CARLETON UNIVERSITY
Department of Systems and Computer Engineering

SYSC 3600
Lab #3: Control of an Inverted Pendulum

Before coming to Lab #3:

- Read and understand the below pre-lab for Lab #3.
- Read the "SIMULINK Extras" in Appendix A.
- Familiarize yourself with the SIMULINK models given in Appendix B.

Instructions:
Read through and complete the entire lab. Your report should address the points made in the **Report** suggestion boxes and demonstrate your understanding of the material.

1 Purpose

The purpose of this lab is:

(a) To look at the inverted pendulum problem, which is an unstable, non-linear system.

(b) To study a proportional-plus-derivative (PD) controller design using SIMULINK.

(c) To study the validity of a *linear approximation* for a *non-linear system*.

2 Pre-Lab

2.1 Introduction

A useful design problem in control theory is known as the inverted pendulum problem. The basic problem is to balance a long object on end in a vertical position. One example would be to balance an inverted broom on your hand (where the brain acts as the control system to do the balancing). Another example is the attitude control used to balance a space booster in a vertical position during takeoff (where the space booster is balanced on top of the thrust generated by the booster). The difficulty with an inverted pendulum is that it is naturally an *unstable* and *non-linear* system.

In this lab we will model the inverted pendulum illustrated in Fig. 1 and simulate this system when a proportional-plus-derivative (PD) controller is used to attempt to keep the inverted pendulum balanced[1]. The inverted pendulum is attached to a cart with a bracket that allows the pendulum to swing only to the left or to the right. Similarly, the cart can move only to the left or to the right. The object is to design a control system that keeps the inverted pendulum in a vertical position by applying a force f to the cart under the pendulum. Note that for the inverted pendulum we are assuming no friction for the cart or the pivot of the pendulum to simplify the problem. Also, we are assuming no extra forces on the pendulum, such as air resistance or wind, with the exception of gravity itself. Small external forces, such as air resistance or small amounts of wind, would ultimately be compensated for by the PD controller which is trying to keep the inverted pendulum vertical.

[1]Design problem taken from Ogata, *System Dynamics*, Prentice Hall, 1978, pp. 531–536.

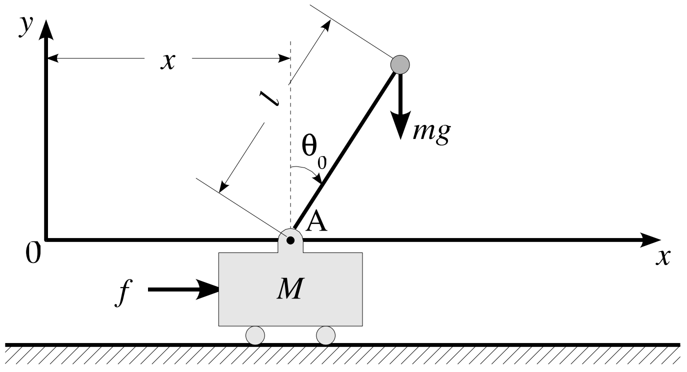

Figure 1: Inverted pendulum system.

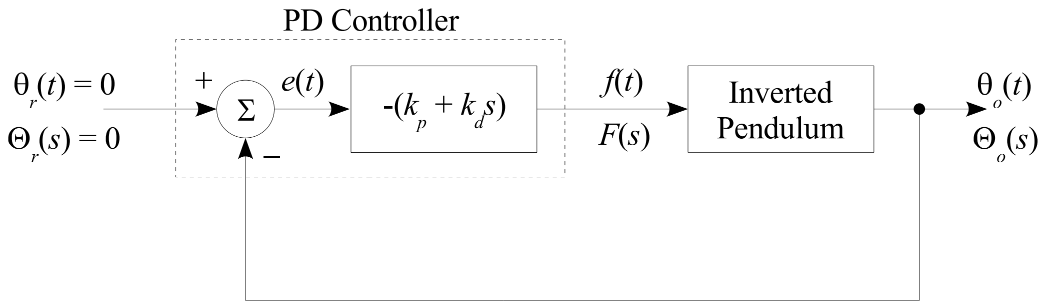

Figure 2: Block diagram of inverted pendulum system controlled with a PD controller.

For the purpose of this lab, we will take that $M = 1000$ kg, $m = 200$ kg, and $\ell = 10$ m. We wish to design a suitable controller that generates a force f such that the overall system has a damping factor of $\zeta = 0.7$ and an undamped natural frequency of $\omega_n = 0.5$ rad/sec. Choosing a PD controller for this problem, the overall system with the controller has a block diagram as shown in Fig. 2.

2.2 PD controller dynamics

In Fig. 2, the input $\theta_r(t)$ serves as a reference angle which we will set to $\theta_r(t) = 0$ (*i.e.*, we want to bring the pendulum to perfectly vertical). The job of the PD controller is to take the error in the actual angle of the inverted pendulum compared to the reference angle, which we write as

$$e(t) = \phi_r(t) - \phi_o(t) = 0 - \phi_o(t) = -\phi_o(t) \tag{1}$$

and turn $e(t)$ into a control signal $f(t)$ to reduce the overall error in angle. In this case, this control signal is the force $f(t)$ needed to push the cart to try to balance the inverted pendulum.

A PD controller has a transfer function of

$$H(s) = k_p + k_d s = k_p\,(1 + t_d s) \tag{2}$$

where k_p is the proportional constant, k_d is the derivative constant of the proportional-plus-derivative controller, and $t_d = k_d/k_p$ is the derivative time constant. Since $e(t) = -\phi_o(t)$, we will include an extra negative sign in the transfer function for convenience to give a PD controller transfer function of

$$H_{PD}(s) = -(k_p + k_d s). \tag{3}$$

Using the transfer function $H_{PD}(s)$, we can write the Laplace transform $F(s)$ of the force $f(t)$ as

$$F(s) = H_{PD}(s)[\Theta_r(s) - \Theta_o(s)] = -(k_p + k_d s)[\Theta_r(s) - \Theta_o(s)] \tag{4}$$

which if $\Theta_r(s) = 0$, then

$$F(s) = k_p \Theta_o(s) + k_d s \Theta_o(s) \tag{5}$$

Inverting the Laplace transform in this equation gives

$$f(t) = k_p \theta_o(t) + k_d \dot{\theta}_o(t). \tag{6}$$

Therefore, Eq. 6 is the force that the PD controller applies to the cart to attempt to keep the inverted pendulum vertical. We now have to work out the dynamics of the card/pendulum to finish the analysis of the system.

2.3 Dynamics of an Inverted Pendulum

Using the xy co-ordinate system shown in Fig. 1, we can determine the dynamics of the motion for the inverted pendulum and the cart. Applying Newton's second law of motion ($\sum ma = \sum f$) to the system in the x-direction of the motion we obtain

$$M\ddot{x} + m\frac{d^2}{dt^2}\left[x + \ell \sin\theta_o\right] = f. \tag{7}$$

This equation includes the inertia of mass M at position $x(t)$, the x-direction inertia of mass m at position $x(t) + \ell \sin\theta_o(t)$, and the force $f(t)$ from the controller. Performing the differentiation of the second term and simplifying Eq. 7 results in

$$(M + m)\ddot{x} - m\ell(\dot{\theta}_o)^2 \sin\theta_o + m\ell\ddot{\theta}_o \cos\theta_o = f. \tag{8}$$

Ultimately, we wish an equation in terms of only $f(t)$ and $\theta_o(t)$. To remove $x(t)$ in Eq. 8, we first need a second equation in terms of $x(t)$, $f(t)$, and $\theta_o(t)$. If we consider the rotational motion of the pendulum about point A (the pivot point of pendulum) we obtain

$$A_y \ell \sin\theta_o - A_x \ell \cos\theta_o = J\ddot{\theta}_o = 0 \tag{9}$$

where A_x and A_y are the forces at point A in the x- and y-directions, respectively, and J is the moment of inertia of the pendulum around its centre of gravity. Since the shaft of the pendulum is assumed to be massless, then the centre of gravity for the pendulum is concentrated at the centre of the mass m. For this case, the moment of inertia about the centre of gravity is small so we assume $J = 0$ (*i.e.*, we assume that mass m is a point mass with no diameter so there is no moment of inertia J).

Plugging into Eq. 9 the directional forces A_x and A_y that occur due to the rotation of mass m around point A gives

$$-\left[mg + m\frac{d^2}{dt^2}(\ell\cos\theta_o)\right]\ell\sin\theta_o + \left[m\frac{d^2}{dt^2}(x + \ell\sin\theta_o)\right]\ell\cos\theta_o = 0 \tag{10}$$

which after performing the differentiation simplifies to

$$\ddot{x}\cos\theta_o + \ell\ddot{\theta}_o = g\sin\theta_o. \tag{11}$$

Finally, we can eliminate $\ddot{x}$ by combining Eq. 8 and Eq. 11 to give

$$(M + m)(g\sin\theta_o - \ell\ddot{\theta}_o) - m\ell(\sin\theta_o)(\cos\theta_o)(\dot{\theta}_o)^2 + m\ell(\cos^2\theta_o)\ddot{\theta}_o = f\cos\theta_o \tag{12}$$

which is a 2^{nd}-order *non-linear* differential equation that relates the inverted pendulum's angle $\theta_o(t)$ and the input force $f(t)$ from the PD controller. When simulating this non-linear system, it is convenient to solve Eq. 12 for $\ddot{\theta}_o$ as follows

$$\ddot{\theta}_o = \frac{f \cos\theta_o - (M+m)g\sin\theta_o + m\ell(\sin\theta_o)(\cos\theta_o)(\dot{\theta}_o)^2}{m\ell(\cos^2\theta_o) - (M+m)\ell}. \tag{13}$$

The SIMULINK subsystem in Fig. 7 realizes Eq. 13. Double check that you understand how the subsystem implements this equation.

2.4 Motion of the cart

Not only can we determine how the angle θ_o of the inverted pendulum changes as given in Eq. 13, but we can also determine how the cart itself will move from the different forces in action. Notice that the lateral motion of the cart x is described by Eq. 8. Solving for $\ddot{x}$ in Eq. 8 gives

$$\ddot{x} = \frac{f + m\ell(\dot{\theta}_o)^2\sin\theta_o - m\ell\ddot{\theta}_o\cos\theta_o}{M+m}. \tag{14}$$

This equation is incorporated into Fig. 10 and Fig. 9 to calculate the position of the cart x along with the angle θ_o of the inverted pendulum.

2.5 PD Controller Design

2.5.1 Overall PD controller/inverted pendulum system

Recall the simulation diagram given in Fig. 2. Now that we have dynamics for the cart/inverted pendulum as given in Eq. 13, we can implement a simulation diagram that will include the PD controller as given in Fig. 2. A SIMULINK simulation model of this realization for the overall PD controller/inverted pendulum system is given in Fig. 8.

2.5.2 Choosing controller values k_p and k_d.

One part missing to get the whole PD controlled inverted pendulum working is to select appropriate values for k_p and k_d. This is not a straightforward task since the inverted pendulum is a *non-linear* system.

We can get a ballpark estimate of the desired values for k_p and k_d by approximating the non-linear dynamics of the overall PD controlled inverted pendulum with a linear system. Let's start by obtaining a single equation that describes the overall PD controlled inverted pendulum. An overall equation of the dynamics is obtained by substituting the force $f(t)$ from Eq. 6 into Eq. 13 to give

$$\ddot{\theta}_o = \frac{k_p\theta_o\cos\theta_o + k_d\dot{\theta}_o\cos\theta_o + m\ell(\dot{\theta}_o)^2\cos\theta_o\sin\theta_o - (M+m)g\sin\theta_o}{m\ell\cos^2\theta_o - (M+m)\ell}. \tag{15}$$

This equation describes all of the dynamics we have defined for the PD controlled inverted pendulum.

With the PD controlled inverted pendulum now described in Eq. 15, the task now is to design the PD controller by appropriately choosing values for k_p and k_d. As previously indicated, we ultimately want to design this system to have a damping factor of $\zeta = 0.7$ and an undamped natural frequency of $\omega_n = 0.5$ rad/sec.

In order to make the problem analytically tractable (and use what we have learned about linear time-invariant systems), Eq. 15 can be approximated with a linear system. We can assume that θ_o and $\dot{\theta}_o$ are small, which is reasonable because the purpose of the controller is to keep the pendulum vertical (or at least reasonably close to vertical). With small values for θ_o and $\dot{\theta}_o$, we can form the following approximations

$$\sin\theta_o \approx \theta_o \tag{16}$$

$$\cos\theta_o \approx 1 \tag{17}$$

$$(\dot{\theta}_o)^2 \approx 0. \tag{18}$$

Using the approximations given in Eq. 16 to Eq. 18, we can simplify Eq. 15 to

$$-M\ell\ddot{\theta}_o = k_p\theta_o + k_d\dot{\theta}_o - (M+m)g\theta_o \tag{19}$$

which putting Eq. 19 into standard 2^{nd}-order linear differential equation form gives

$$\ddot{\theta}_o + \frac{k_d}{M\ell}\dot{\theta}_o + \left[\frac{k_p - (M+m)g}{M\ell}\right]\theta_o = 0. \tag{20}$$

Notice that the right hand side of Eq. 20 is zero which suggests that there is no input to the system. Recall that the PD controller block diagram given in Fig. 2 has a reference angle input $\theta_r(t)$ which we have set to zero. If a non-zero angle is desired for $\theta_r(t)$, then the formation of Eq. 20 would have to include a non-zero $\theta_r(t)$ when Eq. 6 is formed from Eq. 4.

Even in the form given in Eq. 20, we can still determine the transfer function for the system to be in the form

$$H(s) = \frac{\Theta_o(s)}{\Theta_r(s)} = \frac{B}{s^2 + \frac{k_d}{M\ell}s + \left[\frac{k_p - (M+m)g}{M\ell}\right]} \tag{21}$$

where B is unknown since $\Theta_r(s) = 0$. B does not have to be known in this case since B is associated with the input which is just zero (*i.e.*, $\theta_r(t) = 0$).

3 Preparation

Using the linearized approximate transfer function given in Eq. 21, write equations to calculate k_p and k_d in terms of the damping ratio ζ and the undamped natural frequency ω_n. You might want to first first the equations for ω_n and ζ, and afterwards solve for k_p and k_d.

$k_p =$

$k_d =$

(22)

Now, given that $M = 1000$ kg, $m = 200$ kg, $\ell = 10$ m, $\omega_n = 0.5$ rad/sec, $\zeta = 0.7$, and $g = 9.81$ m/s^2, determine the initial estimates for k_p and k_d. These values for k_p and k_d should work reasonably well in the nonlinear system.

Report: Include the equations for k_p and k_d in your report along with their values.

4 Lab

4.1 Inverted pendulum demo in SIMULINK

SIMULINK comes with an inverted pendulum demo which can be started by typing

```
penddemo
```

in MATLAB's `Command Window`. Start up the inverted pendulum demo.

The control system for the pendulum demo is more complicated (and more complete) than the one we will be using, but it does serve to reinforce what we are trying to accomplish. After starting the simulation of the inverted pendulum, an animation of the cart/pendulum is displayed and the control system moves the cart to balance the inverted pendulum. You can move the slider bar at the bottom of the animation window to "knock" the pendulum in that direction, causing the cart to react and re-balance the inverted pendulum.

Try knocking the pendulum hard such that the cart moves quickly to re-balance the inverted pendulum. You should observe that the cart does overshoot slightly when it is balancing the inverted pendulum, which is indicative that the system is underdamped.

Report: In your report, discuss why an overall underdamped response in balancing the inverted pendulum is preferred over a critically damped or an overdamped response.

Hint: What would happen to the position of the cart if $\theta_o(t)$ was always positive (but approaching 0) instead of $\theta_o(t)$ alternating from positive to negative.

4.2 Testing the pendulum

Start by implementing and testing the pendulum on its own as shown in the SIMULINK subsystem given in Fig. 10 and used as shown in Fig. 3. Notice that the model in Fig. 3 performs an *open-loop* test of the cart/inverted pendulum. Also, notice that the model in Fig. 3 is effectively testing the zero-input response of the inverted pendulum since $f(t) = 0$.

Run the simulation for $M = 1000$ kg, $m = 200$ kg, $\ell = 10$ m, and $g = 9.81$ m/s^2. Set these variables by entering the following commands into the MATLAB `Command Window`.

```
M=1000;
m=200;
l=10;
g=9.81;
```

Intuitively, you would expect that the pendulum will just drop. What do you observe for $\theta_o(t)$ and $x(t)$ in the `Scope` plots? To help visualize what is happening with the pendulum, the following simple MATLAB function will give a quick animation of the pendulum's motion. Type `edit` in MATLAB `Command Window`, then type in the following MATLAB function and save it as `drawpendulum.m`.

```
function drawpendulum(time,theta,x)

% Check if x was passed or not
```

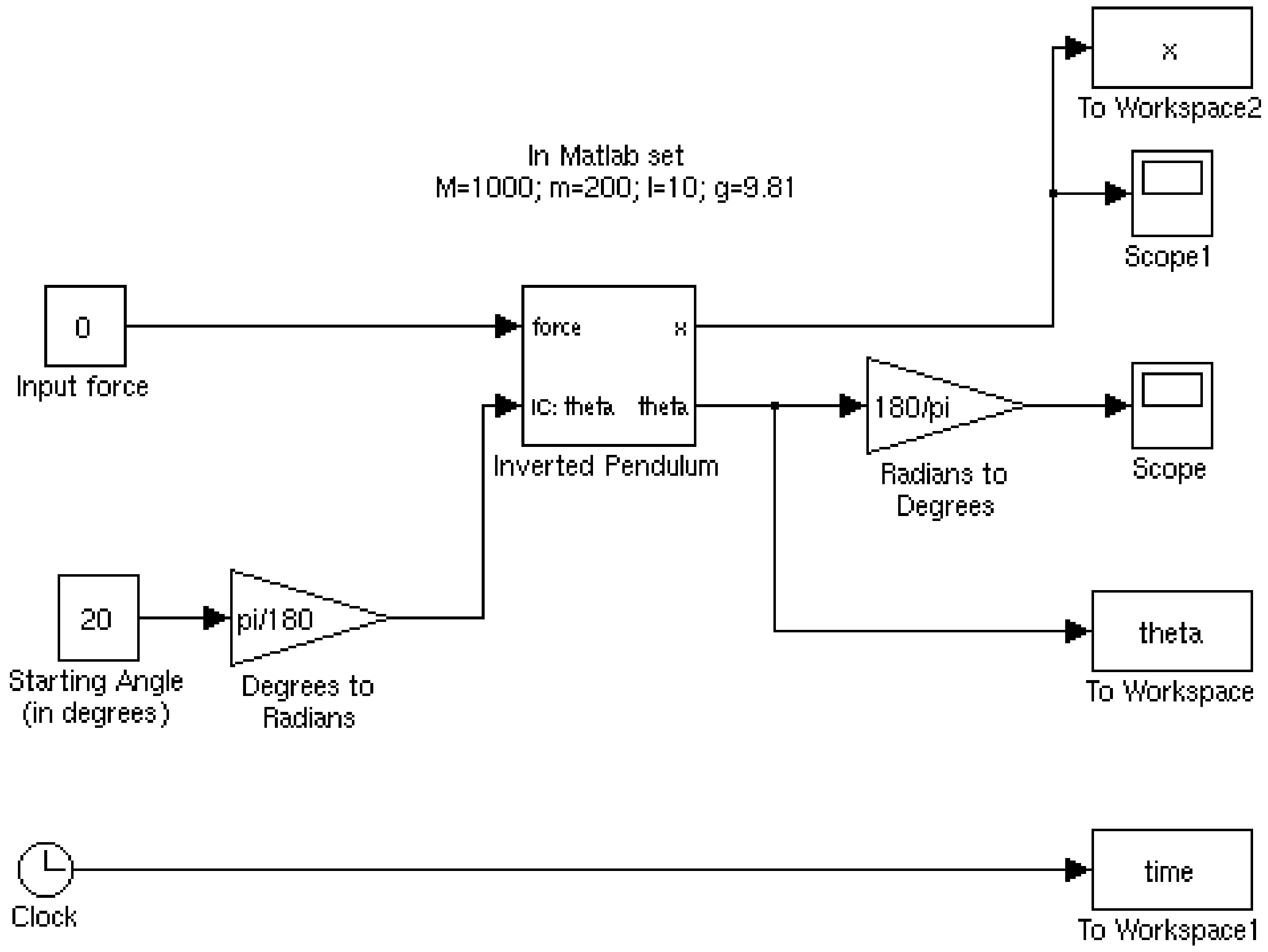

Figure 3: SIMULINK model to simulate the zero-input response of the inverted pendulum and the cart.

```
if nargin==2
   % If x was not passed, just set x to a vector of zeros
   x = zeros(1,length(theta));
end

% Calculate the time different between subsequent time samples
timediff=[diff(time)];

% Limit the refresh to about 10 per second
skip=ceil(0.1*length(time)/(time(end)-time(1)));

figure(1); clf; hold on;
axis([min(x)-2, max(x)+2, -1.1, 1.1]);
for i=1:skip:length(theta)-1
   h1=plot(x(i)+sin(theta(i)), cos(theta(i)), 'o');
   h2=line(x(i)+[0 sin(theta(i))], [0 cos(theta(i))]);
   pause(timediff(i));
   drawnow;
   delete(h1); delete(h2);
end
```

Notice now that `time`, `theta`, and `x` are variables passed to MATLAB in Fig. 3 using the `To Workspace` blocks. After a simulation run, these variables are available to pass to the `drawpendulum()` function. Before running a simulation, be sure that you have set the `To Workspace` blocks to output an `Array`.

After you have run a simulation, you can call the `drawpendulum()` function in two ways. To display the movement of the pendulum by itself, type the following command in the MATLAB `Command Window`.

```
drawpendulum(time, theta);
```

The second method of calling `drawpendulum()` will also move the pivot point of the pendulum to indicate the x-position of the cart and is performed using the following command.

```
drawpendulum(time, theta, x);
```

You should compare the `Scope` block's plot of $\theta_o(t)$ and compare it to what you observe when you run `drawpendulum(time, theta)`.

Warning: You will likely see some unexpected results (like a ghost has given the pendulum an extra kick). SIMULINK is trying to simulate this 2^{nd}-order system and if the solver isn't configured correctly, the pendulum may sometimes do a full circle unexpectedly. For instance, on my computer when I leave the solver at `ode45` and the `Max step size` as `auto`, then if I simulate the system for a long period of time, such as for 1000 seconds, then the pendulum starts to do full circles. These results occur because SIMULINK is using a variable step size and might choose a step size that is too large at some point. To remove this problem, set the `Max step size` to something small, such as `0.1`.

This is a lesson that you shouldn't always trust simulation results without careful setup of the simulation and interpretation of the results.

One possible side effect of making `Max step size` small is that your `Scope` block may have too many points to process. By default, the `Scope` block keeps track of only the last 5000 sample points. If your `Scope` output is cutoff, click on the properties icon in the `Scope` output window, and then under the `Data history` tab unselect the `Limit data points to last` option so that all simulation data points are retained.

Report: Describe the zero-input response of the pendulum as well as what happens to the position of the cart. How is this affected by the starting angle of the inverted pendulum?

4.3 Simulation of PD controlled non-linear inverted pendulum

Now that the inverted pendulum model has been tested in open-loop, you can move on to *closed-loop* control of the inverted pendulum with the PD controller. First we must address an important issue with a PD controller.

4.3.1 Problem of derivative with the PD controller

Unfortunately, the PD controller given in Eq. 3 should not be realized directly. The s term from the "D" portion of the PD controller requires that the derivative of the error signal be taken. If there is any noise disturbance introduced into the error signal, then taking the derivative will magnify any high frequency noise.

To alleviate magnifying high frequency noise, you can design a lowpass filter and place it before the PD controller to attenuate high frequency components of the error signal. A very simple lowpass filter can be designed with the following transfer function

$$H_{LP}(s) = \frac{1}{\dfrac{s}{\omega_{cf}} + 1} \tag{23}$$

where ω_{cf} is the -3 dB cutoff frequency for the lowpass filter (*i.e.*, the 3 dB bandwidth of this lowpass filter is ω_{cf}). If this form has not yet been discussed in class, it will be covered near the end of the course when Bode plots for 1^{st}-order factors is discussed. What Eq. 23 effectively gives is a lowpass filter with roughly a unity gain (*i.e.*, 0 dB) within the passband from $\omega = 0$ to $\omega = \omega_{cf}$, and a -20 dB/decade attenuation after the cutoff frequency ω_{cf}.

Design a lowpass filter with cutoff frequency of $\omega_{cf} = 100$ rads/s for use with your PD controller.

With the Control System Toolbox in MATLAB, you can plot the Bode plots (*i.e.*, the frequency response) for the designed filter using the `sys=tf(num, den)` and `bode(sys)` functions. As an example of how to plot Bode plots, consider the following example transfer function.

$$H_{example}(s) = \frac{s - 100}{s^2 - 1010s + 10000} \tag{24}$$

From the numerator we would form the vector `[1, -100]` and from the denominator we would form the vector `[1, -1010, 10000]`. Using these vectors, the following MATLAB code will plot the Bode plots (both magnitude and phase) for the example transfer function.

```
num = [1, -100];
den = [1, -1010, 10000];
sys = tf(num, den)
bode(sys, {0.1, 10^5})  % plot between 0.1 and 100,000 rads/s
```

Report: In your report you should include the transfer function for your lowpass filter with cutoff frequency of $\omega_{cf} = 100$ rads/s, along with the Bode plots for the filter. Make any observations you can about the lowpass filter from the Bode plots.

4.3.2 Incorporating lowpass filter with PD controller

With your designed lowpass filter, you can effectively cascade it in series with the PD controller. Since MATLAB and SIMULINK will not like the derivative of the PD controller, you must block reduce the lowpass filter and the PD controller into one transfer function before implementing it. The transfer function to implement will therefore be in the form

$$H_{PDwithLP}(s) = \frac{-(k_p + k_d s)}{\dfrac{s}{\omega_{cf}} + 1}. \tag{25}$$

As a side note, SIMULINK and MATLAB won't let you implement Eq. 3 directly anyways since they require that the order of the polynomial in the denominator be the same or greater than the order of the polynomial in the numerator for a transfer function. This shows that SIMULINK has simulation limitations with the derivative and requires transfer functions more like Eq. 25.

4.3.3 Simulating the PD controlled inverted pendulum

Implement the closed-loop system of Fig. 2 with the lowpass filter/PD controller cascaded transfer function given in Eq. 25. The resulting SIMULINK simulation model is shown in Fig. 9. Be sure to enter the values for k_p and k_d that you computed in Eq. 22 into the MATLAB `Command Window` before running your simulation.

```
kp= <computed value>
kd= <computed value>
```

Once you have simulated the closed-loop system, you can again use the MATLAB function

```
drawpendulum(time, theta);
```

to visualize the dynamics of the inverted pendulum.

Run the simulation for the following starting angles for the inverted pendulum

- $\theta_o(t_0^-) = 5°$
- $\theta_o(t_0^-) = -30°$
- $\theta_o(t_0^-) = 65°$
- $\theta_o(t_0^-) = 75°$

where t_0 is the start time of your simulation. Run each simulation for at least 30 seconds. Make sure you have set the `Max step size` simulation parameter to a small enough value to avoid the problems discussed in Sec. 4.2.

Report: In your report you should include plots of the angle $\theta_o(t)$ of the inverted pendulum. For each plot, describe what is happening in the physical system to the inverted pendulum.

Does the PD controller fail to keep the pendulum inverted in any of the cases? If so, discuss why this might happen.

4.4 Additional questions:

1. What if the reference angle θ_r is set to something other than zero? Will the closed-loop system keep the pendulum at that angle? Try it for a few small angles θ_r before coming to a conclusion.

2. How many initial conditions should exist for Eq. 13 and what is the purpose for each initial condition? Are there any initial conditions missing in the realization shown in Fig. 10? Are any of the initial conditions assumed to be zero in Fig. 10?

A SIMULINK Extras

A.1 `Goto` blocks and `From` blocks

When making complicated SIMULINK models, sometimes there are connections that need to be made that cannot be *neatly* routed. One option of making a SIMULINK model look neater is to use the `Goto` and `From` blocks which are available under the `Signal Routing` section of the SIMULINK Library Browser window.

The `Goto` and `From` blocks act sort of like signal "wormholes" where a signal passed into a `Goto` block will emerge out of the corresponding `From` block(s). An example of using the `Goto` and `From` blocks is shown in Fig. 4. There are three `Goto` blocks in Fig. 4, each with a different tag. The tag for a `Goto`

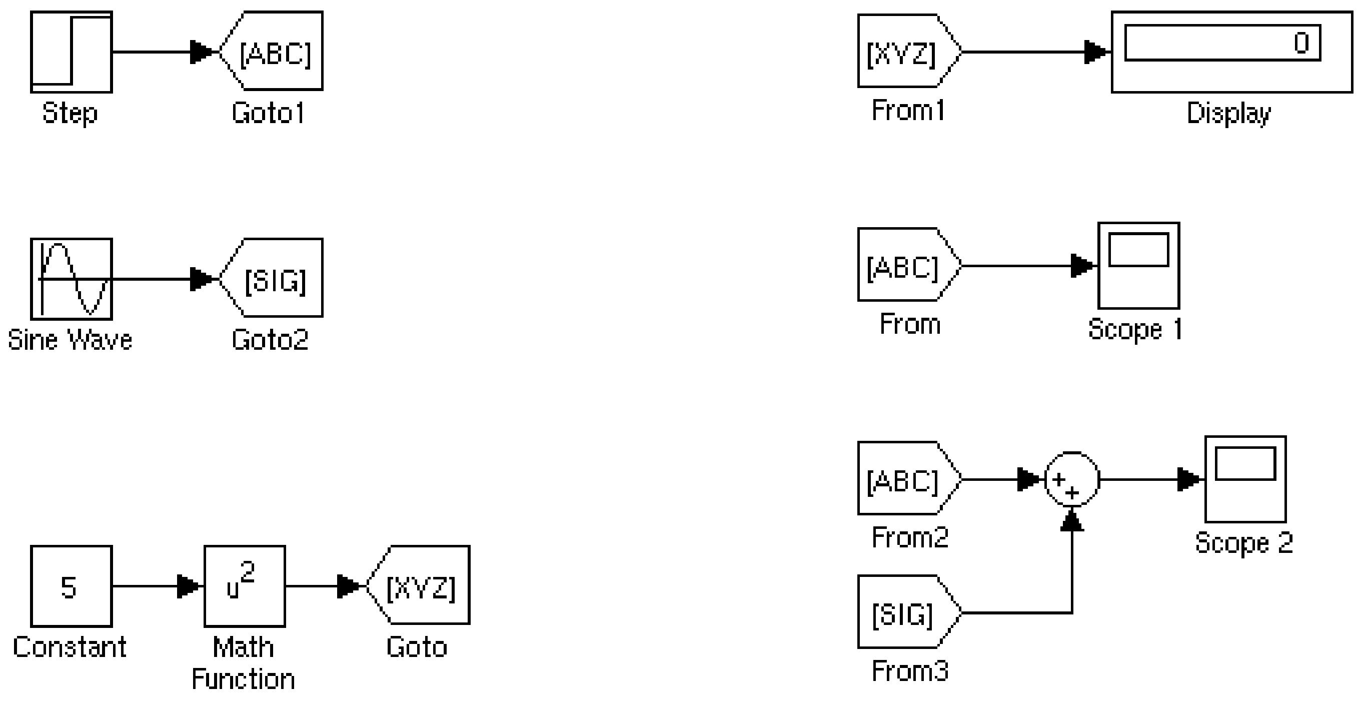

Figure 4: Simple SIMULINK model using `Goto` and `From` blocks.

block can be set by double-clicking on the `Goto` block and changing the tag's name in its block parameters window. For the examples in Fig. 4, there is a step function passed to a `Goto` block with tag `ABC`, a sine wave passed to a `Goto` block with tag `SIG`, and the square of some constant passed to a `Goto` block with tag `XYZ`.

Any `From` block that has the same tag set as a `Goto` block will output a copy of the signal passed into that `Goto` block. For instance, the `Display` block will show the value of the squared constant since it receives the signal through the `From` block with the tag `XYZ`. The `Scope 1` block will plot the step function. Finally, the `Scope 2` block will plot the sum of the step function and the sine wave.

A.2 `Fcn` blocks

In SIMULINK you can generally implement most basic mathematical operations using specialized blocks. These include operations such as add, multiply, sine, cosine, absolute value, etc. Check under the `Math` section of SIMULINK Library Browser for others. Unfortunately, using these basic blocks for complicated equations can make for a complicated looking model.

One option to simplify models is to create a user defined function through the `Fcn` block, found under the `Functions & Tables` section of the SIMULINK Library Browser. The `Fcn` block allows you to quickly implement mathematical expressions in a single block instead of needing many basic mathematical blocks.

For instance, let's say you want to implement the following function.

$$y(t) = x(t)\sin^2(x(t)) \tag{26}$$

This can be done with the `Fcn` block as shown in Fig. 5 using the `Fcn` block expression

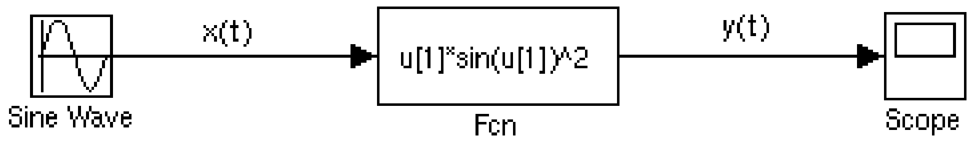

Figure 5: Example use of the `Fcn` block to implement Eq. 26.

```
u[1]*sin(u[1])^2
```

where `u[1]` is the input variable for the `Fcn` block.

You may notice that the variable `u` for the `Fcn` block is a vector the same size as the number of inputs to the `Fcn` block. You can actually multiplex many signals together using the `Mux` block and pass this into the `Fcn` block to perform calculations with multiple signals. For instance, consider that you want to implement the following function that uses $x_1(t)$ and $x_2(t)$ as two separate input signals.

$$y(t) = \sin\left(x_1(t)e^{(2.3-x_2(t))}\right) \tag{27}$$

This equation can be realized as shown in Fig. 6 using the expression

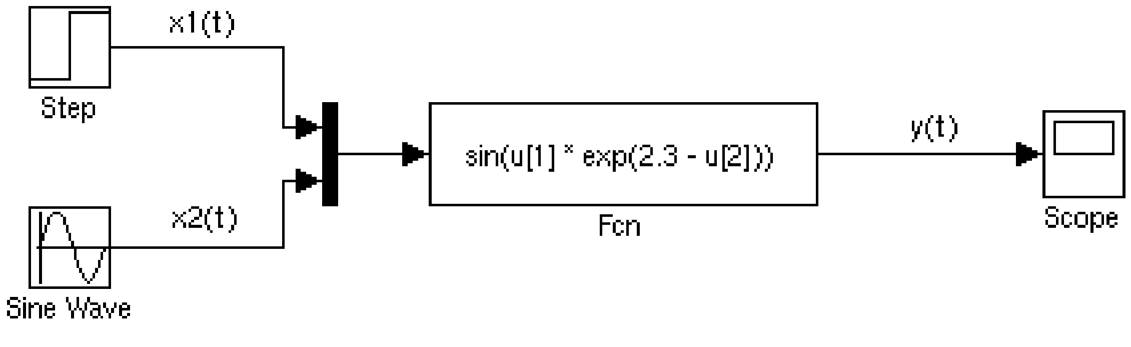

Figure 6: Example use of the `Fcn` block to implement Eq. 27.

```
sin(u[1] * exp(2.3 - u[2]))
```

for the `Fcn` block. Notice that the first input to the `Mux` block is associated with `u[1]`, the second input with `u[2]`, etc. Using the `Mux` block and `Fcn` block in this way can allow for quite complicated expressions using simple mathematics to be implemented.

B Simulink Implementations of the Block Diagrams

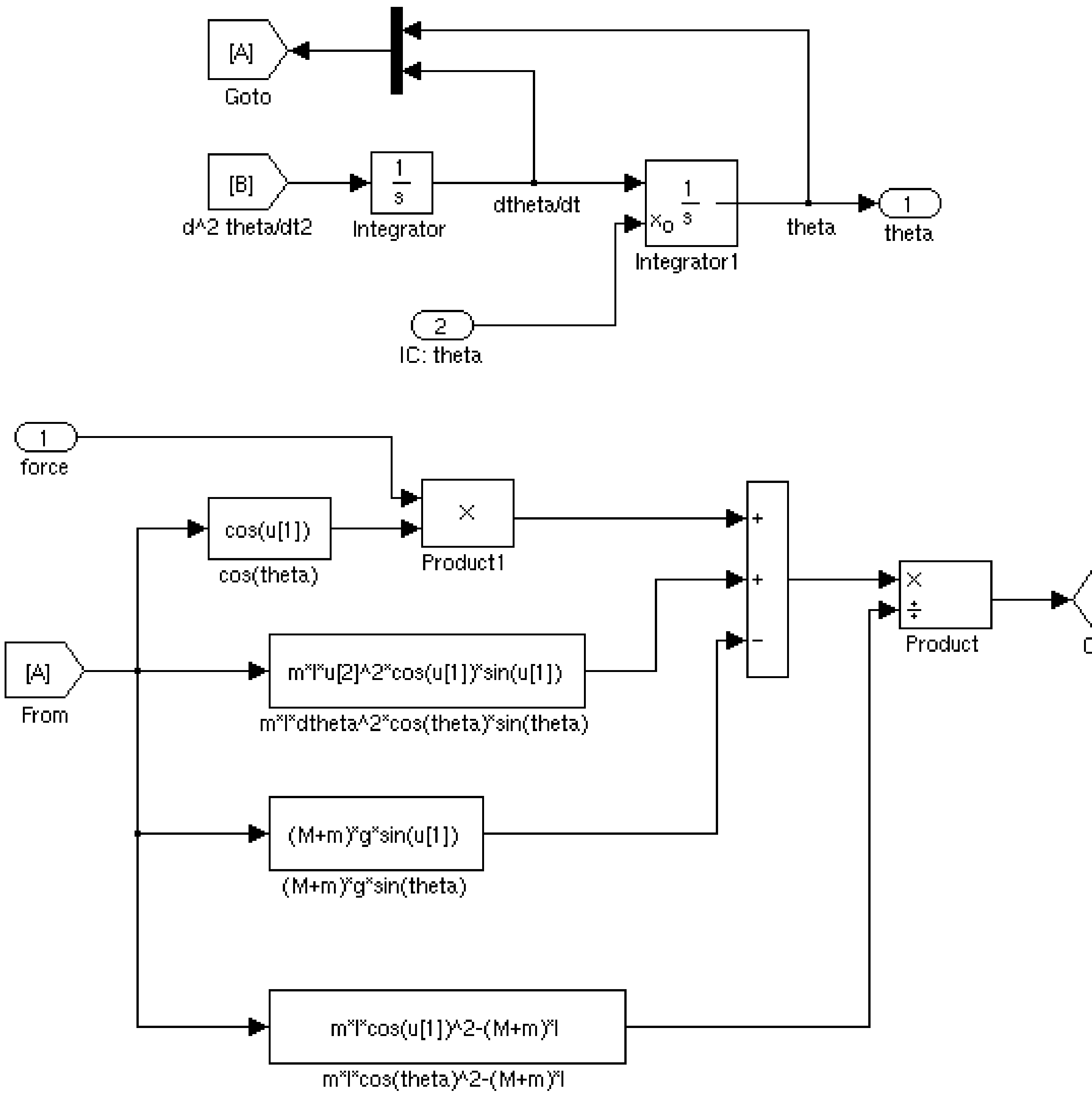

Figure 7: SIMULINK subsystem that implements the non-linear dynamics of angle of pendulum $\theta_o(t)$. This subsystem implements Eq. 13.

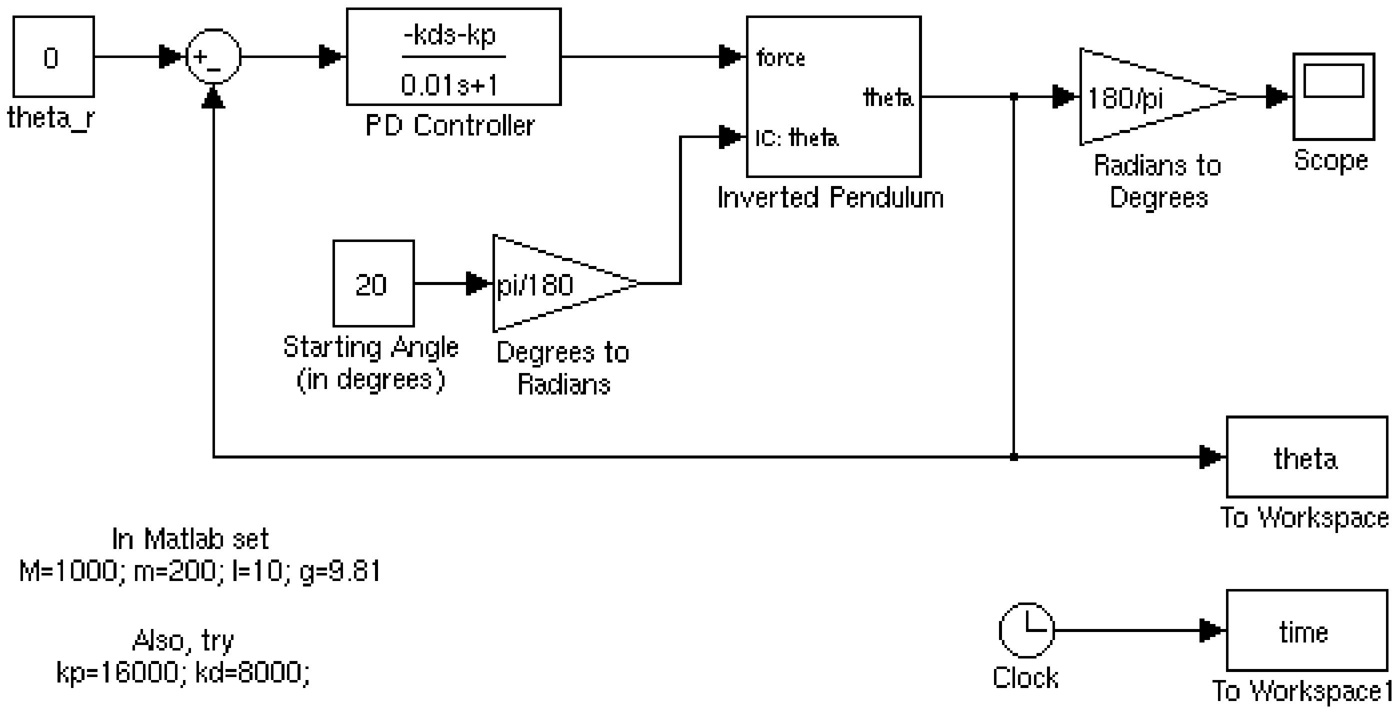

Figure 8: SIMULINK system that implements the PD controller to simulate the cart/pendulum non-linear dynamics.

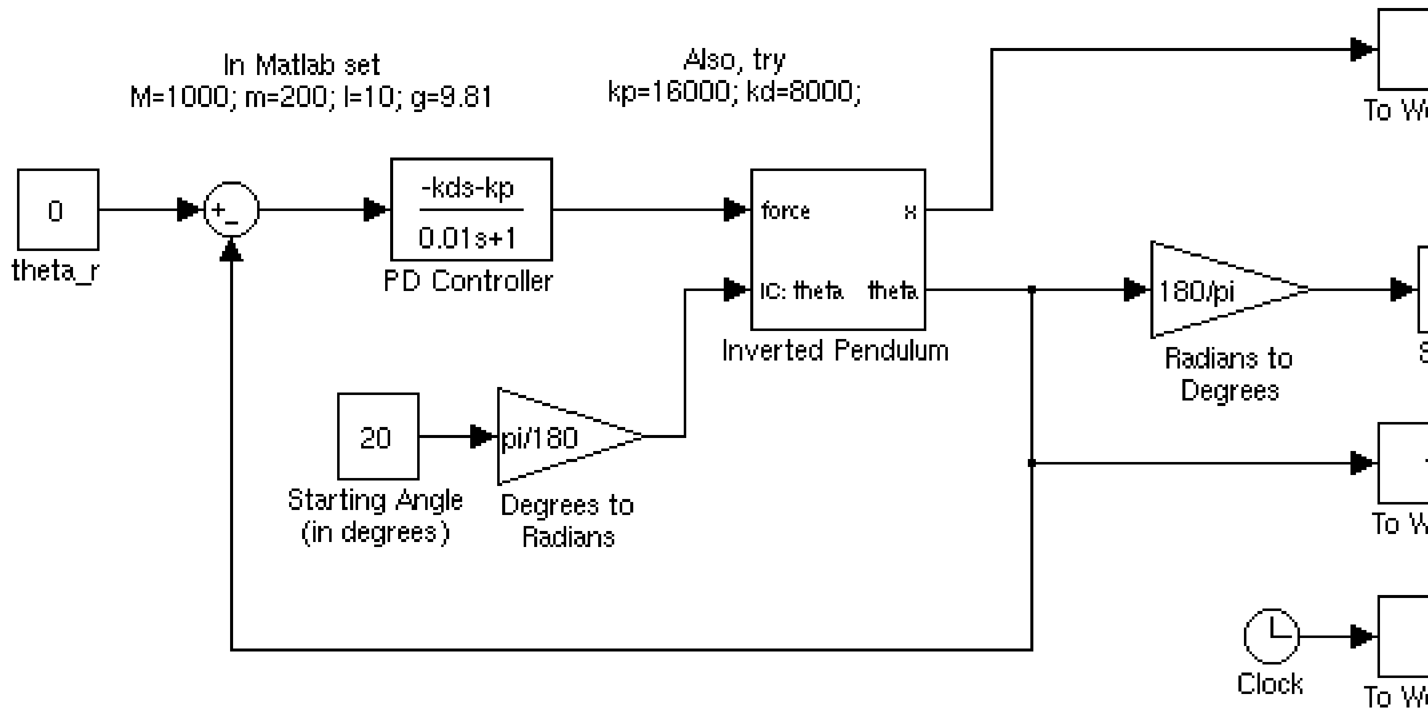

Figure 9: SIMULINK system that implements the PD controller to simulate the cart/pendulum non-linear dynamics with both angle $\theta_o(t)$ of the pendulum and position $x(t)$ of the cart.

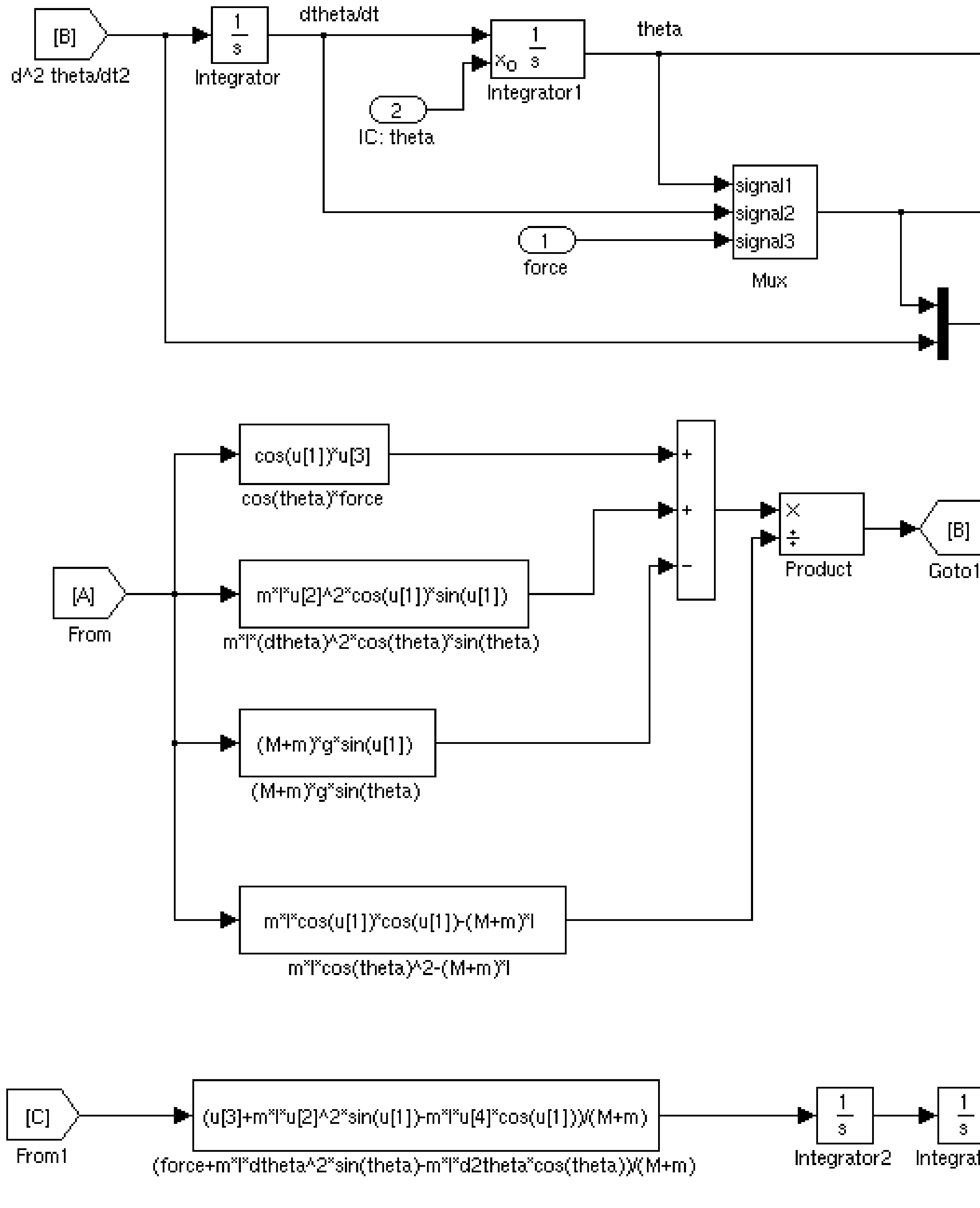

Figure 10: SIMULINK subsystem that implements the non-linear dynamics of angle of pendulum $\theta_o(t)$ and position of cart $x(t)$.