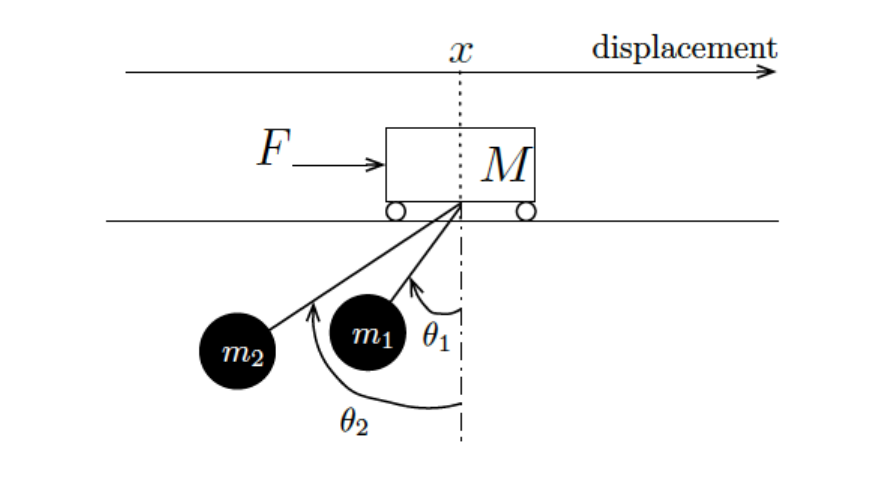
Final Project - ENPM667

Mack Tang and Ramzi Sayegh



# Equations of Motion and Non-linear State Space Representation

To solve for the equations of motion of this system, we used the Euler-Lagrange method. The three equations of motion for this system will be computed using:

where

The process of solving for these equations of motion is shown below. Note that for the entire report the argument of time for each variable is commonly omitted for neatness, i.e., .

And it follows that

So we can write our equation for *L* as

Then we solve for the first equation of motion given by

So our first equation of motion can be written as

Now we can solve for our second equation of motion given by

And this can be simplified to obtain our final second equation of motion as

Following the same procedure used to find the second equation of motion, we can see that the third equation of motion is simply

Now that we have our three equations of motion for this system, we can represent the nonlinear system in state space form as:

\*\*insert non-linear state space representation

The MATLAB code used to solve for this representation is shown in \*\*Appendix? Non-linear rep.

# Linearization of System around Equilibrium Point

To simplify analysis and simulation of this system, we must obtain the linearized state space representation. To do this, we linearize our system about the equilibrium point specified by . This process is done using the following steps:

\*\*insert steps to linearize about equilibrium

And the resulting linearized state space representation is

\*\*insert linearized system in state space form

# Obtain Conditions on *M , m1 , m2 , l1 , l2* for which the Linearized System is Controllable

To find relationships for these variables that make the system controllable, we want the rank of matrix [B AB A2B A3B A4B A4B] to be equal to 6, because n=6for this system. This matrix is as follows

\*\*insert controllability matrix

To determine when this matrix is full rank, we want the determinant of the matrix not equal to 0. Using MATLAB’s *det* function, we see that the determinant is given by

\*\*insert determinant of controllability matrix

This determinant is equal to 0 when l1=l2. So as long as l1l2, our controllability matrix has full rank and thus the system is controllable.

# Check System is Controllable with given values and Obtain LQR Controller

Here, we are given that *M* = 1000*Kg*, *m1* = *m2* = 100*Kg*, *l1* = 20*m* and *l2* = 10*m*. Substituting these values into our A matrix for the linearized system gives

\*\*insert A matrix with values

Then we can check that this system is controllable using the same matrix test used inpart **C**. The controllability matrix reads

\*\*insert controllability matrix B AB A2B… with values

We can see that this matrix is full rank by taking the determinant.

\*\*insert determinant of matrix

Which is not equal to 0. Thus this system is controllable with these values.

Now we wish to design an LQR controller for this system. We want to find a K that minimizes the cost function

\*\*insert cost function J

where K=-R-1BKTP, and *P* is the symmetric positive solution of the following Riccati equation ATP+PA-PBR-1BTP=-Q. Here, *Q* and *R* are chosen to best simulate the response and were chosen as follows.

\*\*insert Q and R final

Using MATLAB’s *icare* function, we were able to solve for matrix *P* as

\*\*insert P matrix

Then *K* is calculated as

\*\*insert K matrix

Lyapunov’s indirect method to certify stability states that if the eigenvalues of the closed-loop *AF* matrix have all negative real parts, the system is locally stable around the equilibrium point. Here, the closed-loop system is

\*\*insert closed loop xdot = (A+BK)x

and the eigenvalues of (A+BK) are calculated in MATLAB as

\*\*insert eigenvalues of A+BK

Clearly, the eigenvalues all have negative real parts, so this certifies that our closed-loop system is locally stable. Since Lyapunov’s indirect method was applied to the linearized system, we can conclude that local stability implies global stability.

Now we apply the LQR controller to both the linearized and non-linearized system and simulate the response to initial conditions. The resulting plots of linear and non-linear simulations for each of our state variables are shown below.

\*\*insert simulations linear and non-linear

Discuss plots…

# Determine for which Output Vectors the Linearized System is Observable

Given four output vectors: *x(t)*, *(θ1(t)*, *θ2(t))*, *(x(t), θ2(t))*, *and (x(t), θ1(t), θ2(t))*, we are tasked to determine for which of these vectors is the linearized system observable. To do this, we utilize the observability matrix rank test shown below. The system is observable if the rank of the observability matrix is *n=6*.

\*\*insert observability matrix general form

Each of these output vectors gives a different *C* matrix, and we can test each one for observability.

For the output vector of *x(t)*, our *C* matrix would read *Cx* = [1 0 0 0 0 0]. The other vectors would read as follows. *Cθ1,θ2* = [0 0 1 0 1 0], *Cx,θ2*= [1 0 0 0 1 0], and *Cx,θ1,θ2*= [1 0 1 0 1 0]. Then performing the rank test for each of these *C* matrices gives

\*\*insert 4 rank matrices

So we can conclude that the system is observable for output vectors:

*x(t)*, *(x(t), θ2(t))*, *and (x(t), θ1(t), θ2(t)).*

# Obtain Luenberger Observer for Each of the Output Vectors that Results in an Observable System

The Luenberger Observer has the following state space representation:

where *L* is the gain matrix and is the correction term, where , the limited state feedback that we do have access to measure, as indicated by C. The errors between the few measured states and same few estimated states combine into a scalar and are multiplied by L. Intuitively what the math is doing here is, assuming there is indeed a mismatch, is adjusted to speed up or slow down so that at the *next* timestep can be more accurate, so the estimated states don’t “fall behind”.

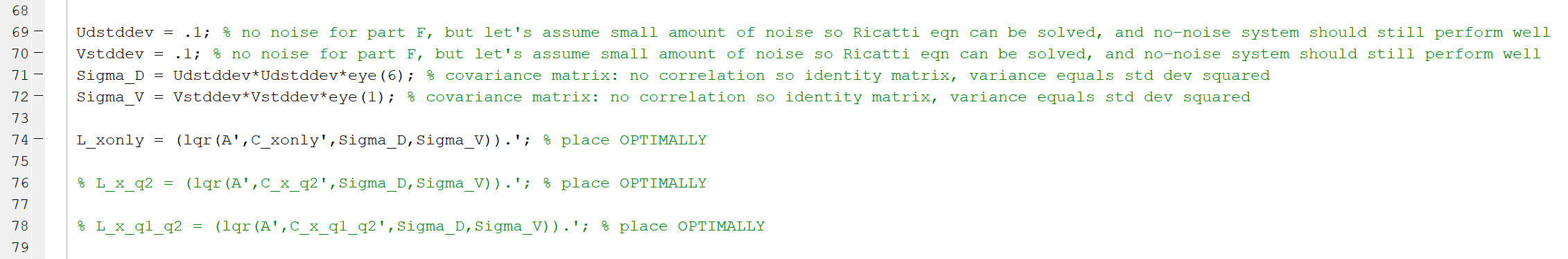
Defining the estimation error as , and hence , we can obtain through simple manipulations that , assuming that D=0. Note for simplicity we will drop the (t), but remember some of these variables are functions of time. We can convert back from to and from to , substitute in for Full State Feedback using the estimated state, and stack the state equations so we have one long 12x1 state vector, and a “combined A” and “combined B” matrix, such that:

This is the standard form the Luenberger Observer, with Full State Feedback where the control input . We can redo all of these steps with a unit step input instead, , and assume there is no noise for now, to obtain:

and similarly for the same linearized state estimation, but against the fully nonlinear system,

These are the Luenberger Observers with unit step input for linear and nonlinear system, where L and C depend on which output vector is selected. To solve for the best *L* for each output vector, we implement the Kalman-Bucy method, which uses the same LQR mathematical framework of solving the Algebraic Ricatti equation, but using in place of the usual and and in place of the usual Q and R. Note that as we will see soon, and have meanings concerning noise, but for now since we have no noise they merely are tunable parameters to help us find the “best” or most optimal L such that the eigenvalues of are placed in a good place. Note that it was found that simply trying to place the eigenvalues at an arbitrary very negative position did not result in good estimation performance, so assuming a small amount of noise and then placing at the optimal eigenvalues (even while there is no noise) is a good idea.

Then we can find the optimal , where *P* is the solution to . As we have covered the solving of the Algebraic Ricatti Equation previously, we will now augment MATLAB’s *lqr* function to solve the Ricatti equation for our optimal *L*:



The resulting simulations of the Luenberger observer applied to the linear system with 0 initial conditions, for each output vector, are shown below.

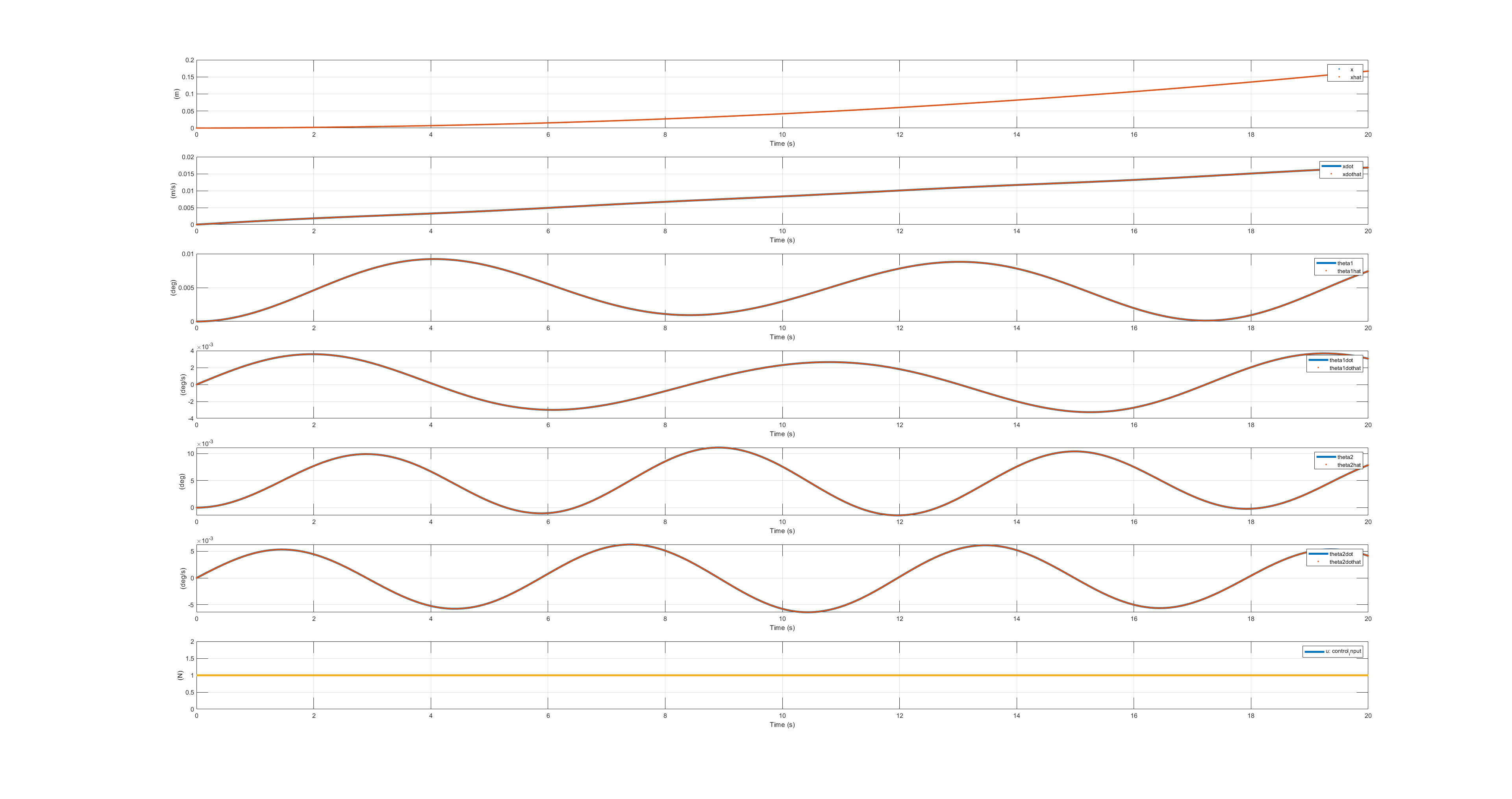


Figure : Output vector x only, 0 initial conditions, unit step input.

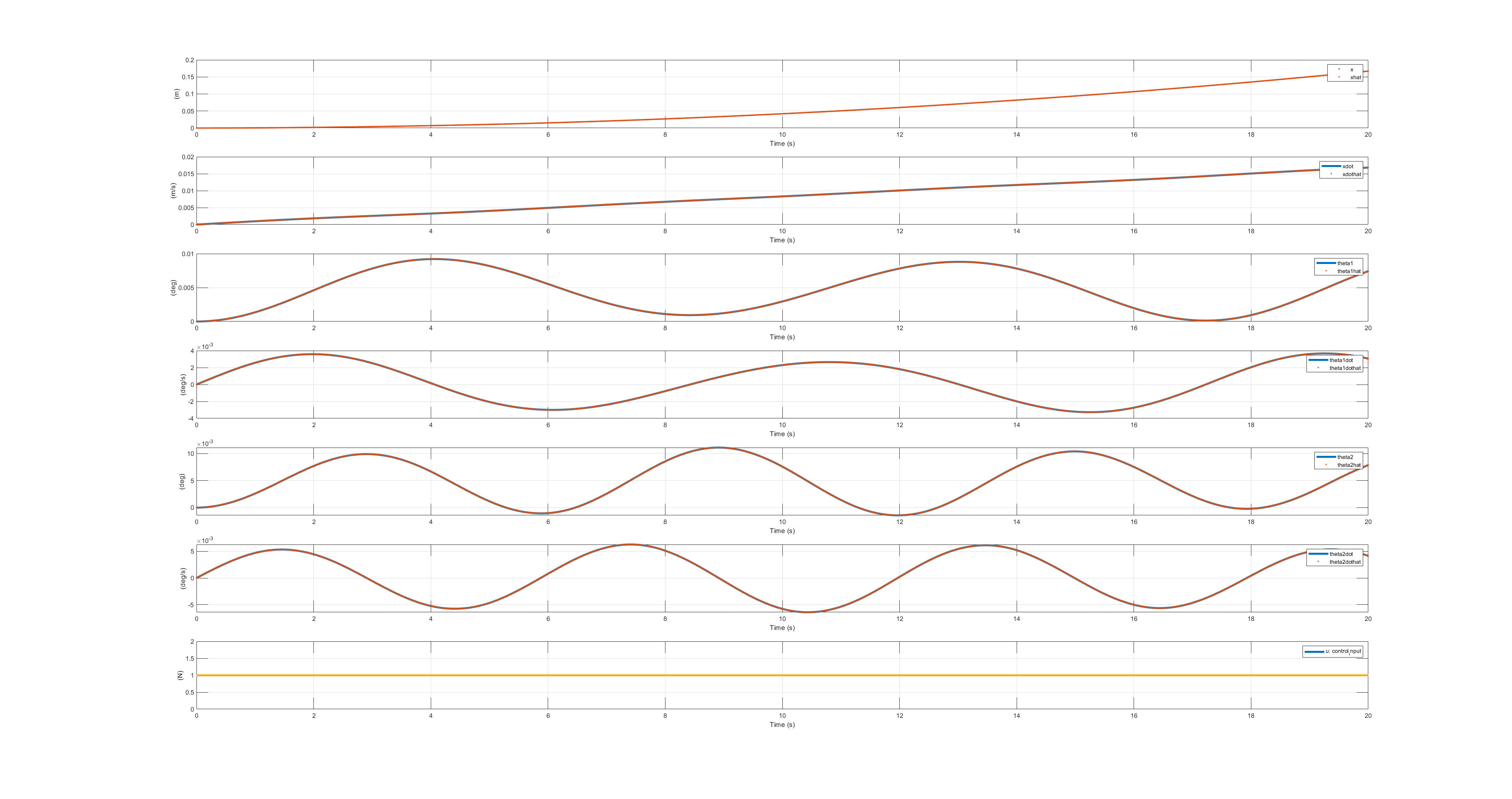


Figure : Output vector (x(t), θ2(t)), 0 initial conditions, unit step input.

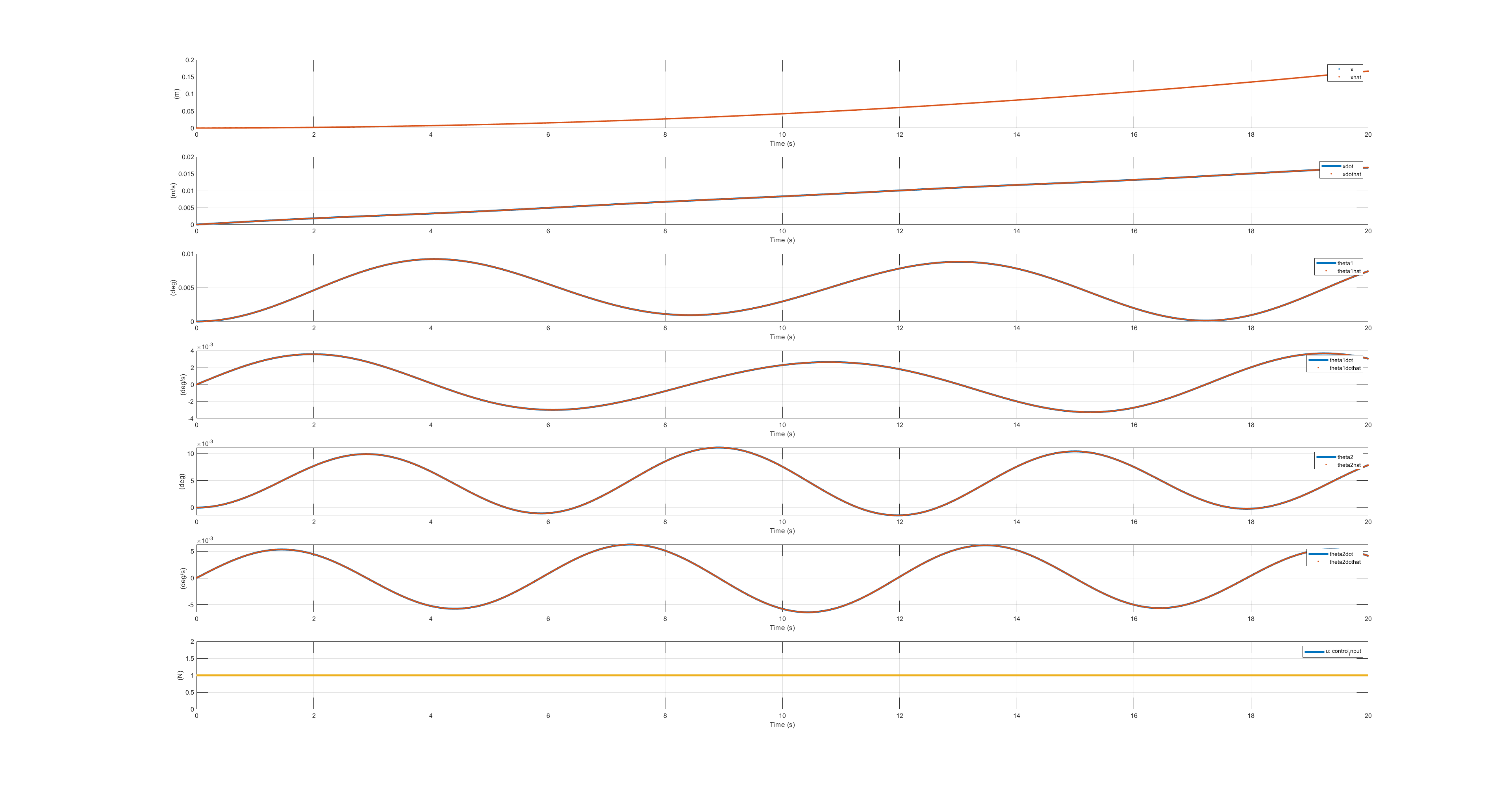


Figure : Output vector (x(t), θ1(t), θ2(t)), 0 initial conditions, unit step input.

We also obtained good results for the nonlinear system:

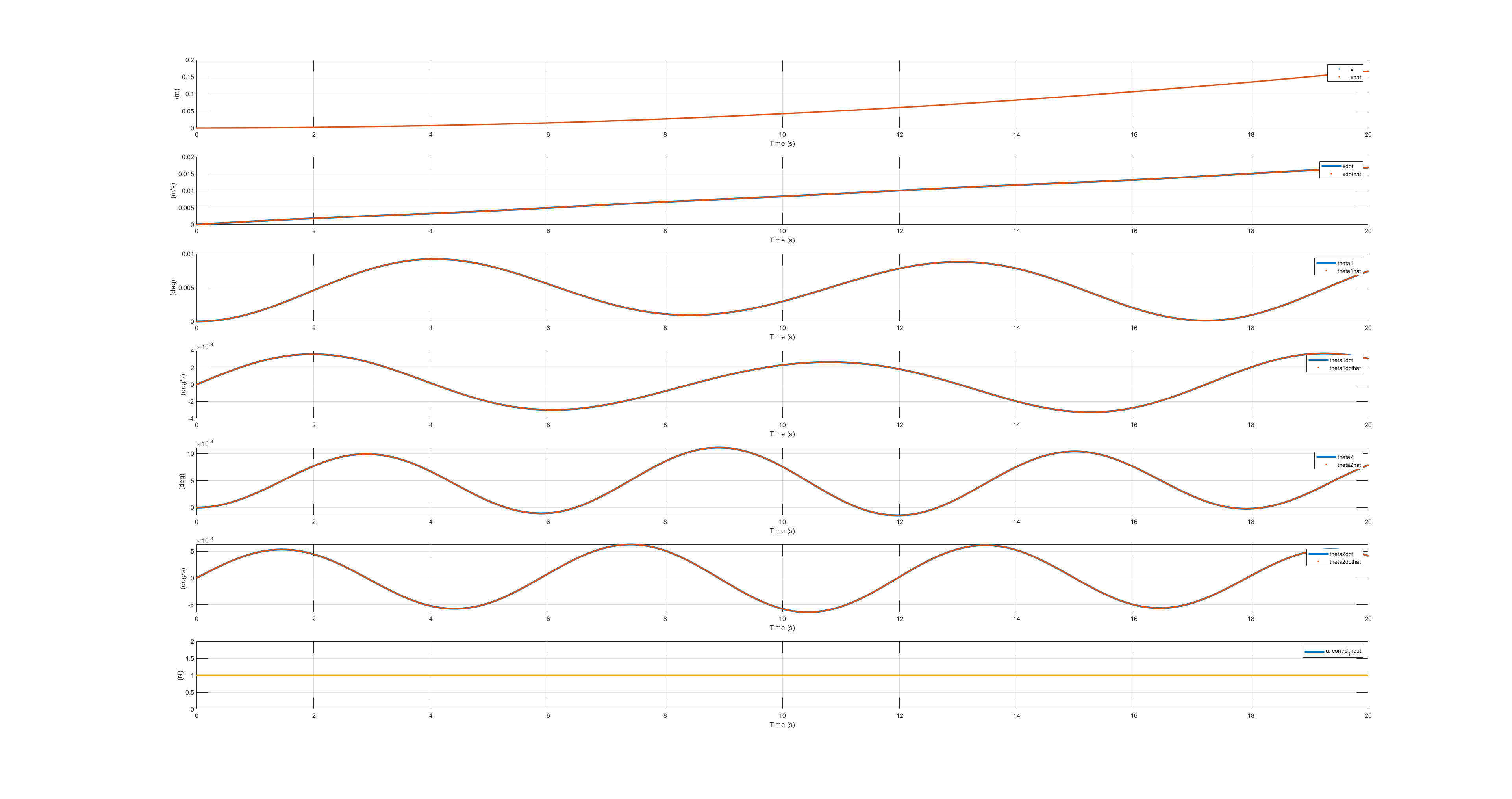


Figure : Nonlinear system, output vector x only, 0 initial conditions, unit step input.

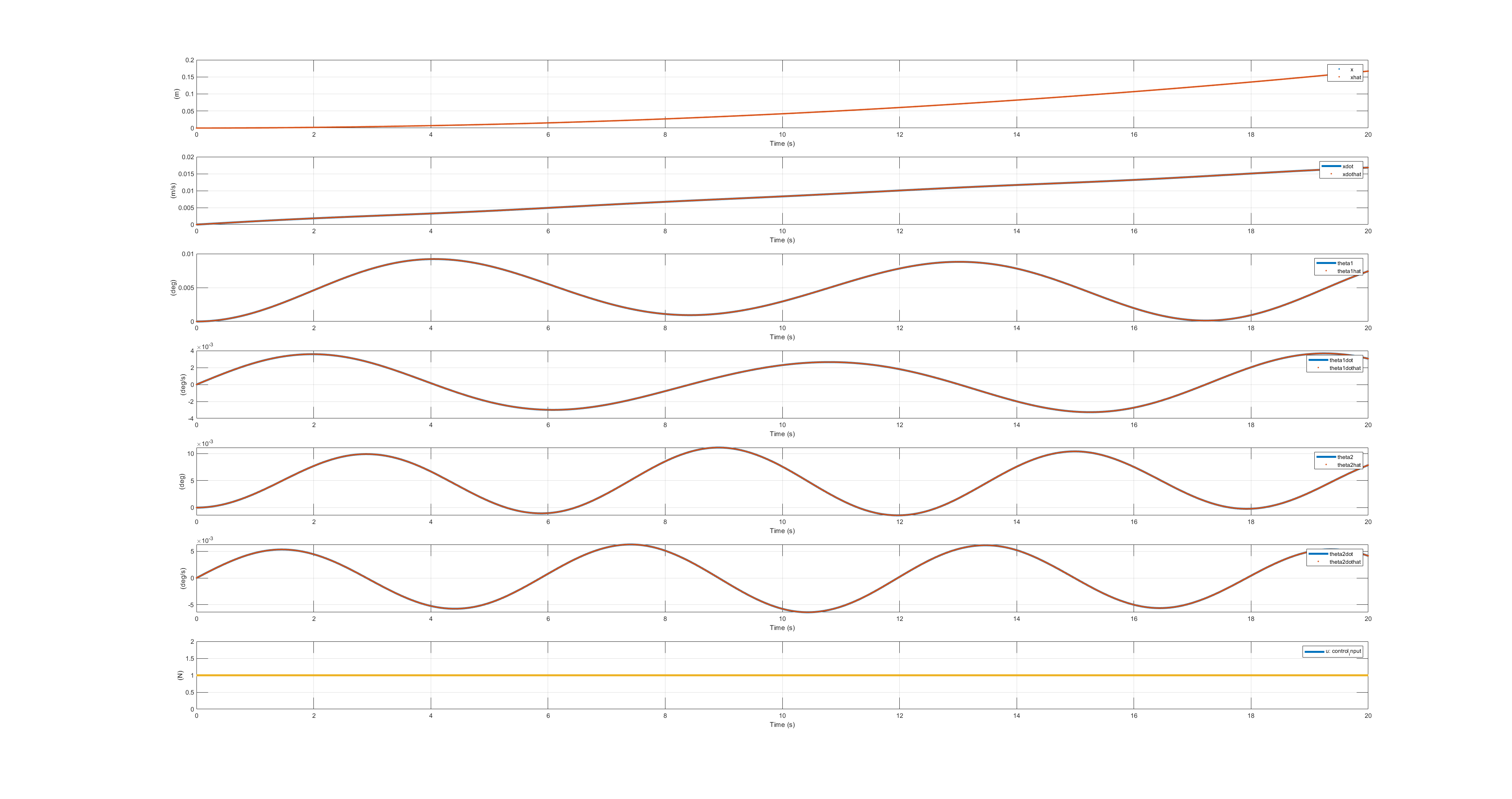


Figure : Nonlinear system, output vector (x(t), θ2(t)), 0 initial conditions, unit step input.

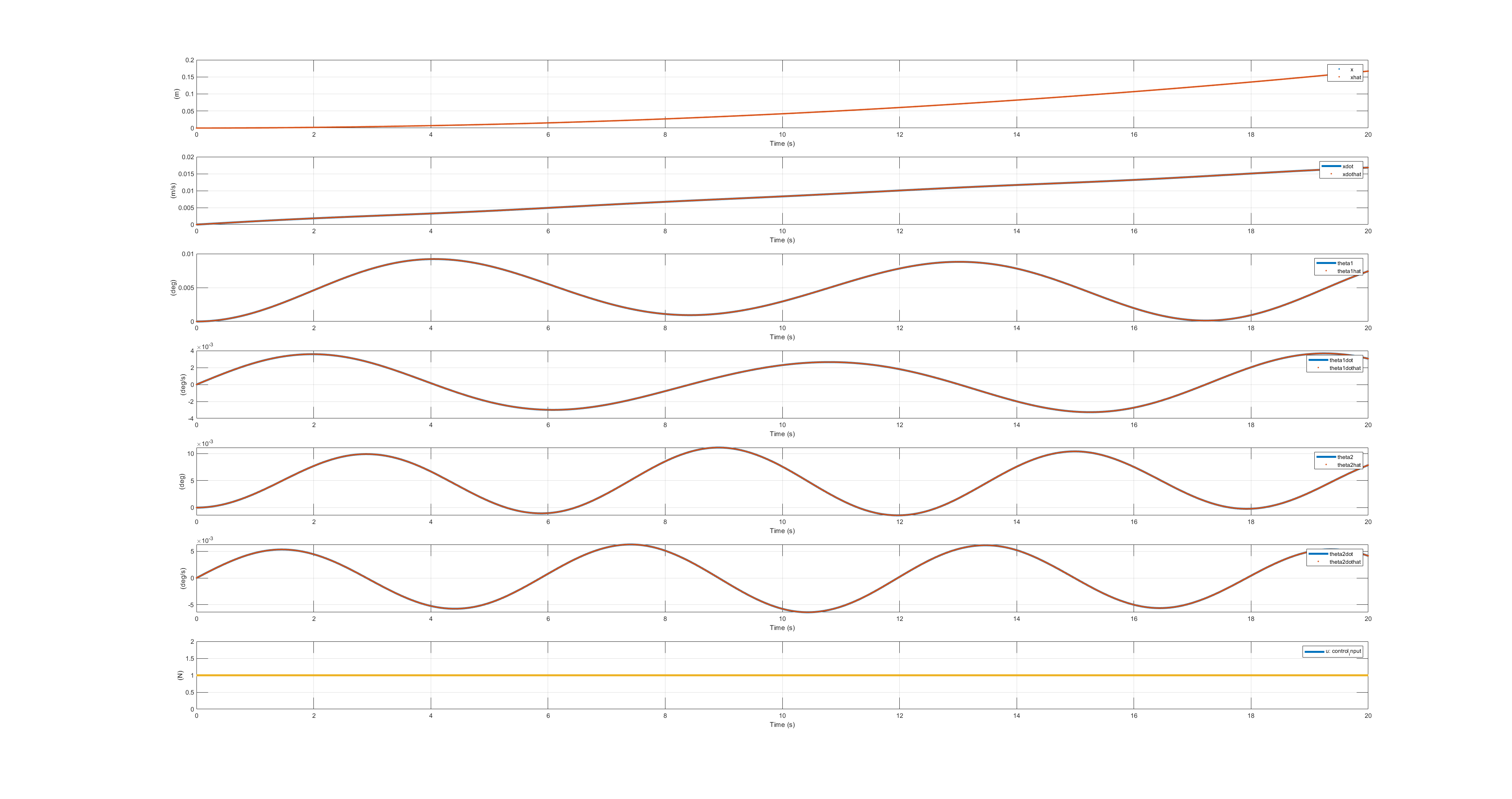


Figure : Nonlinear system, output vector (x(t), θ1(t), θ2(t)), 0 initial conditions, unit step input.

Finally let’s take a look at the nonlinear system with nonzero initial conditions:

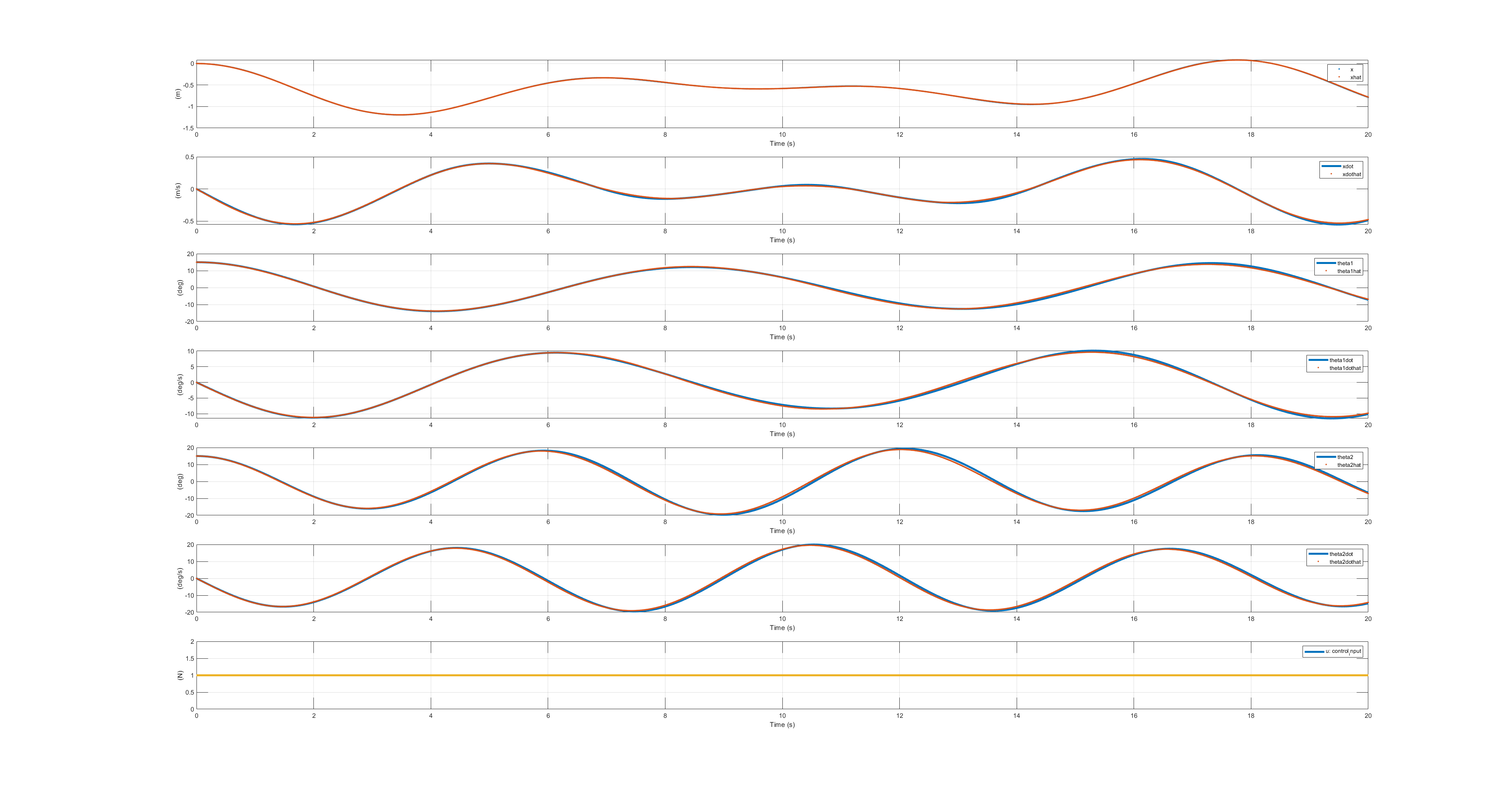


Figure : More challenging: Nonlinear system, output vector x only, 15 degree initial conditions on both pendulums, unit step input.

We see that even with the nonlinear system and 15 degrees initial condition, and measuring only the cart position, the state estimator still performs quite well.

# Use the LQG Method and apply resulting Output Feedback Controller to Original Non-Linear System

In this last part of the project, we are tasked to design an output feedback controller using the LQG method for the “smallest” output vector, which we chose to be *x(t)*. The LQG method basically consists of combining results from LQR controller and Kalman-Bucy filter. These gains will be applied to the original non-linear system and its performance will be illustrated in simulation. This simulation will differ from part **F** because now we are assuming some noise injected into the system, and its response will not be to a unit step input, but rather full state feedback with LQR selected gains.

floating:  
*UD(t)* and *V(t)* are the process noise and measurement noise, respectively. Here, *UD(t)* and *V(t)* are independent zero mean white Gaussian processes with covariances 𝛴D and 𝛴V, respectively.