EENG498BC: Systems Exploration, Engineering, and Design Laboratory

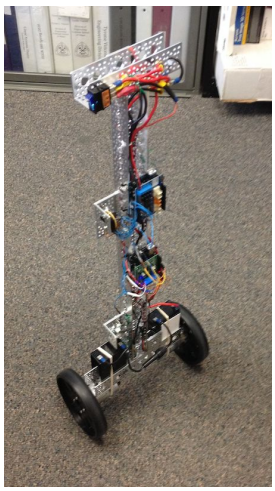
Spring 2016

Pendulum Robot Model

Vibhuti Dave and Tyrone Vincent*

1. Full Pendulum model

To design a controller for the balancing pendulum, it will be useful to have a simulation model to test your designs. A picture of of a two-wheeled robot and an idealized description is shown below. The robot is approximated as a two-mass pendulum, with angle ϕ defining the tilt, and x defining the translational position of the base of the robot. The force f and torque τ_m is due to the motors acting on the wheels.



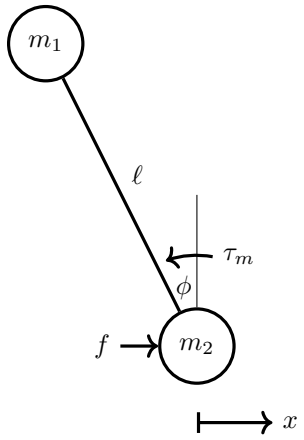


Figure 1: Idealized component diagram for pendulum robot

The differential equations that relate f and τ_m to the motion ϕ and x are the following.

$$\begin{aligned}\ddot{\phi} &= \frac{(g(m_1+m_2) - m_1\ell\cos\phi(\dot{\phi})^2)\sin\phi - \cos\phi b_x\dot{x} - \frac{(m_1+m_2)}{m_1\ell}\left(b_\phi\dot{\phi} - \tau_m\right) + \cos\phi f}{\ell\left(m_2 + m_1\sin^2\phi\right)} \\ \ddot{x} &= \frac{gm_1\cos\phi\sin\phi - m_1\ell\sin\phi(\dot{\phi})^2 - b_x\dot{x} - \frac{\cos\phi}{\ell}\left(b_\phi\dot{\phi} - \tau_m\right) + f}{(m_2 + m_1\sin^2\phi)}\end{aligned} \tag{1}$$

In these equations, we have also assumed translational and rotational friction, and b_x is the translational friction coefficient, while b_ϕ is the rotational friction coefficient.

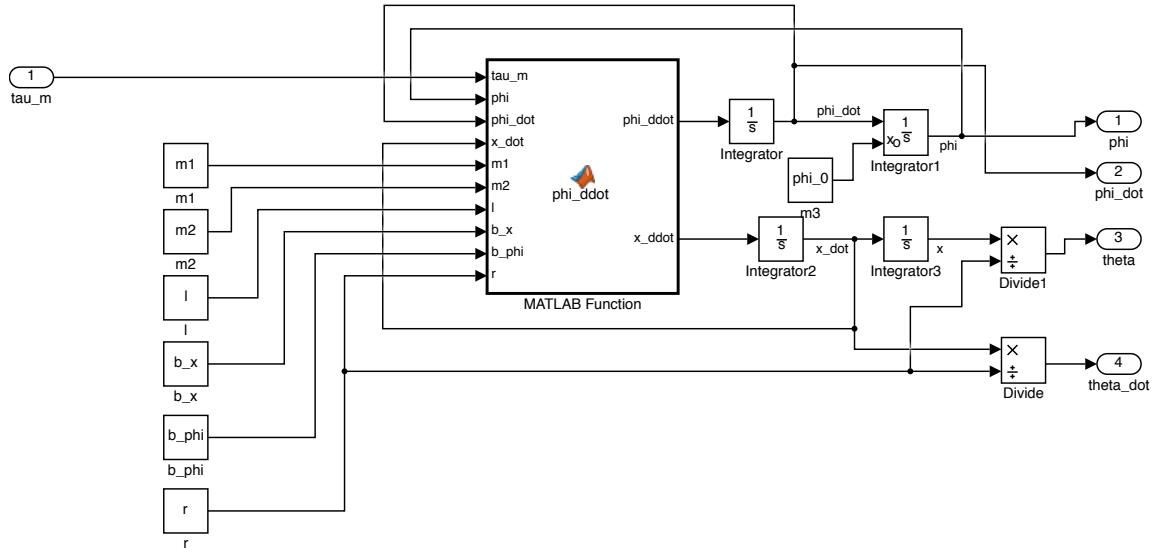
The motor will apply torque τ_m, but rotation of the wheels on the ground will cause force f to also appear. We need to relate these two signals. Assuming the inertia of the wheels and motor is much less than that of the robot, (so that the torque does not need to accelerate the inertial of the wheels) and that the wheels do not slip, the force on the wheels from the ground will be $f = \frac{\tau_m}{r}$. Thus, our final equations become

*Developed and edited by Tyrone Vincent and Vibhuti Dave

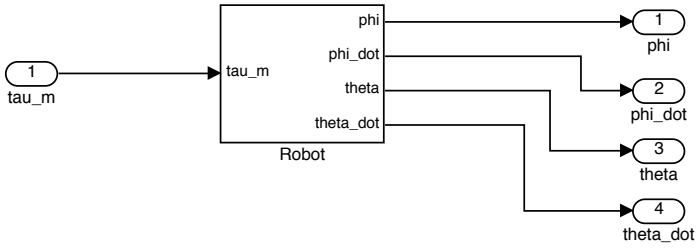
$$\ddot{\phi} = \frac{(g(m_1+m_2) - m_1\ell\cos\phi(\dot{\phi})^2)\sin\phi - \cos\phi b_x\dot{x} - \frac{(m_1+m_2)}{m_1\ell}b_\phi\dot{\phi} + \left(\frac{(m_1+m_2)}{m_1\ell} + \frac{\cos\phi}{r}\right)\tau_m}{\ell\left(m_2 + m_1\sin^2\phi\right)}$$
$$\ddot{x} = \frac{gm_1\cos\phi\sin\phi - m_1\ell\sin\phi(\dot{\phi})^2 - b_x\dot{x} - \frac{\cos\phi}{\ell}b_\phi\dot{\phi} + \left(\frac{\cos\phi}{\ell} + \frac{1}{r}\right)\tau_m}{(m_2 + m_1\sin^2\phi)} \tag{2}$$

These equations assume that m_2 is supported in the vertical direction (for example, by the wheels of the robot) but do not account for m_1 hitting the ground. Thus, you can also use these equations to simulate around $\phi = \pi$, which would look similar to how a gantry crane moves.

A simulink model to simulate this system would look like the following. The Matlab function block would implement the equations (2). Note that we have created the signals $\dot{\theta} = \frac{\dot{x}}{r}$ and $\theta = \frac{x}{r}$, which corresponds to the angular velocity and angular position of the wheel. Also note that we have made the initial condition of ϕ set externally to the integrator block, using the variable `phi_0`. This will make it easy for us to set for different types of experiments.

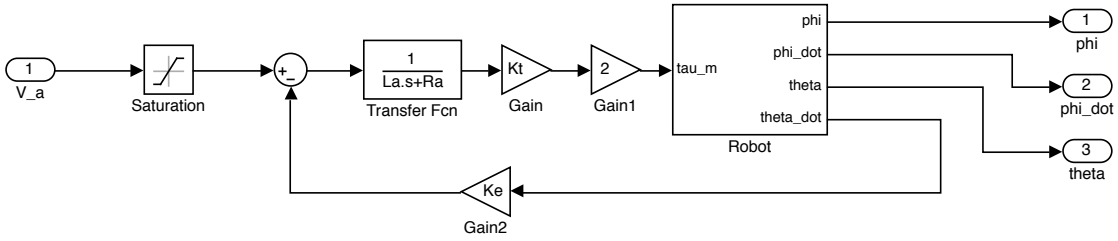


A powerful technique for both modeling and control design is *encapsulation*, where we can hide the details of a system or subsystem and just concentrate on the input output behavior. If we want to use this block diagram at a higher level of abstraction, we encapsulate everything inside the dotted lines as a single block. In Simulink you can do this by creating a subsystem. You can learn about creating subsystems here: `http://www.mathworks.com/help/simulink/ug/creating-subsystems.html`. This block shows only the inputs and outputs, and hides all of the internal details. After creating a subsystem, the Simulink block diagram looks like the following.

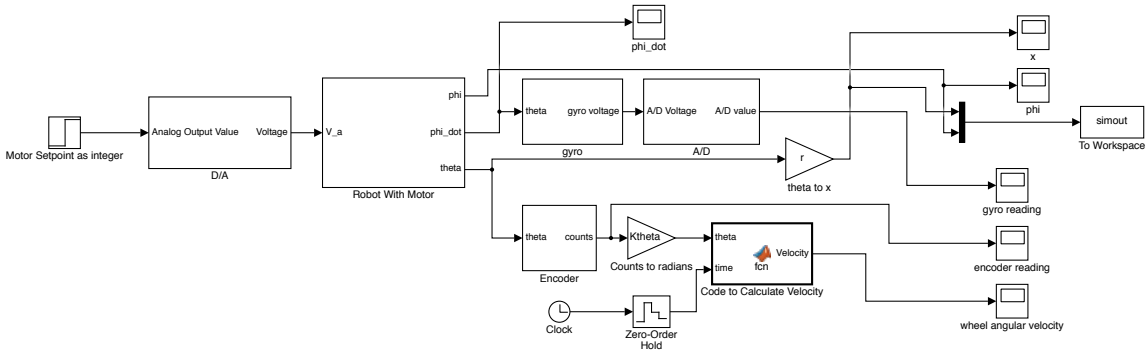


Note that we manually changed the names of the input and output nodes inside the subsystem in order to get these to show up on the subsystem block.

This model does not yet include the effect of the motors. There are two motors and two wheels, so note that there is a gain that multiplies the torque by 2. We also add a saturation block so that the max voltage is less than the battery voltage



We make this a subsystem, and then add some additional blocks that allow us to model some of the sensors and the D/A and A/D conversion of the Arduino. (The one thing that is not simulated is the D/A voltage. For ease of simulation, the equivalent continuous voltage is used instead of a PWM voltage.) You can find Simulink block diagram file on Blackboard. Click on each of the blocks and make sure you understand how they work.



To setup the parameters, we can create a script to run before we run the Simulink block diagram

```
%%
% Motor Parameters
%
stall_torque = 170; % oz-in
stall_current = 5; % A
Max_voltage = 12; % V
Max_RPM = 200; % rpm
Max_RPM_Current = .3; % A
%
% Motor Driver Parameters
%
Value_for_full_PWM = 255; % digital output for 100% PWM
%
% Encoder Paramaters
%
Counts_per_Revolution = 3200; % Encoder counts
%
```

```
% Gyro parameters
%
V_gyro_0 = 1.23 ; % Voltage for 0 rotation
S_gyro = 33.3 ; % mV/deg/s sensitivity
V_gyro_max = 3; % Max gyro output voltage
V_gyro_min = 0; % Min gyro output voltage
%
% Physical Parameters
%
Wheel_radius = 3; % in
l = .38; % m
m1 = .6; % kg
m2 = 1.9; % kg
b_phi=0; % rotational friction
b_x=0; % translational friction
La=0; % Motor inductance
Ts = .001; % Sample time of controller
% initial condition
phi_0 = pi; % initial condition for angle of pendulum

%%
% Conversions
%
r=Wheel_radius/39.37; % convert in to m
tau_max=stall_torque/141.612; % convert oz-in to Nm
K_gyro = (S_gyro / 1000) * (180 / pi); % convert mV/deg/s to V/rad/s
%
% Motor Constants
%
Kt = tau_max/stall_current;    % divide by stall current to get motor constant
Ra = Max_voltage/stall_current; % motor resistance
Ke = (Max_voltage - Ra*Max_RPM_Current)/(Max_RPM*2*pi/60); % back emf constant
%
% Other conversions
%
Kv = Max_voltage/Value_for_full_PWM; % digital output to V conversion for motor
Ktheta = 2*pi/Counts_per_Revolution; %
```

2. Suggested Next Steps

The parameters in this document probably do not correspond to those of your robot. You should do some experiments to determine them. If your robot is unstable, you will want to perform these experiments *upside down* so that the robot acts like a gantry crane. Suggested experiments are a drop experiment (start the robot at an angle of about 90 degress, and then drop, with zero motor voltage), and a pulse experiment (Set the motor to a constant PWM signal for 1 or 2 seconds). Adjust the parameters of the model until your experimental data matches your simulation.