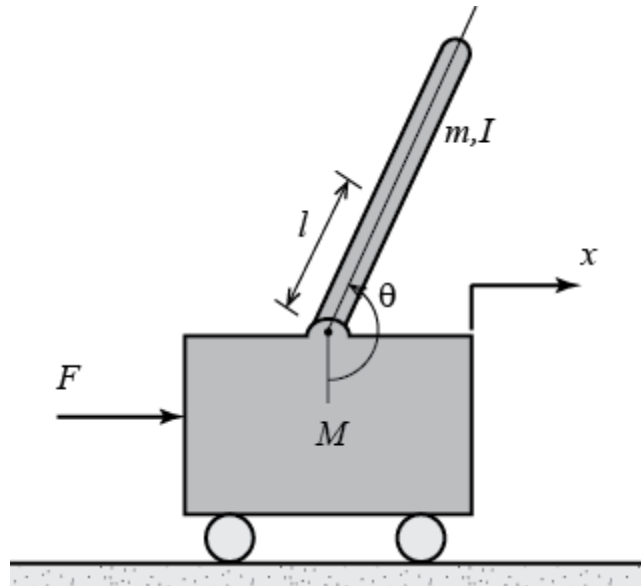

Figure 1. Inverted Pendulum on a Cart

M: Mass of the cart

m: Mass of the pendulum

b: Friction coefficient

l: Distance between the cart and the pendulum center of mass

I: Moment of inertia of the pendulum

F: Force applied to the cart by actuator

x: Cart's position

Θ: Counterclockwise angle between pendulum and the vertical axis.

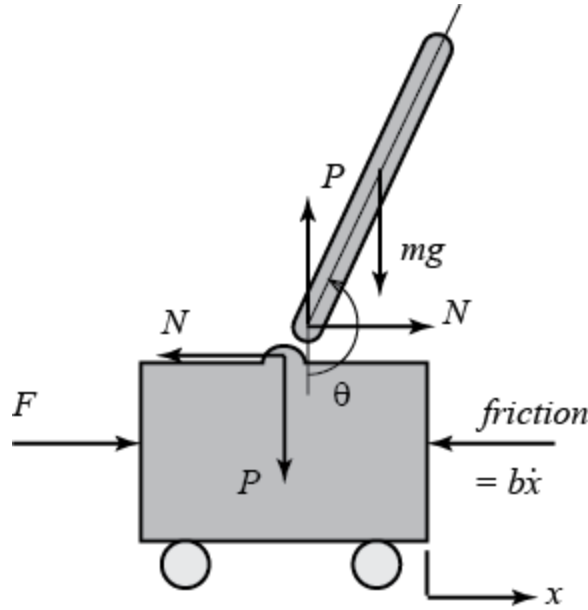Now, let us draw the free body diagram and write the force equations of the system.

Figure 2. Free body diagram of the system

Where P and N shows the action-reaction forces that the pendulum and cart apply to each other. First, the force equation applied to the cart in horizontal direction is written.

$$M\ddot{x} + b\dot{x} + N = F \tag{1}$$

Second, forces acting on the pendulum in the horizontal direction are summed up.

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{2}$$

Finally, N in the equation (2) is put into the equation (1) and following equation is obtained.

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \tag{3}$$

After obtaining this equation, forces acting on the pendulum in the perpendicular direction of the pendulum are summed up.

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \tag{4}$$

Also, moment equation around the center of the pendulum is written.

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \tag{5}$$

When the equations (4) and (5) are combined, following equation is obtained.

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \tag{6}$$

5

Now we have the equations (3) and (6), which are required to obtain the continuous time state space model of the system. Note that both equations include sine and cosine terms, so they are nonlinear. To analyze the system and design a controller for the system, the equations (3) and (6) should be linearized. To do that, the angle that we want to keep the pendulum must be determined. Let us define that the reference angle $\Theta = \pi$. This means that we want to control the pendulum angle such that it does not deviate from this point much. So, we can linearize the trigonometric functions around this point. If we define the deviation angle ϕ, we can approximate the pendulum angle Θ as $\Theta = \pi \pm \phi$.

$$\cos\theta = \cos(\pi + \phi) \approx -1 \tag{7}$$

$$\sin\theta = \sin(\pi + \phi) \approx -\phi \tag{8}$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0 \tag{9}$$

By substituting the equations (7), (8) and (9), final linearized system dynamics equations are obtained. Note that the letter u is used to represent the force instead of the F because it represents the input.

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \tag{10}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \tag{11}$$

Finally, we are ready to obtain the state-space model. To completely control the system, state vector is chosen as follows.

$$x(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix}$$

By using this state vector, both carts and pendulums position can be controlled. Otherwise, we could control the pendulum angle with the cart's position diverging to the infinity or vice versa. However, we can control both carts and pendulum's position now. Also, the output vector y is determined as follows.

$$y(t) = \begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix}$$

6

Finally, continuous time state space model of the system is obtained by using equations (10) and (11).

$$
\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u
$$

$$
\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u
$$

Figure 3. Continuous Time State Space Model

After having the continuous time state-space model, discrete time model is obtained by using MATLAB. Before continuing to the controller design, plant properties are observed. First, stability of the plant is checked by calculating the eigenvalues of matrix A in the discrete time state-space model. As shown in the figure 4, there are two eigenvalues that outside of the unit circle, which makes the system asymptotically instable. This makes sense because if no control is implemented, pendulum would fall. This means that if the system is started without control, output will diverge. In other words, zero input solution does not converge to zero for large times.

1	1.0000 + 0.0000i
2	0.9989 + 0.0000i
3	1.0005 + 0.0012i
4	1.0005 - 0.0012i

Figure 4. Eigenvalues of matrix A

Then, ranks of the controllability and observability matrices are calculated. Both have the rank=4, Which means that the plant is completely controllable and observable. Transfer functions are also obtained. Since two outputs are defined in the system definitions, two transfer functions can be obtained for both outputs. These transfer functions are obtained in MATLAB.

```
tf1 =

            2.105 s^2 + 17.4
    ---------------------------------
    s^4 + 0.2105 s^3 + 0.5263 s^2 + 1.74 s

Continuous-time transfer function.
```

Figure 5. Transfer Function with respect to output x(t)

```
tf2 =

            5.263 s^2
    ---------------------------------
    s^4 + 0.2105 s^3 + 0.5263 s^2 + 1.74 s

Continuous-time transfer function.
```

Figure 6. Transfer function with respect to output theta(t)

Stability of both transfer functions are checked by using the *isstable()* function of the MATLAB and it turns out that both transfer functions are instable as expected. Having full information about the system model, now we can discuss the suitable controller design methods.

1.b. Controller Design Method Discussion

By the end of the last section, DTSS model of the system is obtained. System is asymptotically instable, controllable, and observable. Also, has no plant integrators for both transfer functions. Although the complexity of the system and having multiple state variables call for the methods like LQR and MPC, suitability of other methods is also discussed.

This system is too complicated to use a simple design method like bode plot design. Another possible method is the symmetric optimum. Having a stable plant is a prerequisite for symmetric optimum method. It is obvious that this method is not suitable for this system. Another possible method is the pole placement. Plant structure is suitable for using pole placement method. Therefore, it is suitable for such a system. However, since we have multiple state variables that we want to control, the most suitable design methods are the LQR and MPC.

No input or state constraints are defined in the system definition. Therefore, LQR design seems safe.

1.c. LQR Design

In this part, first, a finite horizon LQR algorithm is developed with different cost matrices. Their differences are compared, and relevant matrices are determined by considering design specifications and control aim. Then, an infinite horizon solution is implemented with the same cost matrices and results are compared. After solving the problem in MATLAB using linearized system model and ignoring the non-idealities such as saturation and disturbances, both linear and non-linear models are implemented in Simulink and different cases are simulated. Effect of the non-idealities, accuracy of the linearization and success of the designed LQR is discussed.

Before starting to the finite horizon LQR design, first the priorities of the control design are considered. Note that different cost matrices lead us to different optimal solutions. Therefore, weights of the input, intermediate states and final states should be adjusted such that the controller would behave suitably to the control aim. Note that LQR may become vulnerable in case of saturation limits on the input or states. Let us assume that our main consideration is this: magnitude of the input signal is more important from the settling time or the overshoot. This verbal definition is a good point to start. We may start choosing cost matrices as shown below and tune them to enhance the performance of the controller.

$$Q_f = Q = C' * C$$

$$R = 0.1$$

Starting from this values, results of finite horizon LQR solutions with different cost matrices are given below. In all simulations, large step number N values are preferred to obtain the complete behavior of the system until reaching the steady state. Initial state vector is provided such that the cart is ordered to go from x=1 to x=0 point while all the other states are 0 initially.
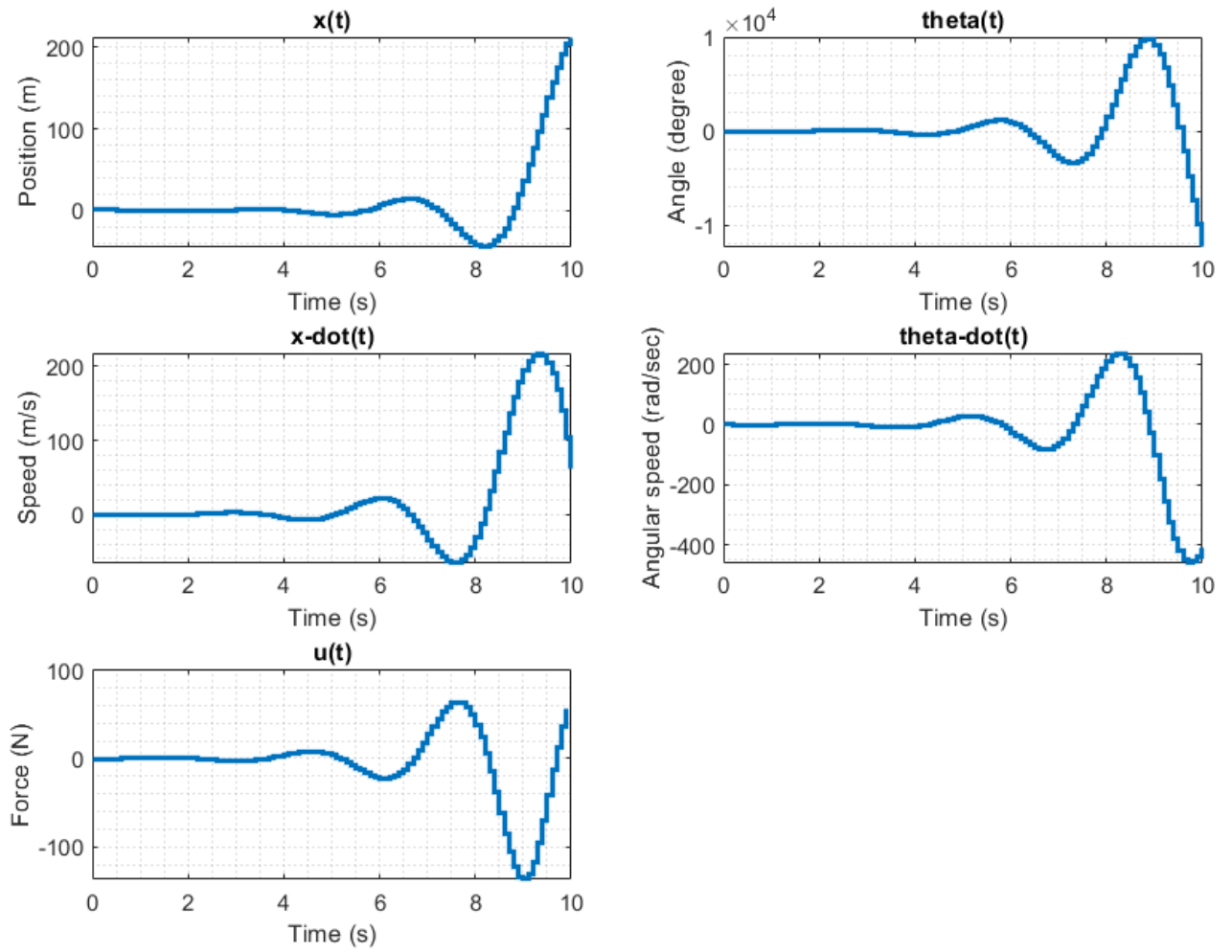
Figure 7: LQR result for R=0.1 and Qf=Q=C'C

Note that the overall system is instable with these cost matrices. This regulator is not able to control the system. At this point, I have tried to increase the magnitude of the elements of the Q matrix.
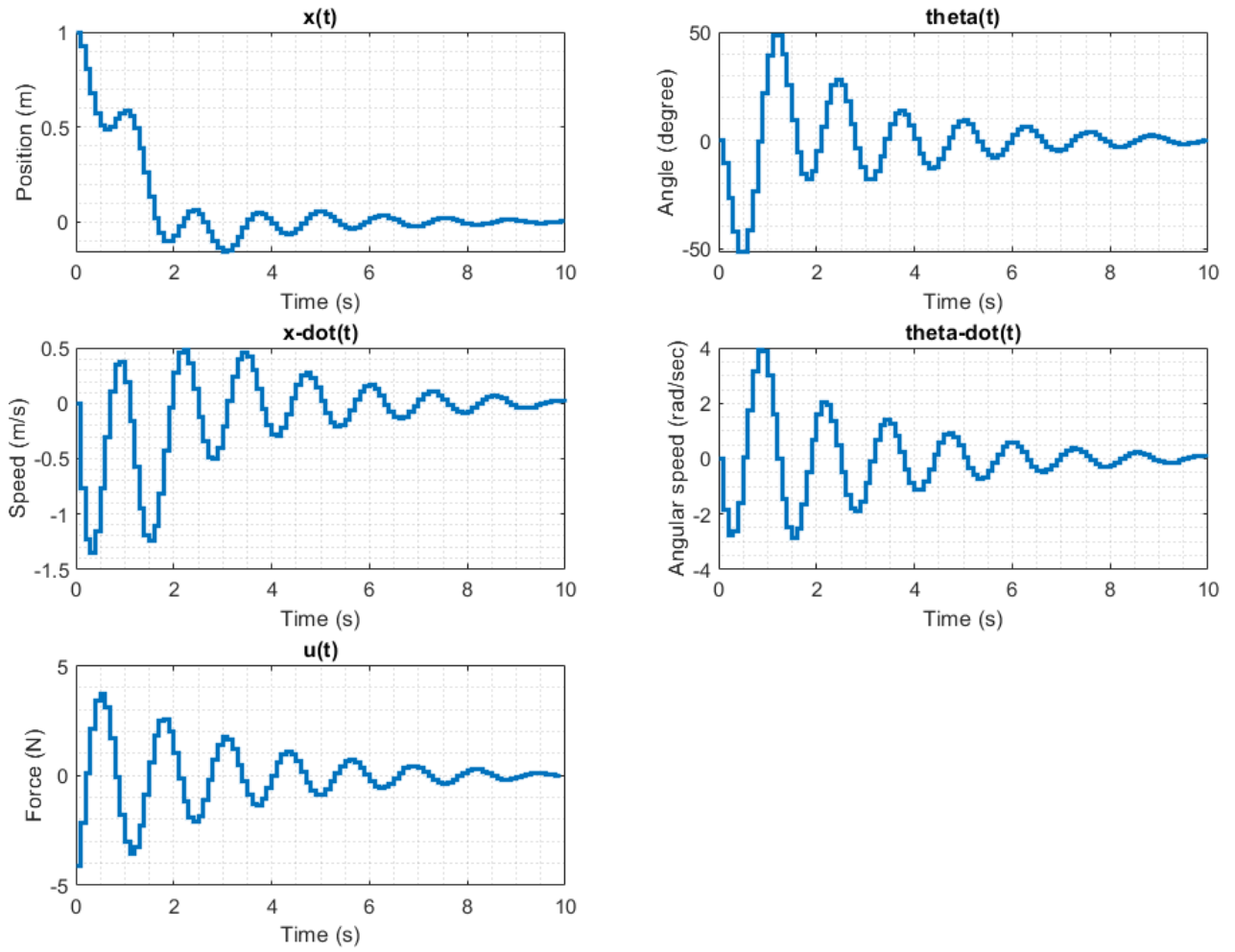
$$Q_f = Q = 10 * C' * C$$

$$R = 0.1$$

*Figure 8: LQR result for R=0.1 and Qf=Q=10*C'C*

With the increased weight of the intermediate and final states, now we can control the system with some oscillations. Note that the input weight is still high, and the magnitude of the input signal is bounded within the -4 < u(t) < 4 region. At this point, I keep trying different cost matrix configurations to see if I can keep input signal low while enhancing the settling time and the overshoot performances.
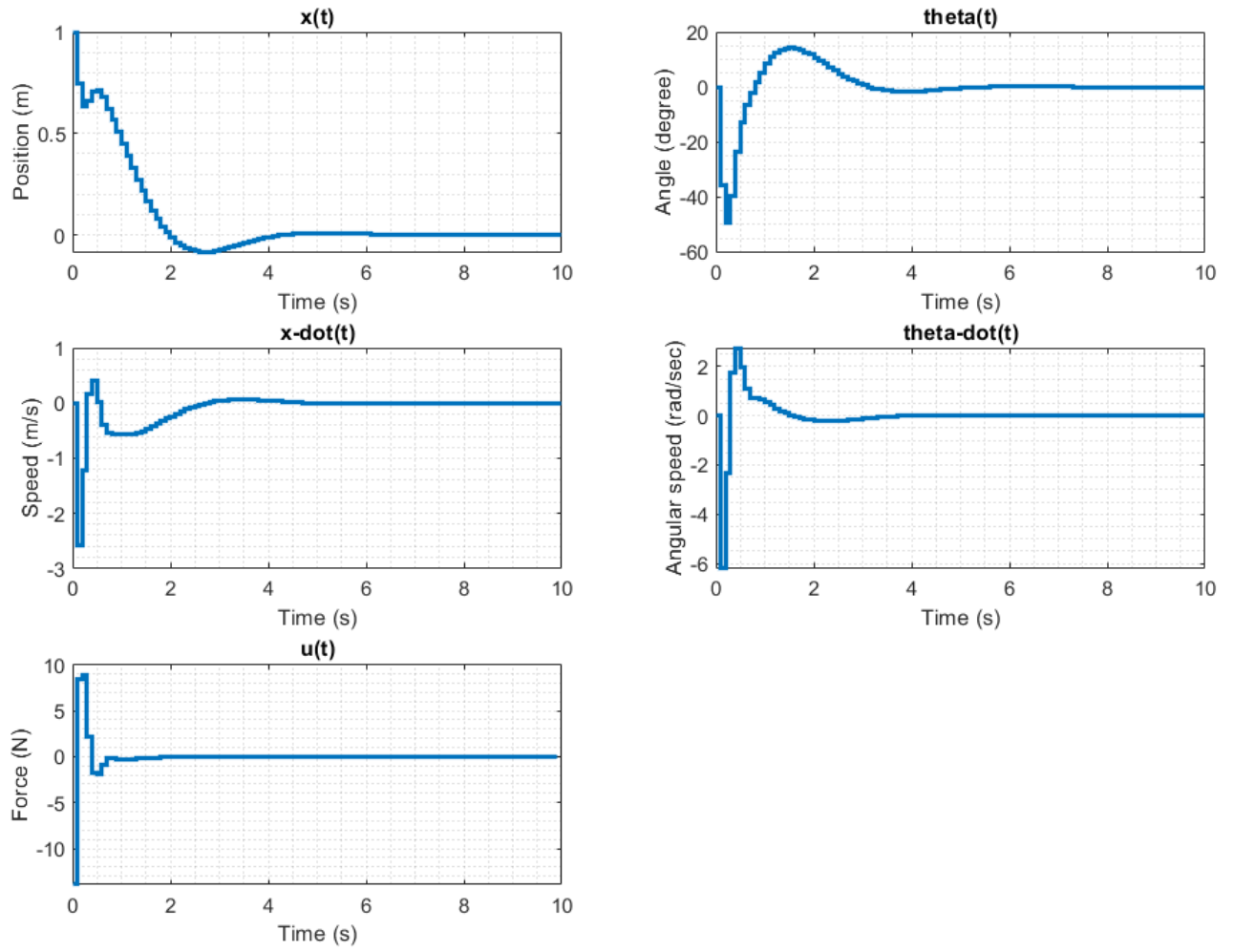
*Figure 9: LQR result for R=0.01 and Qf=Q=10*C'C*

With smaller input weight, now the system stabilizes faster with smaller overshoot because controller gives more importance to the transient and final states. However, maximum value of the input signal becomes larger with this configuration. In case of an input saturation, this result can bring problems. To see the full effect, I try a more extreme example.
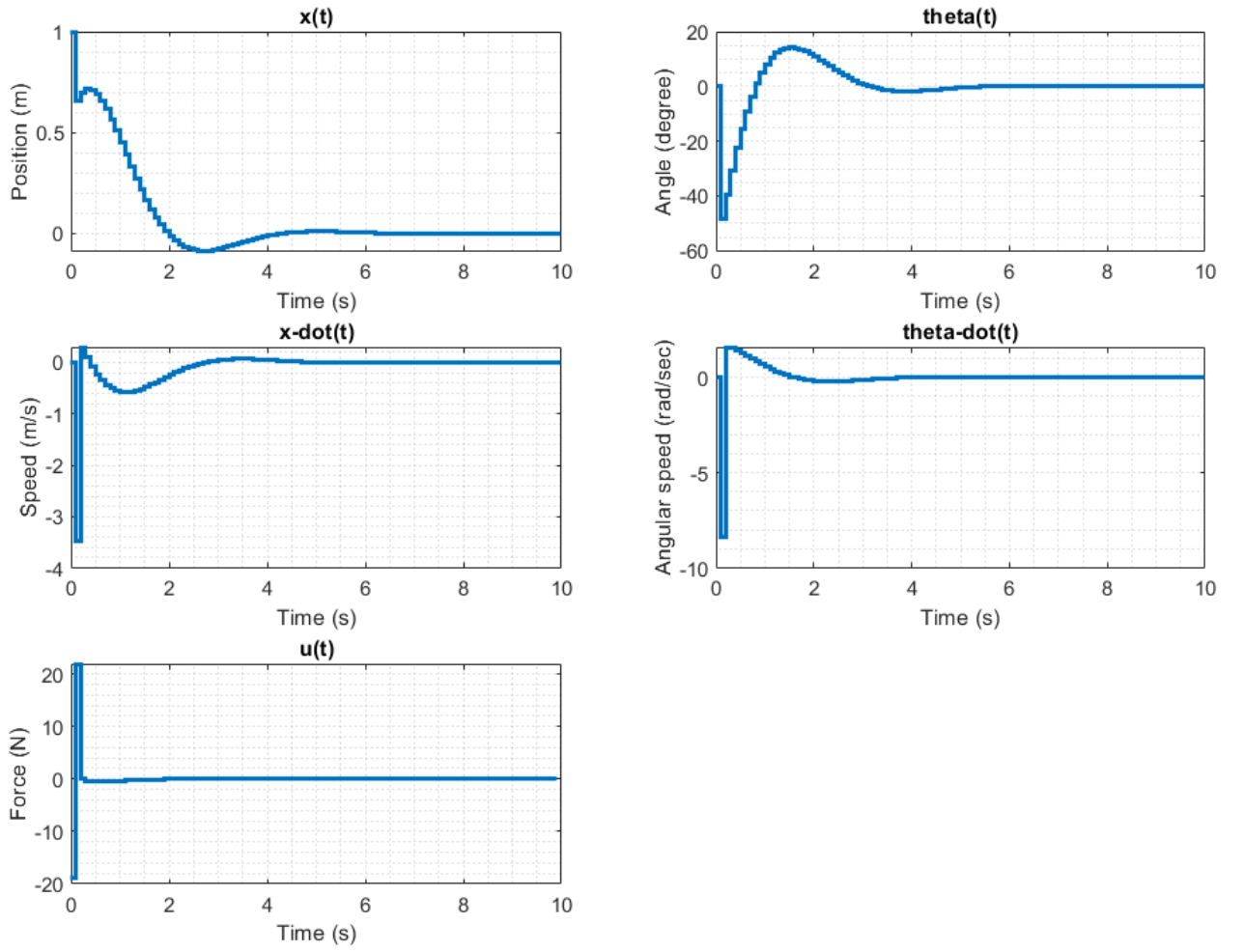
*Figure 10: LQR result for R=0.01 and Qf=Q=1000*C'C*

With increased state weights and relatively low input weight, systems settling time and overshoot performances are considerably improved. However, maximum value of the input signal is 5 time larger than the case in the figure 8.

	Cost Matrices	Result
Figure 7	R=0.1 Qf=Q=C'C	System is Instable
Figure 8	R=0.1 Qf=Q=10*C'C	Input is successfully minimized. Overshoot is obtained in all states. Relatively long settling time
Figure 9	R=0.01 Qf=Q=10*C'C	Maximum value of the input is larger. Better transient performance in terms of overshoot and settling time.
Figure 10	R=0.01 Qf=Q=1000*C'C	Fastest response with the smallest overshoot. However, maximum value of the input signal is the largest.

Figure 11: Infinite horizon LQR result comparison table.

Since the control philosophy is defined so that the minimizing the input signal is more important than having low overshoot and fast response, cost matrices in figure 8 are selected. With these matrices, an infinite horizon LQR is designed and simulated. Results are given below.
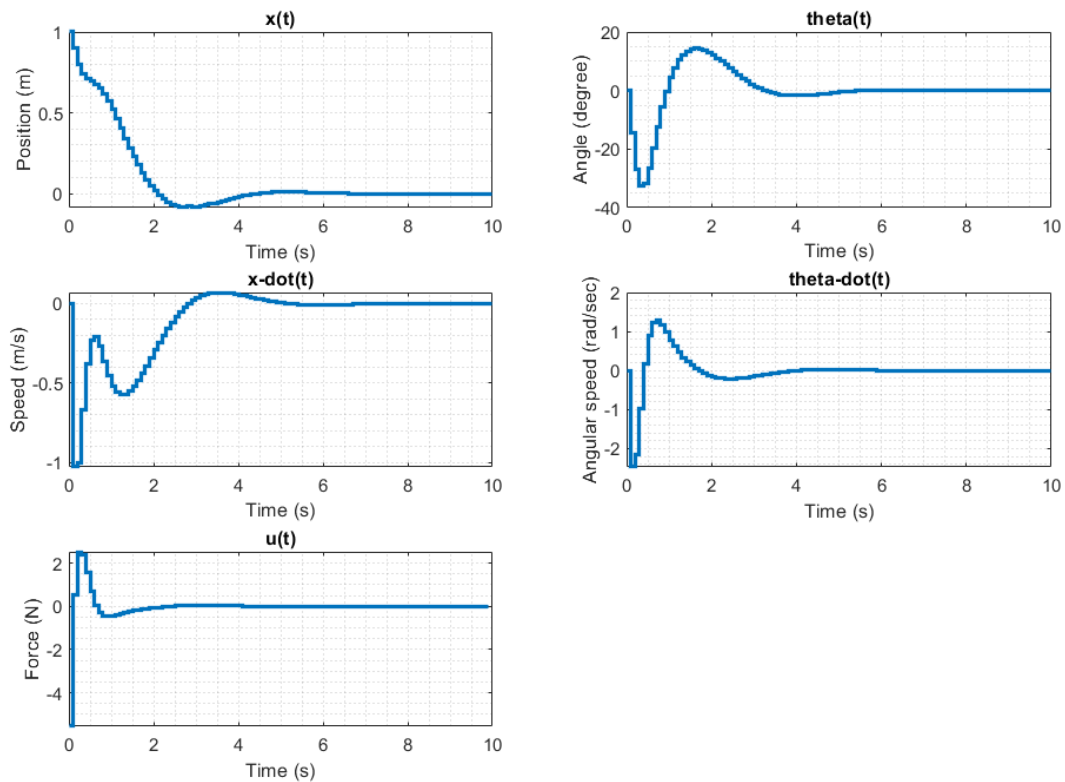
Figure 12: Infinite horizon LQR solution

Note that the performance of the Infinite horizon LQR is better than the finite horizon LQR with the same cost matrices (fig. 8 and fig. 12). It has smaller overshoot, smaller input signal magnitude and faster response. Finally, the controller is designed. Now it will be tested with nonlinear system model in Simulink and will be exposed with some disturbances or saturation.

1.d. Simulation with Nonlinear Model

In this part, both linear and nonlinear models are implemented in Simulink and simulated. Results shows the system behavior when the cart is ordered to go from one point to another by providing required step signal at position state.

Figure 13: Simulink Model of Linearized System

Figure 14: Simulink Model of Nonlinear System

Figure 15: Nonlinear Plant Model

At the time=3 seconds, a step signal is applied into the position signal (first state variable). System response with linear and nonlinear plants are shown below in figures 16 and 17. Note that the simulation with nonlinear plant gives similar results with the linear case. Input minimization is even better with nonlinear model. We can say that the designed LQR can control the nonlinear plant.
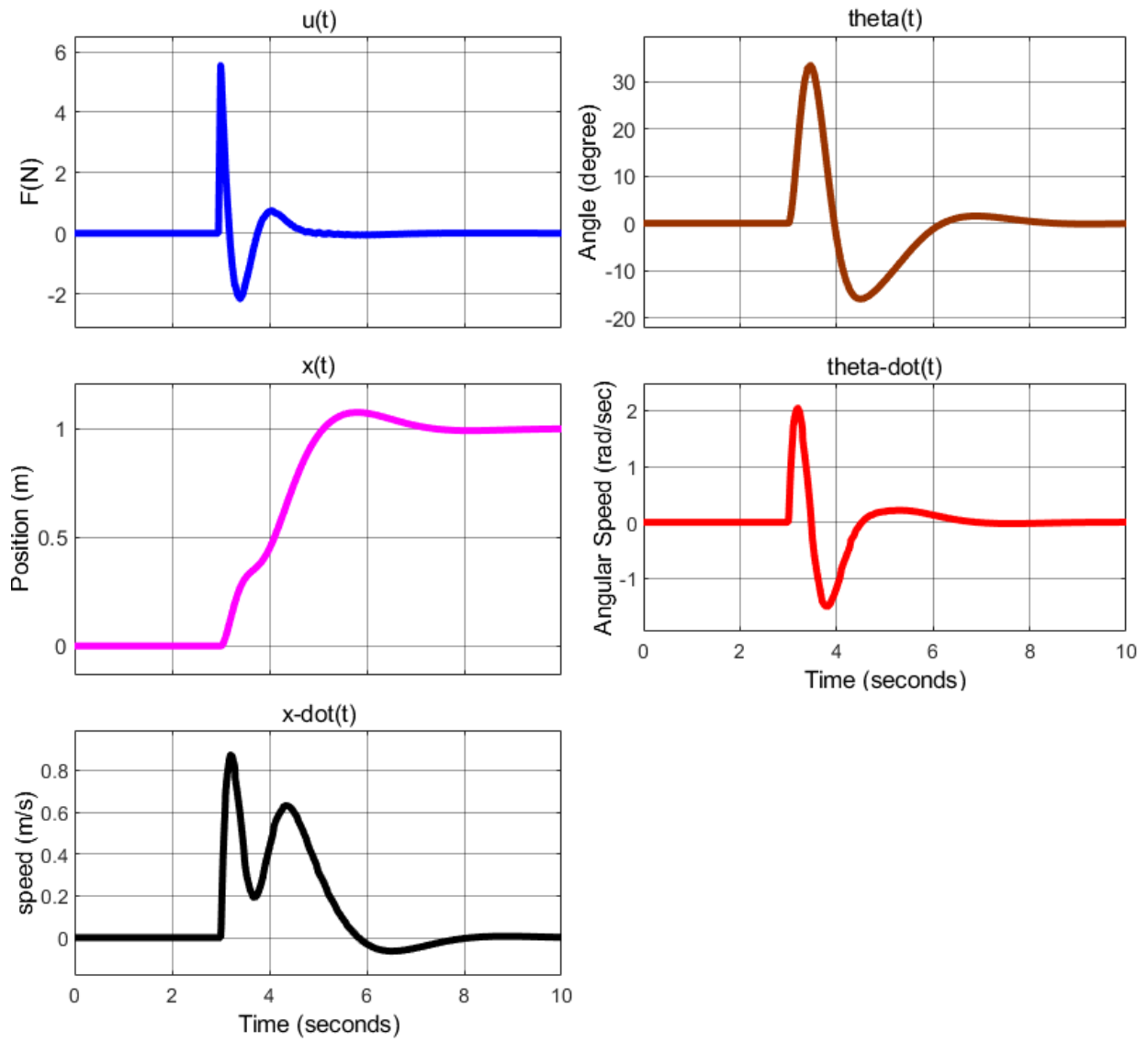
Figure 16: Simulation result with linear model
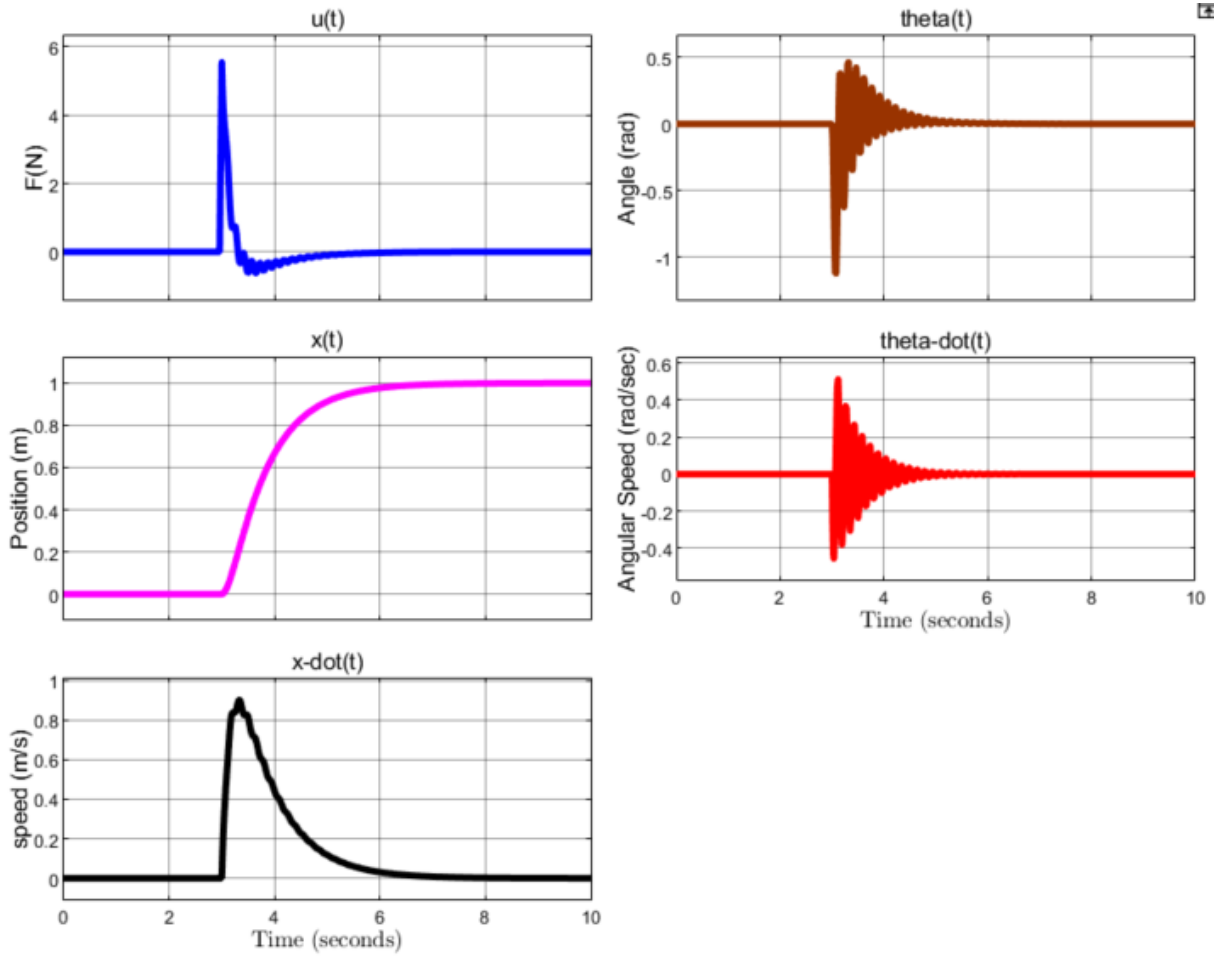
Figure 17: Position step response (1m) with nonlinear model

As figure 17 shows, nonlinear system is contolled succesfully with designed Infinite horizon LQR.

Until now, only the unit position step response performance is shown. Now, effect of larger steps, disturbances and saturation are going to be inserted into the nonlinear model and results are going to be discussed.
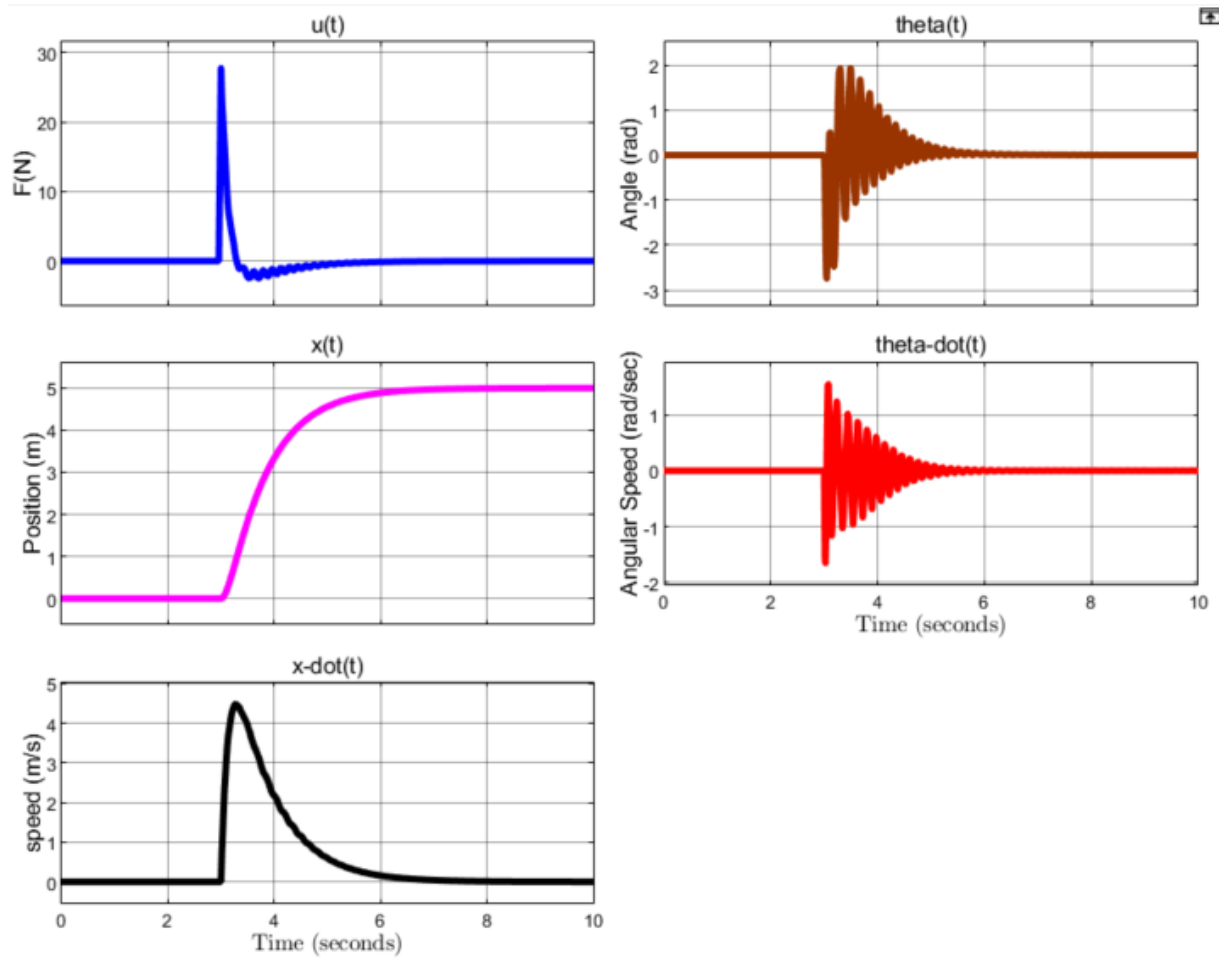
Figure 18: 5m position step at t=3
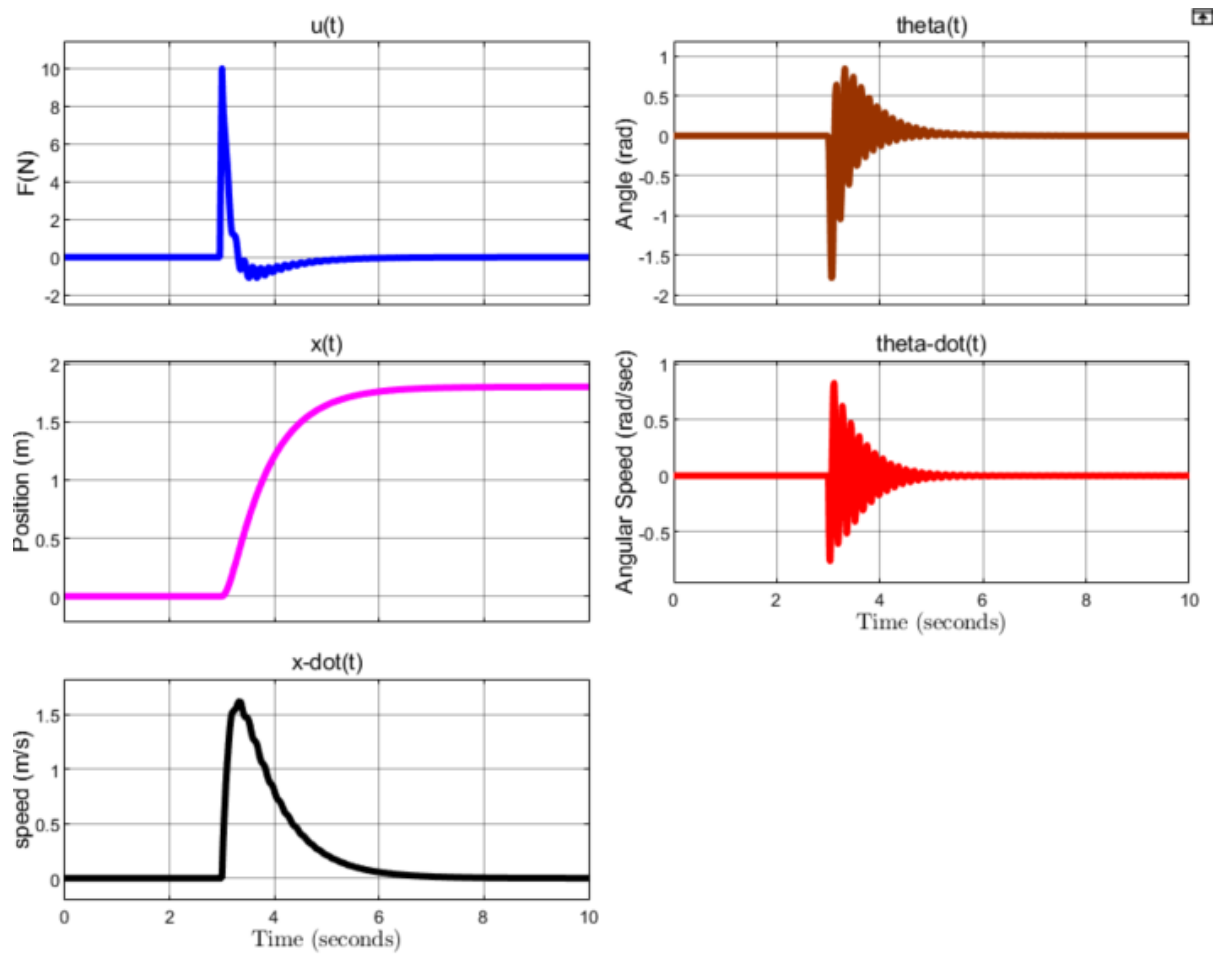
Figure 19: Input Disturbance Step Response (10x(unit step))
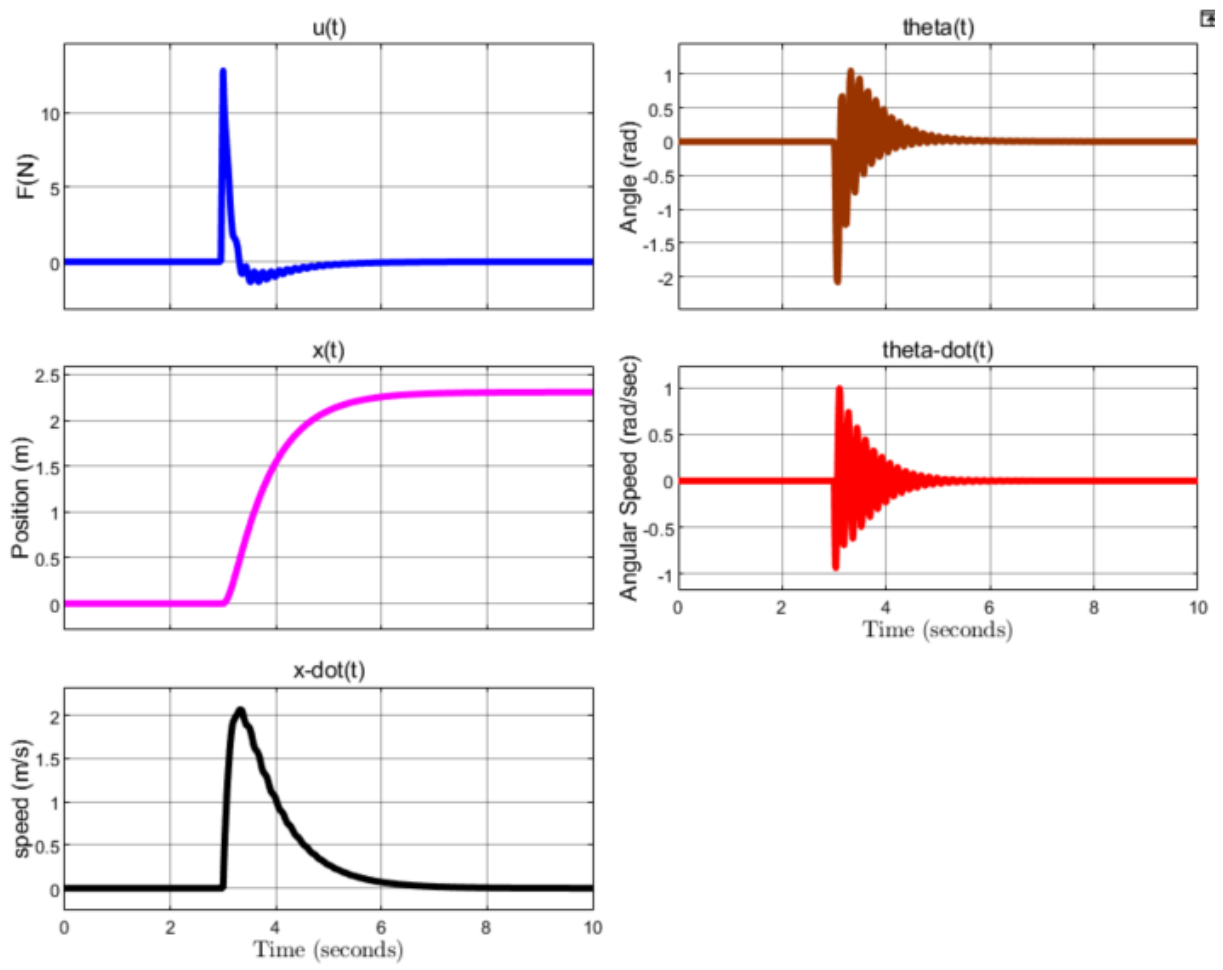
Figure 20: Angle Step Response (2 rad)
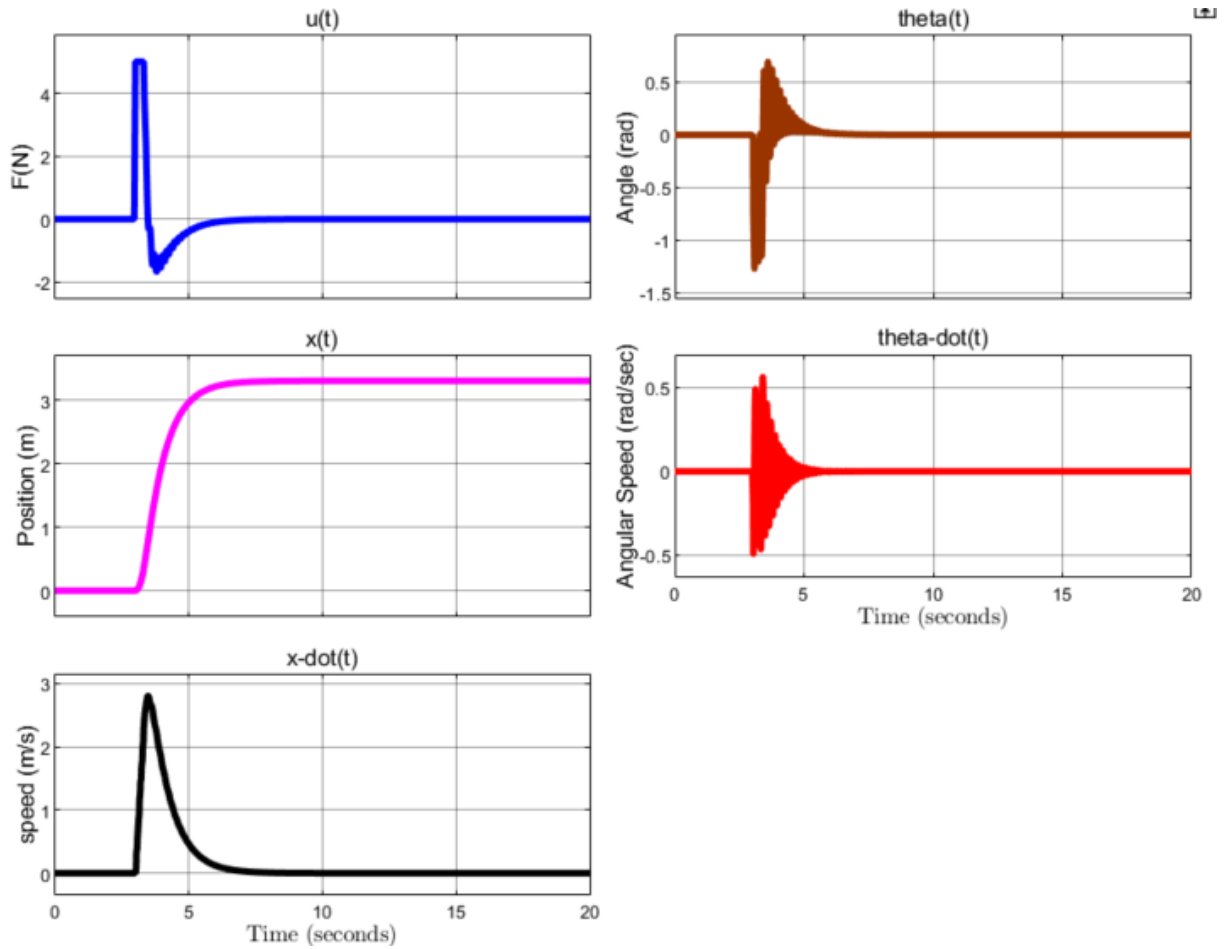
Figure 21: Position step response with input saturation (-5<u<5)

Figures 18, 19, 20 and 21 show the system response under different conditions. When larger state and input steps are inserted, magnitude of input signal increases and states have relatively larger oscillations and settling time. Input saturation is included in figure 21. Note that the LQR is vulnerable in case of input saturation. Keeping input weight large is a solution for small steps. However, having large input signal in inevitable when larger steps are inserted. In case of avoiding saturation is not possible, a MPC should be implemented instead of LQR.

Part 2 – Simulation

2.a. Simulink

In this part, trajectory model of the Apollo satellite is implemented and simulated in Simulink. Maximum step size, relative tolerance and solver type is changed, and results are compared. Also, fixed step size is tried.
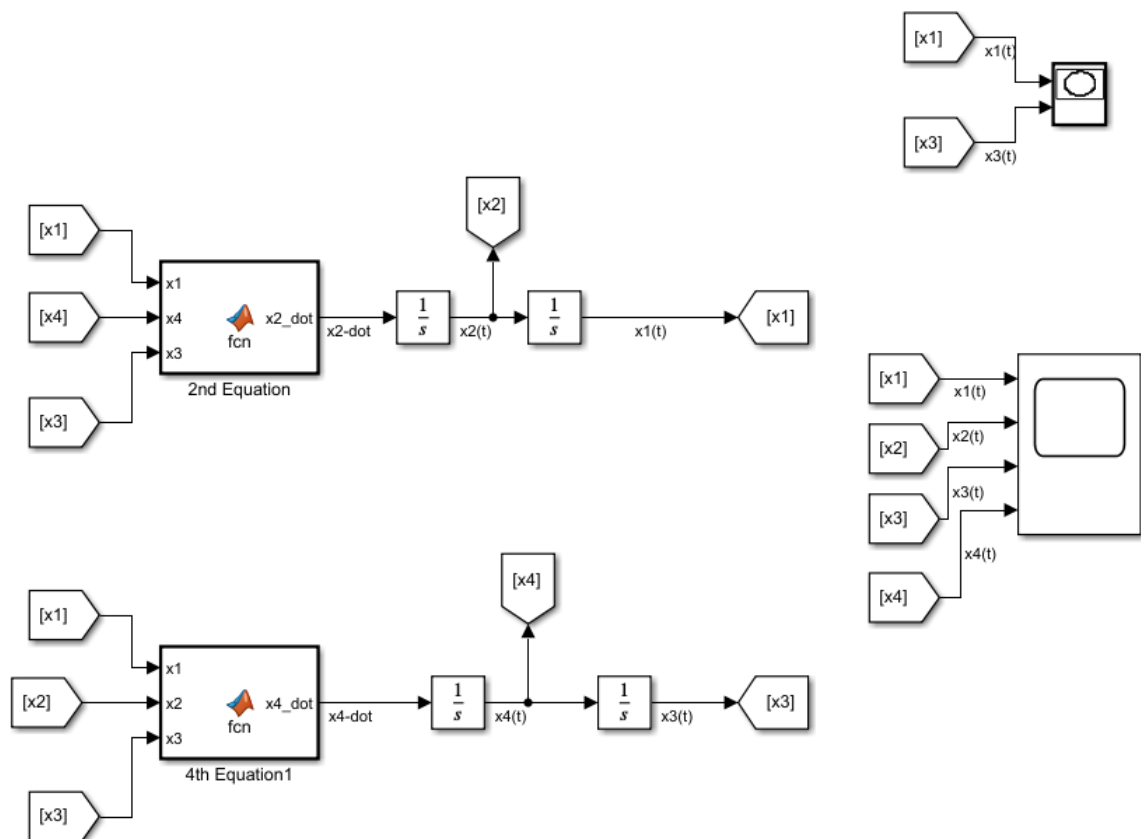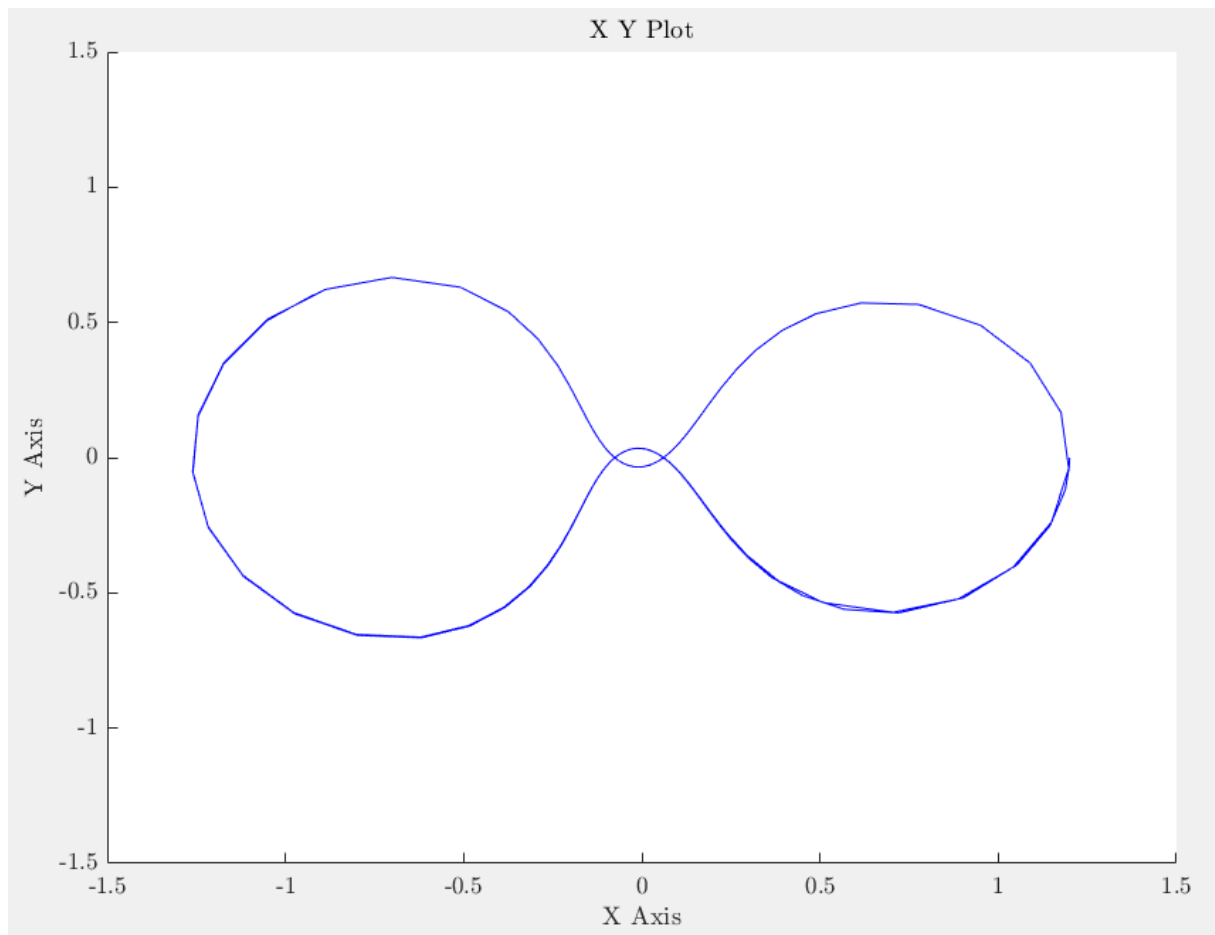

Figure 22: Simulink Model

Figure 23: XY plot with the settings given in project description.

This figure shows the resulting XY plot with given settings. This plot shows the trajectory of the satellite in XY plane. Now, different variations of maximum step size, relative tolerance, solver type and having fixed step size or not are going to be compared.
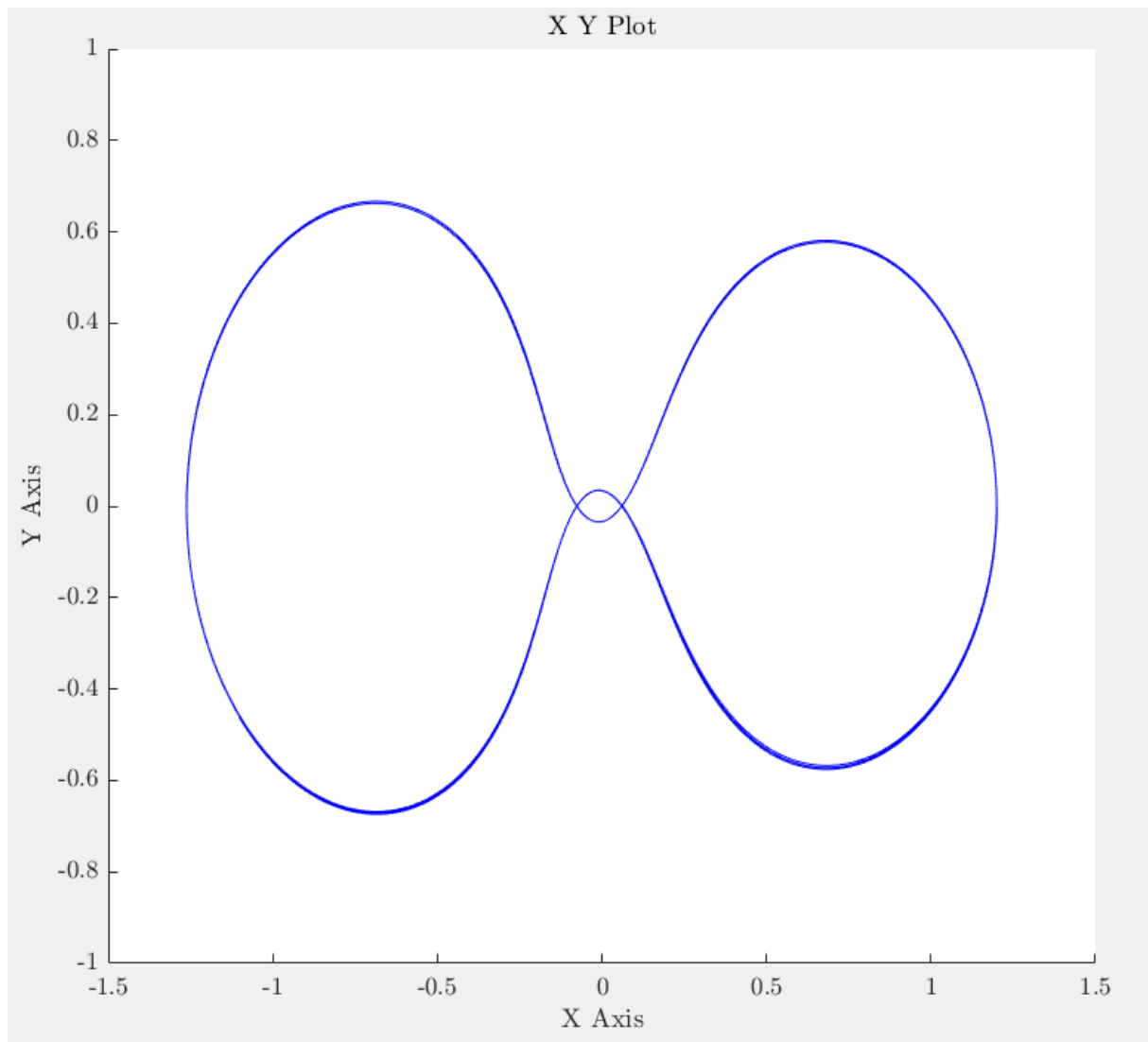
Figure 24: Maximum Step Size = 0.01

Keeping other parameters as given, changing maximum step size resulted in a smoother XY plot. This is due to limited step size. Even if error is bounded, step sizes are bounded so that solver is not allowed to take big steps. This makes XY plot smoother with smaller steps.
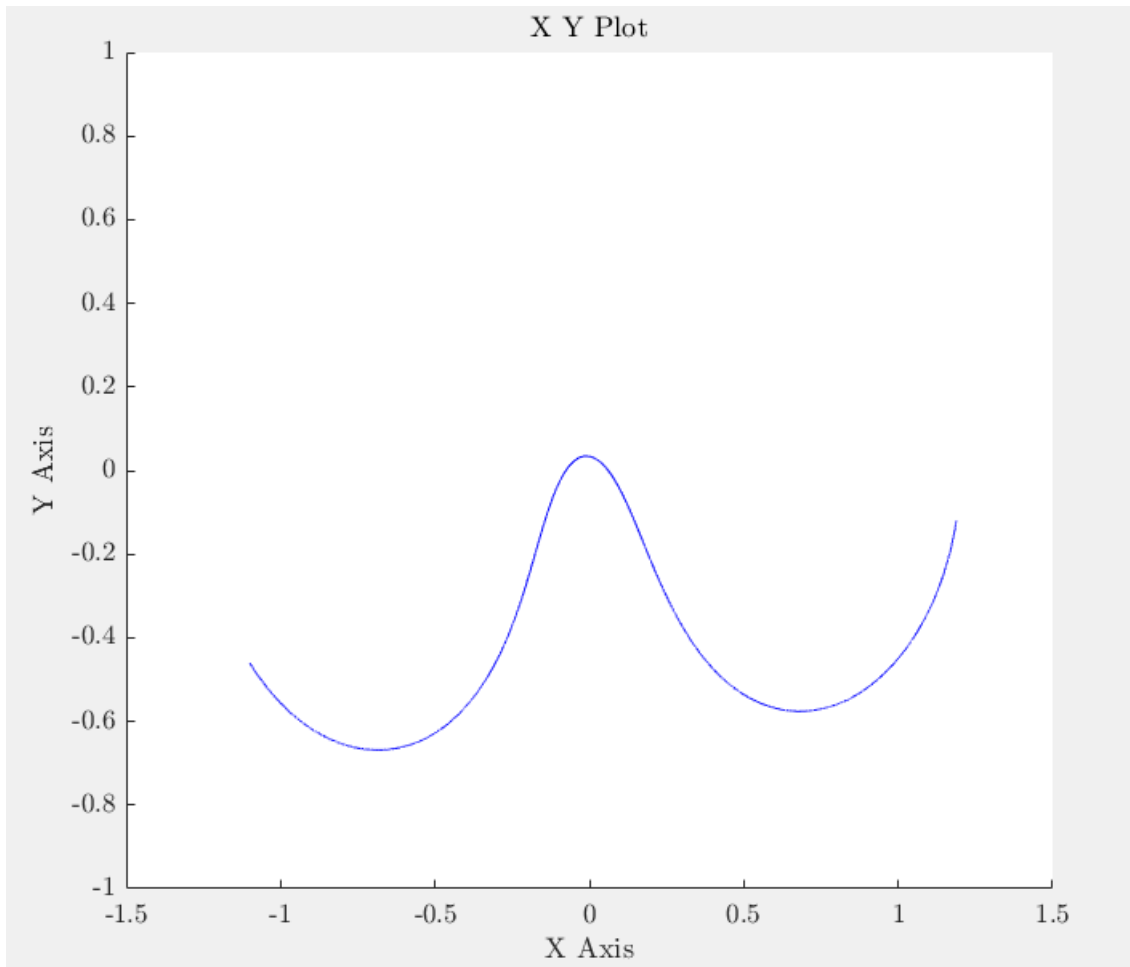
Figure 25: Max Step Size = 0.0001

Maximum step size is decreased even more. In this case, XY plot is extremely smooth, but simulation is so slow so that it cannot even provide the correct results. During the simulation, one can see the trajectory moves forward as time passes. Due to too small steps, simulation is too slow and unsuccessful.

Now, let us keep the maximum step size at 0.01, which gives both smooth and fast result, and change the relative tolerance.
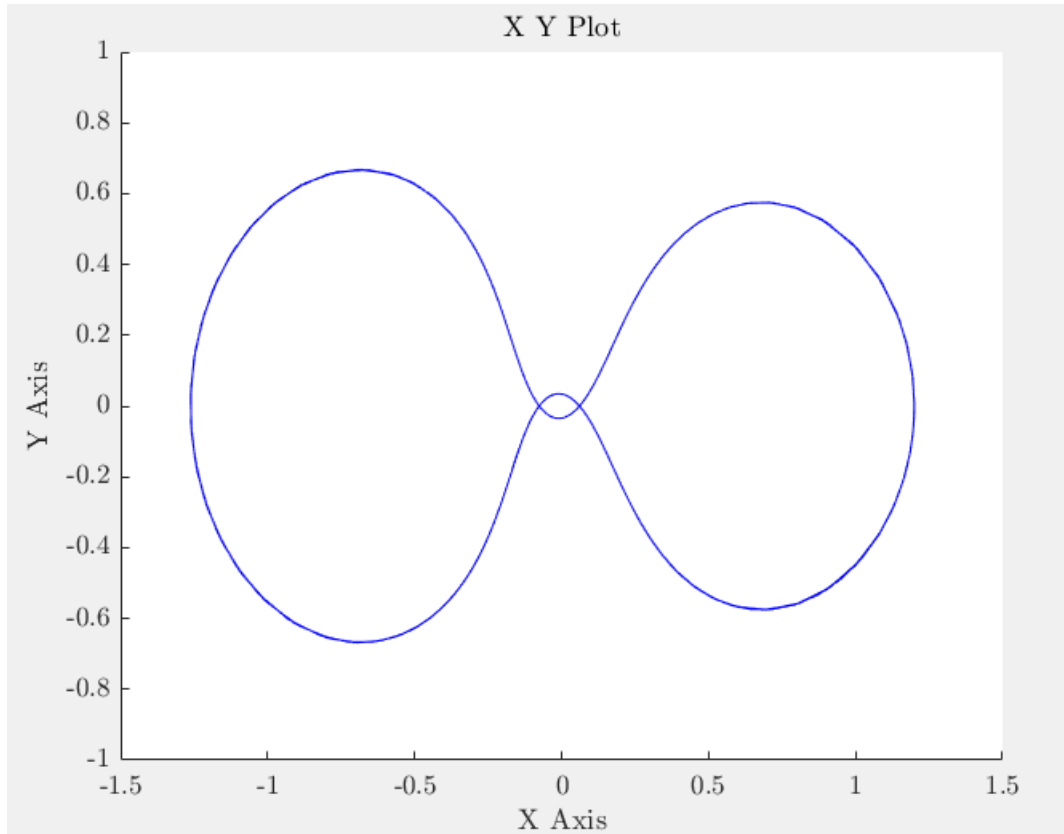
Figure 26: Relative tolerance = 0.0000001

With very low relative tolerance and relatively low step sizes, resulting XY plot is smoother than figure 19. Also note that, error in the trajectory is decreased as compared the figure 20, which has same maximum step size but larger tolerance. Especially when the satellite orbits with a high radius, errors cause a thick blue line of trajectory in figure 20. However, when we tightened the relative tolerance, error in the trajectory is decreased.
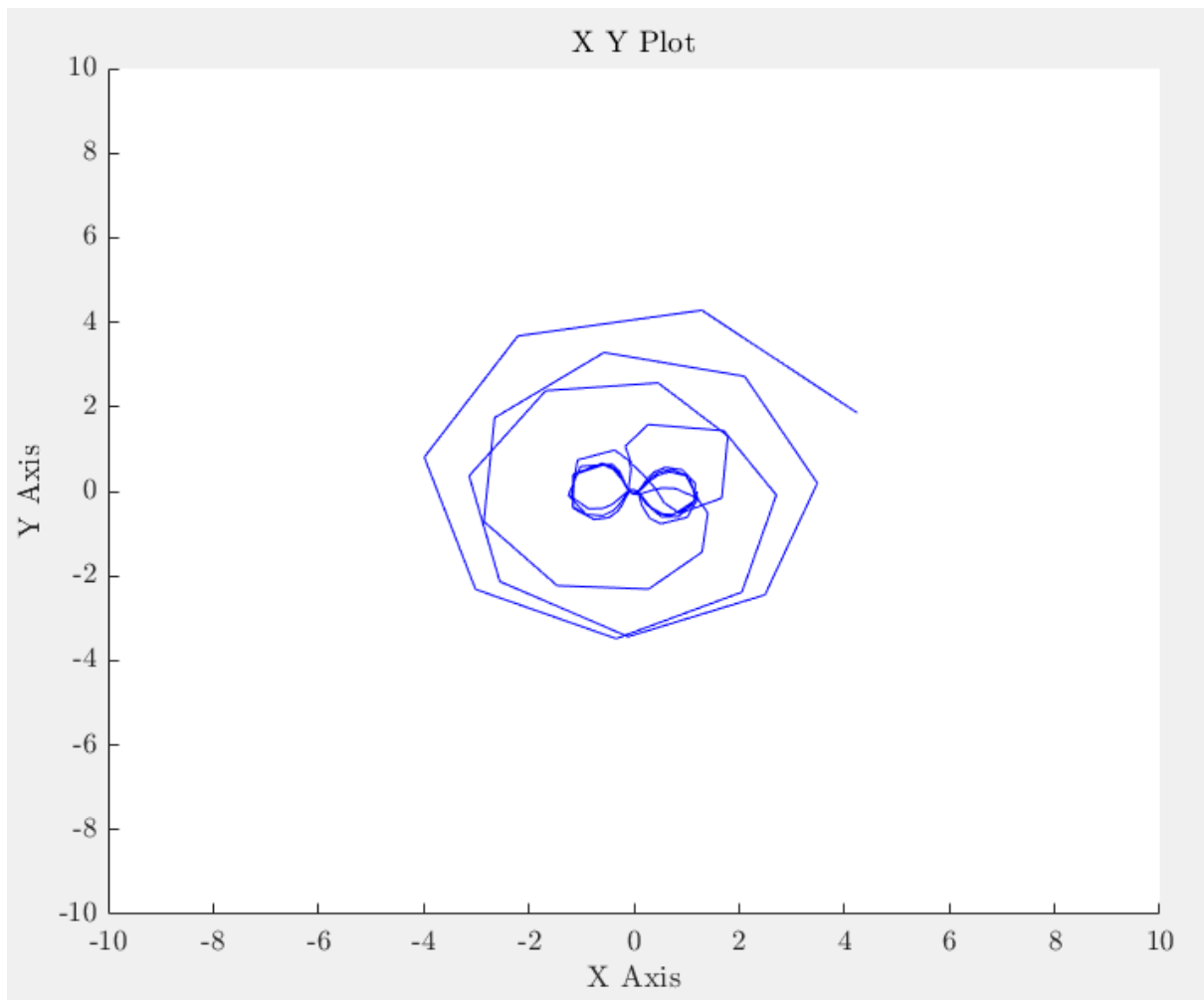
Figure 27: Relative Tolerance = 0.001

With increased tolerance, error in the trajectory is too large so that the result is not correct. We can say that large tolerance causes fast but erroneous simulation. Now let us try different solvers to see how the results change with different solver types.
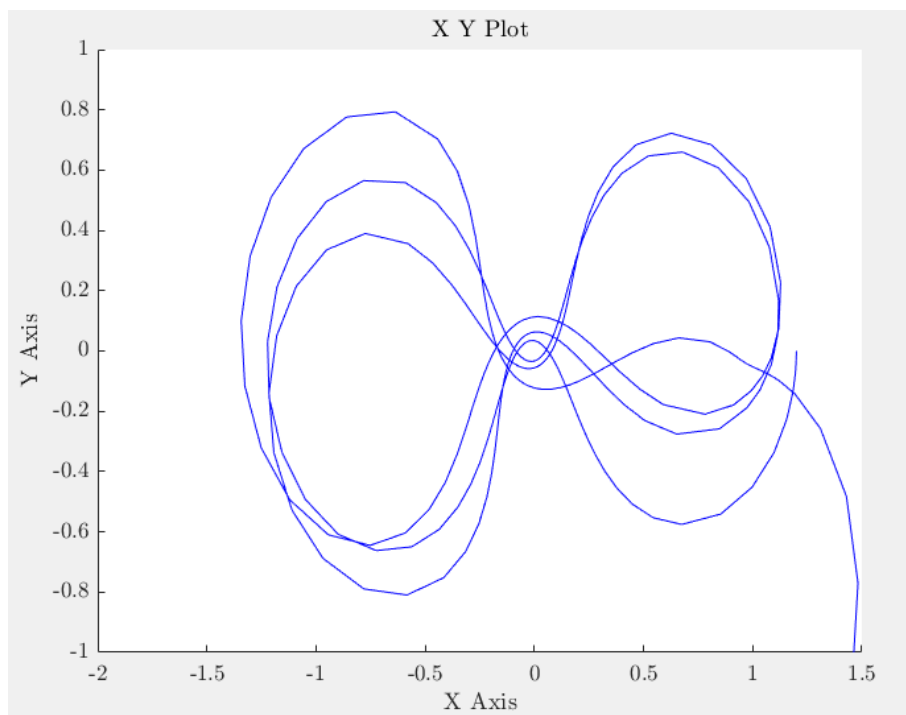
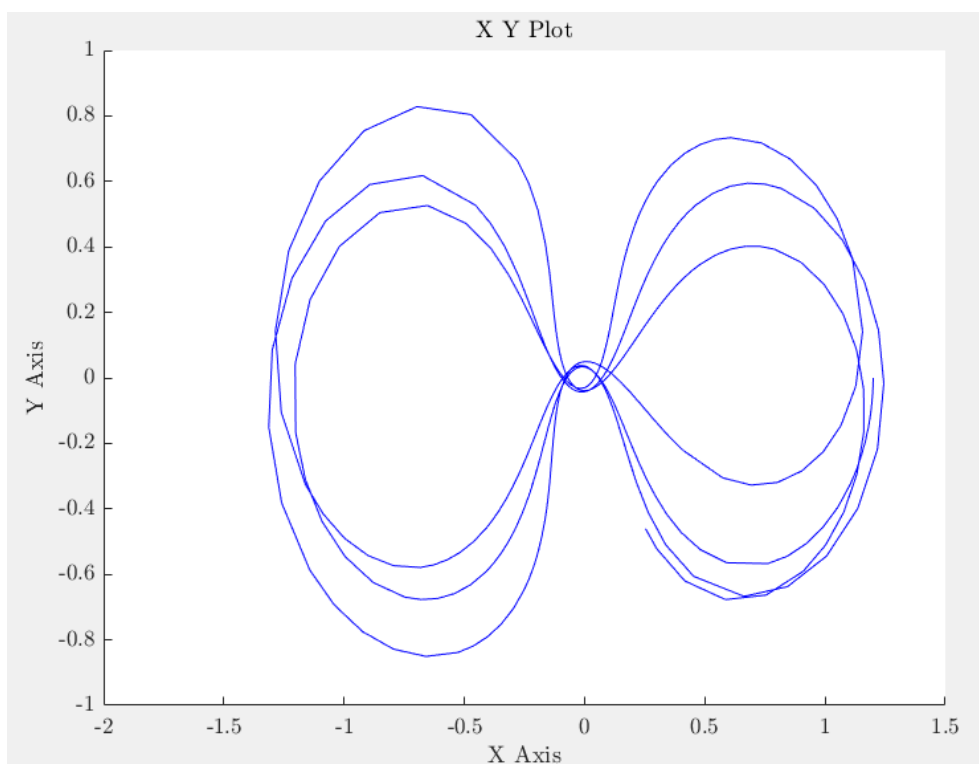*Figure 28: XY plot with the solver ODE23*

Figure 29: XY plot with the solver ODE15s

As seen in figures 24 and 25, these solvers provide high error rates. Step size and tolerance values must be adjusted so that these solvers also can give better results. Now, let us try fixed step size.
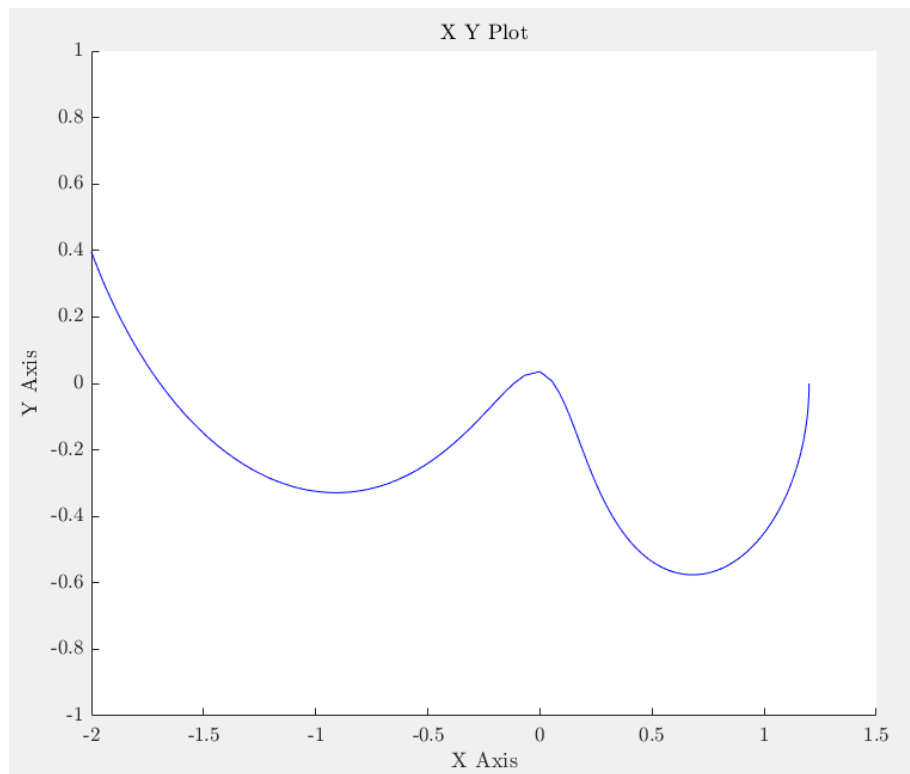

Figure 30: XY plot with fixed step size = 0.01

As seen in figure 26, keeping step size large causes simulation result to diverge. This step size is too large. Let us try a smaller one.
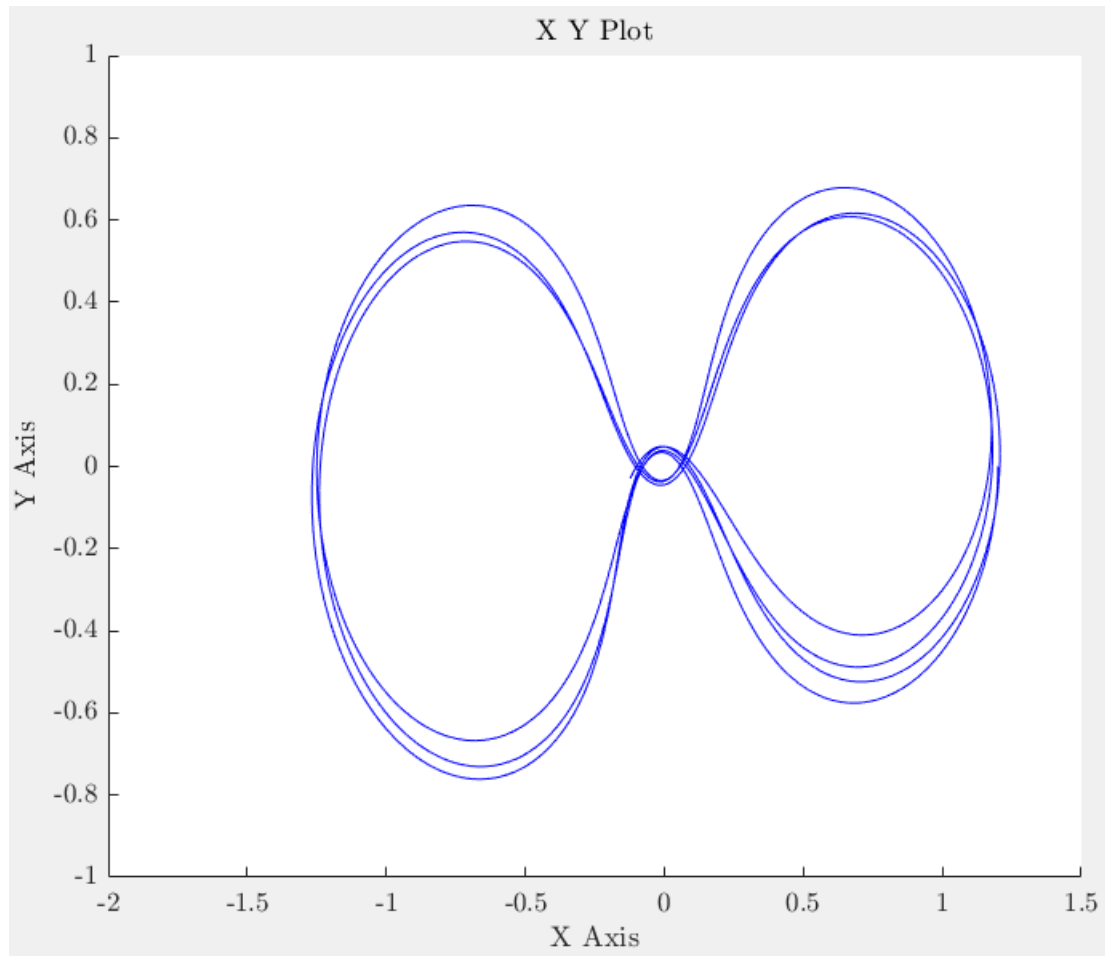
Figure 31: XY plot with fixed step size = 0.001

With the decreased step size, result is better. However, fixed step size results in a slow simulation. Even if it is fixed, it2 value must be kept small enough to have a correct result.

2.b. Runge-Kutta with Fixed Step Size

In this part, a Runge-Kutta method with order 4 is implemented. Related code block is provided in the appendix section. Simulation result is provided below. As seen in the figures 28 and 29, fixed step size h=0.0001 s gives a more accurate result while still having short calculation time as in h=0.001 s case. Therefore, h is selected as 0.00001 seconds.
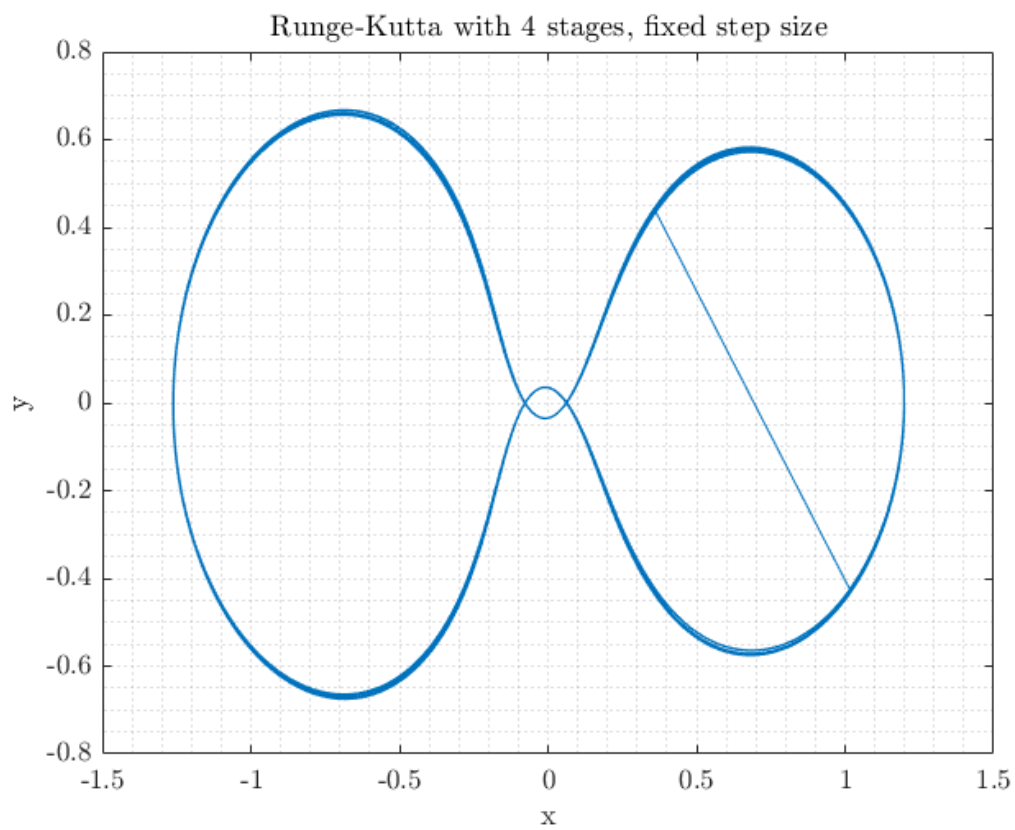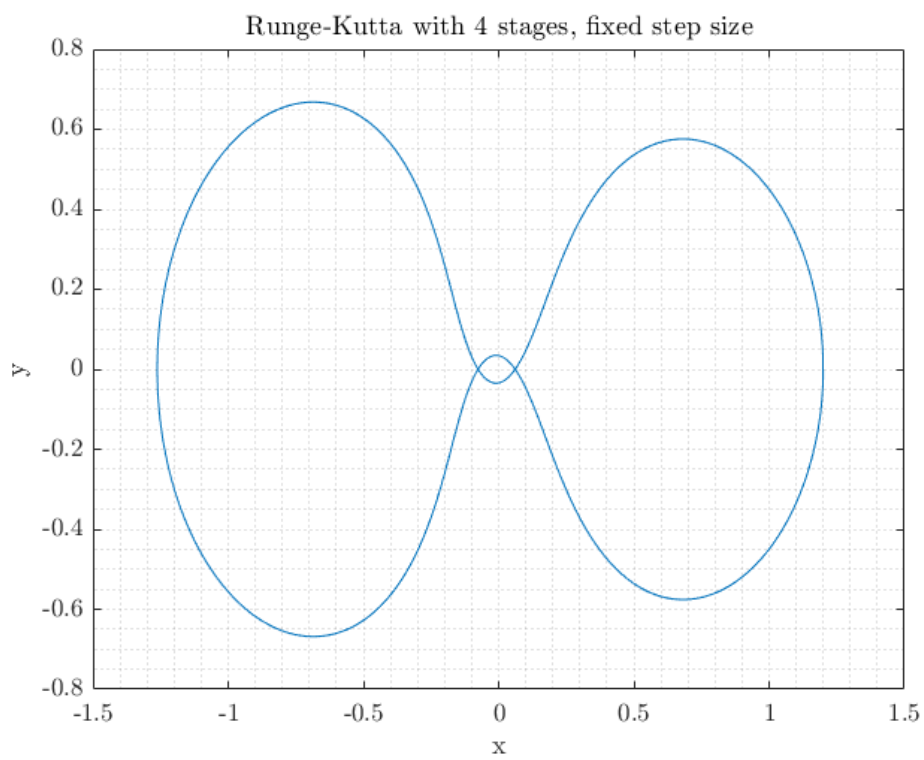
Figure 32: h=0.001

Figure 33: h=0.00001

32

2.c. Step Size Adaptation

Now, a Runge-Kutta method with order 4 and variable step size is going to be implemented. According to the book section provided, variable step size is obtained by following the errors at each step. If error exceeds a predetermined limit, step size is decreased to decrease the error. However, simulation becomes slower in such a case. On the other hand, variable step size may cause simulation to be faster in some cases. If error is less than the boundary, step size is increased to obtain the result faster. Definition of the boundary and error calculations are done according to the application. Such parameters affect the speed and accuracy of the simulation and designer must determine their values accordingly.

In this application, Heun's 3rd order method is used for error estimation and error boundaries are determined by using the formulas given in the book chapter provided on Odtuclass. Related code blocks are provided in the appendix section.

2.d. Variable Step Size Results

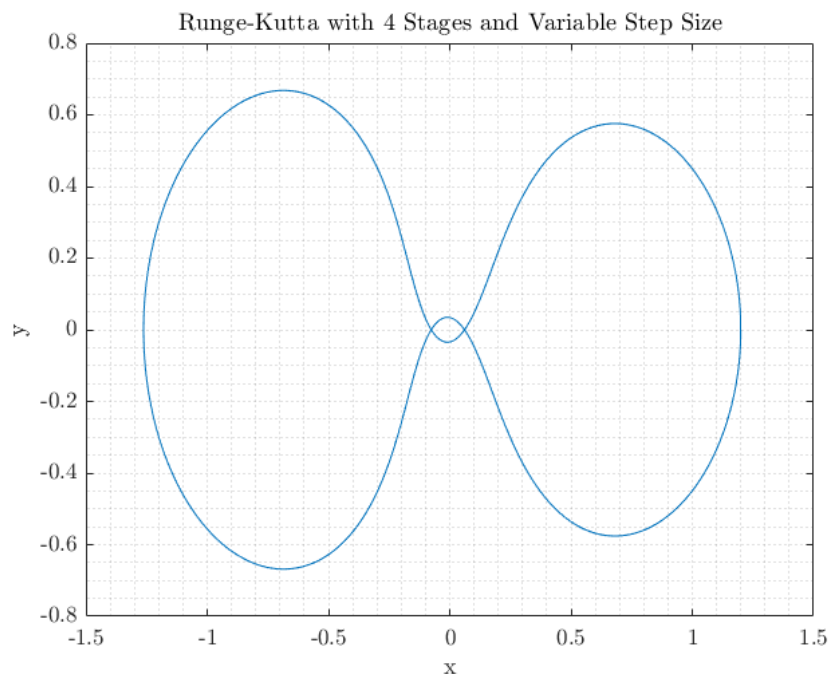As seen in figures 30 and 31, variable step size solution is as accurate as fixed step size solution and faster.

Figure 34: Simulation with Variable Step Size

time_fixed	16.1567
time_variable	3.9625

Figure 35: Simulation durations of variable and fixed step sizes
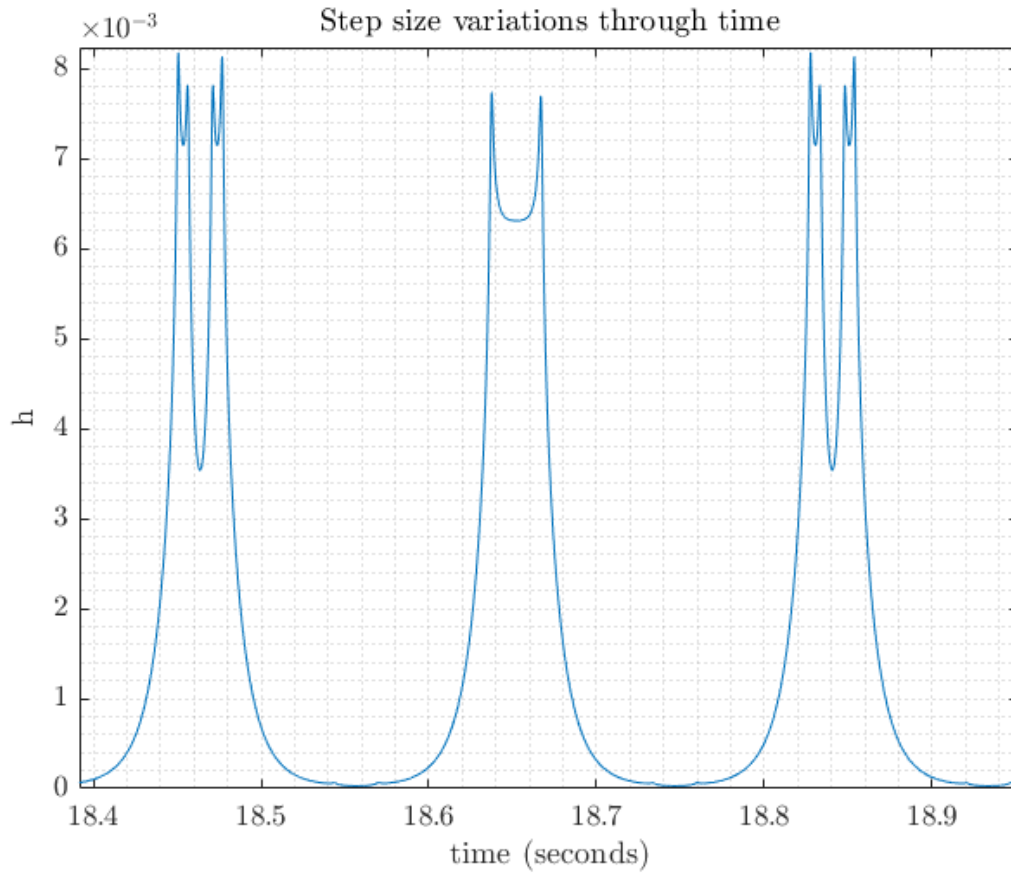

Figure 36: Step size variations

Change in the step size with respect to time is given in figure 32. When the error is large, step size is decreased to suppress th error. This region is coincident with the duration when the satellite orbits with a large radius. Because error in such a calculation is larger compared to a small radius turn. When the satellite is close to the earth, i.e., orbiting radius is small and error between the calculations is low. Therefore, step size is increased. In overall, this method is faster than the fixed step size one because its step size is equal to 0.00001 seconds independent of the error.