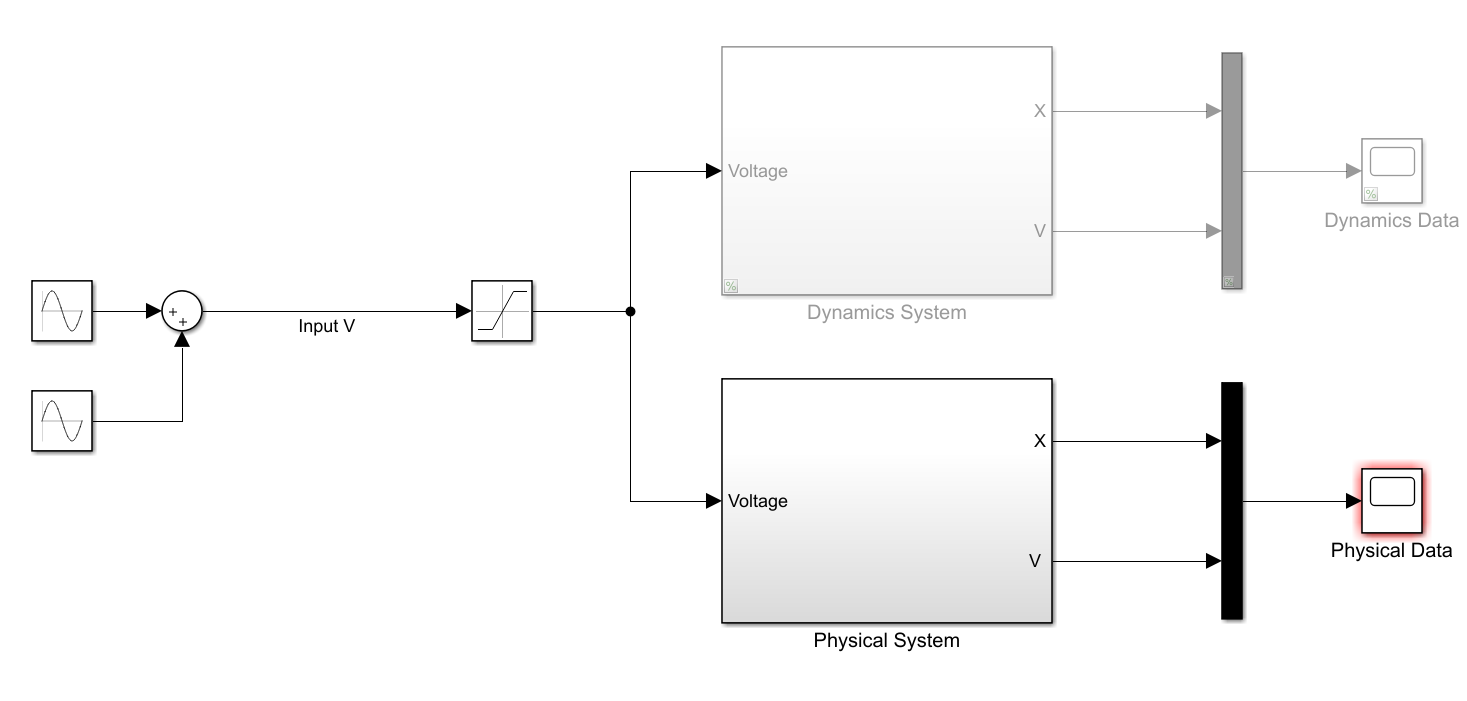
ESE447 Lab 4, System Identification

Chris Bonney, Arnob Dam

**Introduction**

The goal of this lab is to refine the dynamics model we created in the previous lab. We did this by collecting data from the physical robot and using conservation of energy and the Hamiltonian to estimate the physical parameters, shown in Figure 1.

The first step of this is to ensure that both our dynamics model and our Hamiltonian method works. This was done by collecting data from our dynamics model and running the algorithm to estimate the physical parameters and ensuring that we got the physical parameters that we started with, shown commented out in Figure 1. Our goal with the input to the system was to fling the arm around to get a full range of motion in order to better estimate all the parameters.

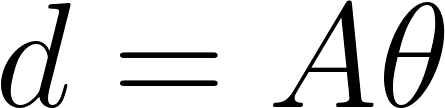


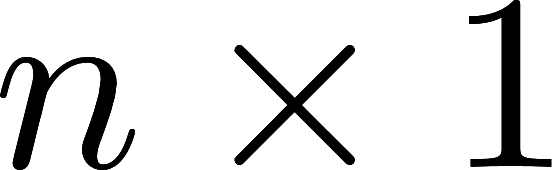
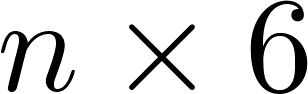
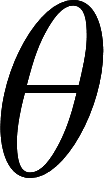
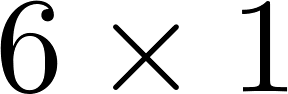
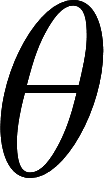
*Figure 1: Collecting Data from the Physical System and the Dynamics Model*

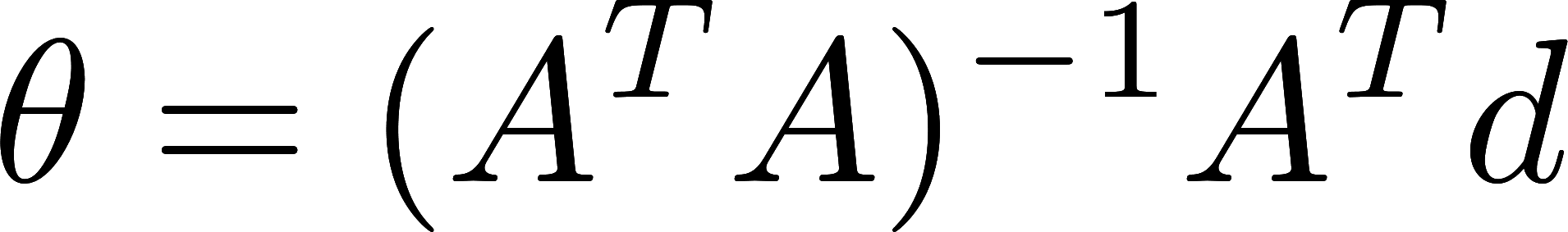
We found a small error in our dynamics model through this and fixed it. After this, we were able to get back the physical parameters we started with, accurate to 4 significant figures, verifying the accuracy of both our model and the algorithm.

**Procedure**

We started by deriving the equations for the Hamiltonian with the purpose of creating an equation that we can then use to solve for the physical parameters:

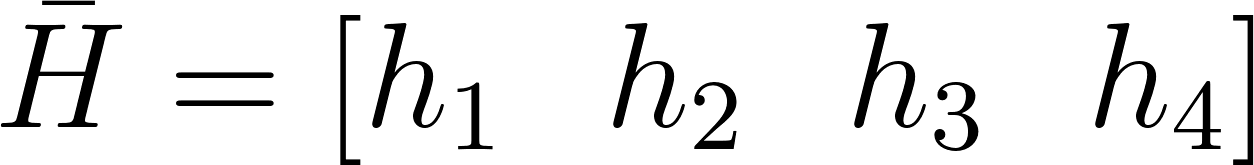
[](https://www.codecogs.com/eqnedit.php?latex=d%20%3D%20A%20%5Ctheta#0)

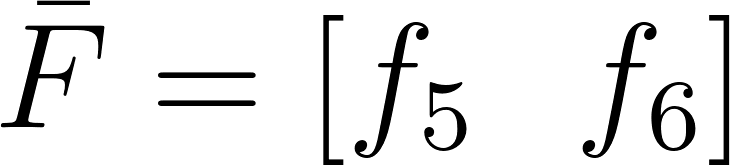
where *d* is a [](http://www.texrendr.com/?eqn=n%20%5Ctimes%201#0) dimensional vector, *A* is a [](https://www.codecogs.com/eqnedit.php?latex=n%20%5Ctimes%206#0) dimensional matrix and the physical parameter vector [](http://www.texrendr.com/?eqn=%5Ctheta#0) is a [](https://www.codecogs.com/eqnedit.php?latex=6%20%5Ctimes%201#0) dimensional vector, where *n* is the number of data points collected. The system is overdetermined and we can solve for [](http://www.texrendr.com/?eqn=%5Ctheta#0) by taking the pseudoinverse of A,

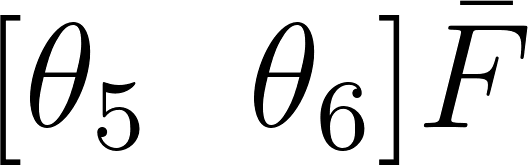
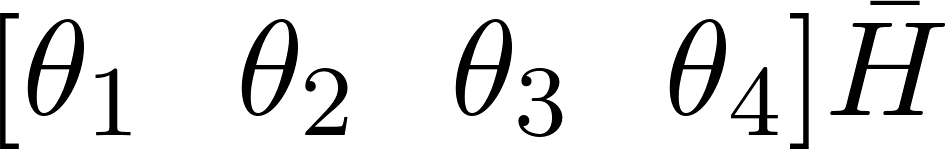
[](http://www.texrendr.com/?eqn=%5Ctheta%20%3D%20(A%5ET%20A)%5E%7B-1%7DA%5ET%20d#0)

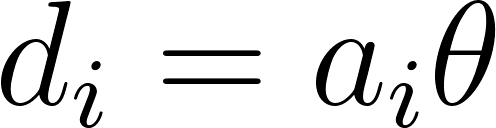
Now, we just have to develop *A* and *d* to make this equation work.

There are 6 physical parameters for our system: four are from the Hamiltonian, and two are from the damping forces on each joint, and they are all independent. Therefore, we should be able to define 6 equations,

[](https://www.codecogs.com/eqnedit.php?latex=%5Cbar%7BH%7D%20%3D%20%5Bh_1%5C%20%5C%20%20h_2%5C%20%5C%20%20h_3%5C%20%5C%20%20h_4%5D%20#0)

[](https://www.codecogs.com/eqnedit.php?latex=%5Cbar%7BF%7D%20%3D%20%5Bf_5%20%5C%20%5C%20f_6%5D#0)

where is the change in energy in the system from *t0* to *ti*, and is the negative of energy lost due to damping and frictional forces in the same time period. Then, we use the fact that the change in energy energy from (*to,ti)*, should be equal to the energy added to the system from the motor plus the energy lost from damping to create an equation for one time step:

[](https://www.codecogs.com/eqnedit.php?latex=d_i%20%3D%20a_i%20%5Ctheta#0)

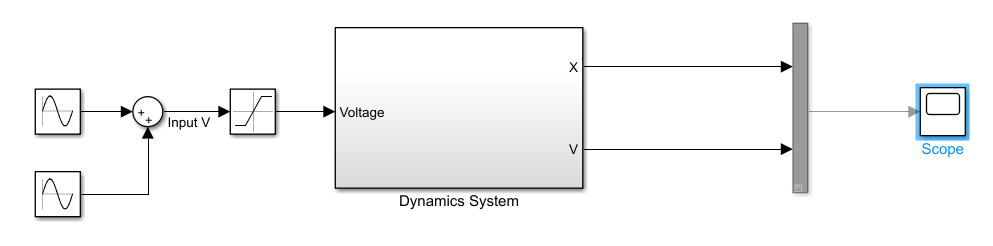
Where *di* is the energy input by the motor, and *ai* is *i*th row of A, [](https://www.codecogs.com/eqnedit.php?latex=a_i%20%3D%20%5B%5Cbar%7BH%7D%20%5C%20%5C%20%5Cbar%7BF%7D%5D#0)

for the *i*th data point collected. We then build the overdetermined system by collecting 10,000 points of data over 10 seconds and solve it as shown above.

The implementation of this algorithm is shown in MATLAB code below:

|  |
| --- |
| %% motor input: integral from (t0,tn) v\*q1dot d = cumtrapz(t,v.\*q1dot);  d = d(2:end);  %% get init conditions to subtract out h1\_init = 1/2\*q1dot\_init.^2; h2\_init = 1/2\*sin(q2\_init).^2.\*q1dot\_init.^2 + 1/2\*q2dot\_init.^2; h3\_init = cos(q2\_init).\*q1dot\_init.\*q2dot\_init; h4\_init = g\*cos(q2\_init);  %% Hbar equation: change in energy in system h1 = 1/2\*q1dot.^2 - h1\_init; h2 = 1/2\*sin(q2).^2.\*q1dot.^2 + 1/2\*q2dot.^2 - h2\_init; h3 = cos(q2).\*q1dot.\*q2dot - h3\_init; h4 = g\*cos(q2) - h4\_init;  Hbar = [h1 h2 h3 h4]; Hbar = Hbar(2:end,:);  %% Fbar equation: energy lost from damping Fbar = [cumtrapz(t,q1dot.^2) cumtrapz(t,q2dot.^2)]; Fbar = Fbar(2:end,:);  %% concatenate to get A A = [Hbar Fbar];  %% pseudo-inverse A to solve for theta Ainv = pinv(A); theta = Ainv\*d; |

Next, in order to test the algorithm, we developed a Simulink model based on our dynamics system that records the data from the simulation, which we can then plug in to our algorithm and test to see if we got the same physical parameters back that we started with. The model is shown in Figure 2, with the Dynamics System subsystem being the same as described in the previous lab.



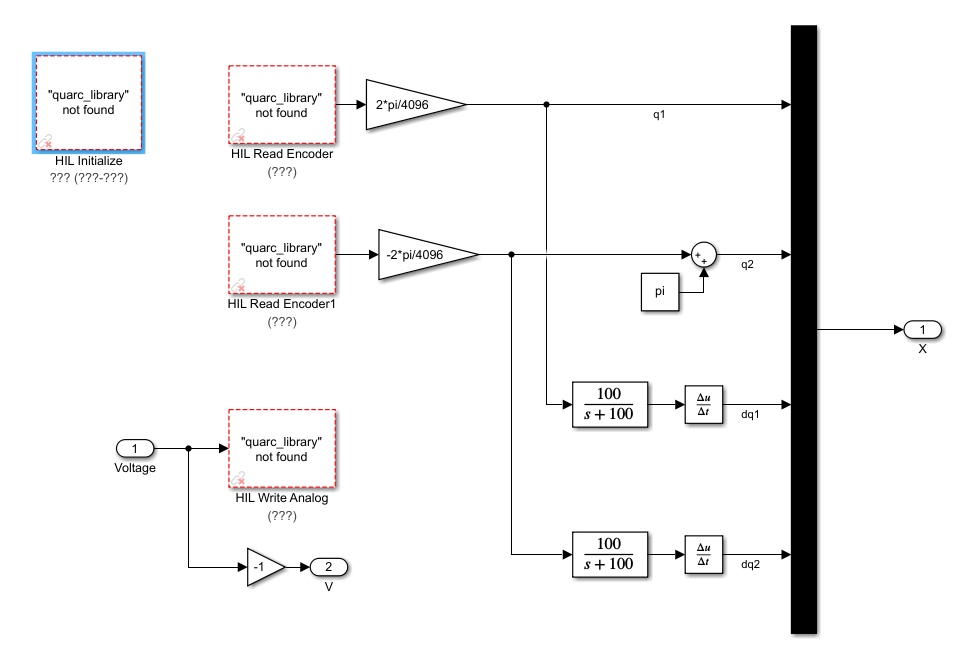
*Figure 2: Data Collector for Dynamics System*

The requirements of this test were to make sure that the motor input did not exceed what the physical motor could put out, which is the purpose of the saturation block, and to cause the pendulum arm to swing vertically so we can test the full range of the dynamics of the system. The input function that creating a good swing for us was



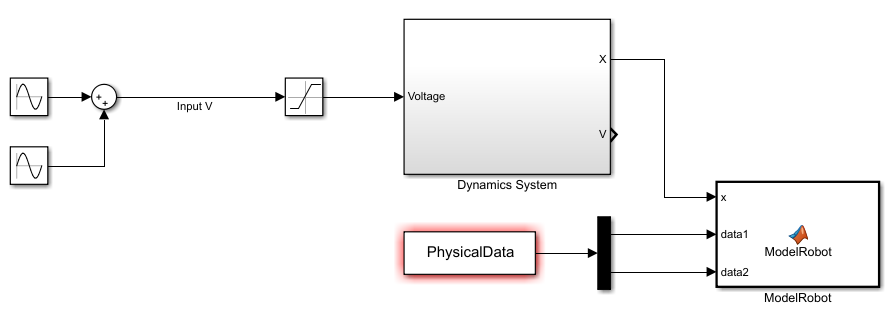
As described in the Introduction, we were able to get back the physical parameters we started with error of less than 0.01%. This confirms that the algorithm works, so we can now test it on the physical system in order to refine the physical parameters for the physical system.

We now need to collect data from the physical system instead of our dynamics model. This is the same as Figure 2, but replacing the Dynamics System with data collection from the physical encoders, and the input running the physical motor, shown in Figure 3. An important note is that we do not have a physical measurement for the velocity, so we filter the position signals and run them through differentiator blocks to get approximations of the joint angular velocities. This will provide some error to our results.



*Figure 3: Physical Data collector*

**NOTE:** This was the point where we got to before the University went online because of the Covid-19 pandemic. We were able to test the algorithm on the physical system but did not save the results. We tested the results of the algorithm by testing the dynamics system with the calculated physical parameters and comparing it with the data collected in the physical simulation, using the Simulink model shown in Figure 4. The simulation matched reasonably well visually to the physical data collected, verifying the results.



*Figure 4: Verifying the Physical Parameters Visually*