Simulation and Control of a Mobile Inverted Pendulum

Aongus O'Murchadha
aongus.omurchadha@gmail.com

I. INTRODUCTION

This document is a write-up of the capstone project for the Coursera Robotics specialization developed by the University of Pennsylvania. The goal of the project is to simulate a real-world mobile robot that estimates its state from noisy sensor data in order to follow a commanded trajectory. Here, I describe a mobile inverted pendulum (MIP) and investigate its control given simulated sensor data. I derive the basic equations of motion, develop an extended Kalman filter to estimate the state variables from noisy sensor data, and combine the two with a goal trajectory.

II. MOBILE INVERTED PENDULUM

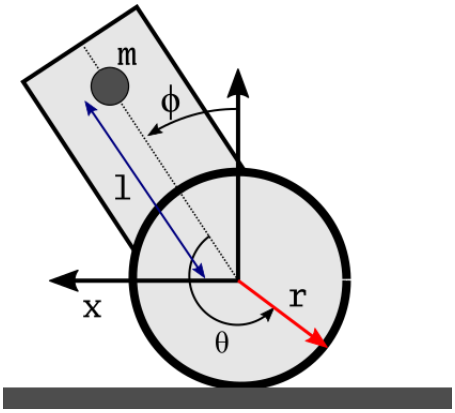

FIG. 1. MIP geometry. We assume the wheel is massless and that it rolls without slipping. The radius of the wheel is r and the distance between the center of the wheel and the center of mass of the body is l. The angles θ and ϕ are the roll and body angles, respectively.

The MIP can be described as a body of mass m_b sitting on top of a mobile wheel of radius r. If the MIP rolls without slipping, the translation x of the wheel is

$$x = (\theta + \phi)r, \tag{1}$$

where the angles θ and ϕ are defined in Fig. 1. The location of the centre of mass in $\mathbf{x}$ and $\mathbf{y}$ is

$$r_{cm} = r \begin{bmatrix} \theta + \phi \\ 0 \end{bmatrix} + l \begin{bmatrix} \sin\phi \\ \cos\phi \end{bmatrix} \tag{2}$$

Then if we assume that the wheel is massless, $m_w = I_w = 0$, the kinetic energy T and potential energy V of the MIP are

$$T = \frac{1}{2} m_b \dot{r}_{cm}^\top \dot{r}_{cm} + \frac{1}{2} I_b \dot{\phi}^2 \tag{3}$$

$$V = m_b g l \cos\phi \tag{4}$$

and we can derive the equations of motion in the variables (θ, ϕ) using the Lagrangian $\mathcal{L} = T - V$:

$$(D_q - D_t D_{\dot{q}})\mathcal{L} = \begin{bmatrix} \tau \\ 0 \end{bmatrix} \tag{5}$$

where τ is the torque on the wheel from the MIP's motor. In order to solve the equations of motion numerically or symbolically, we can re-express them in matrix form $\mathbf{Ax} = \mathbf{b}$:

$$\begin{bmatrix} r & r + l\cos\phi \\ r^2 + rl\cos\phi & r^2 + 2rl\cos\phi + l^2 + I_b r/m_b \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} l\dot{\phi}^2 \sin\phi - \tau/m_b r \\ gl\sin\phi + rl\dot{\phi}^2 \sin\phi \end{bmatrix} \tag{6}$$

We can check these equations of motion with a simple test - given $\tau = 0$ and an initial $\phi_0 = 1$, numerically solving the equations of motion shows that the MIP undergoes stable periodic motion in both angular variables (Fig. 2).

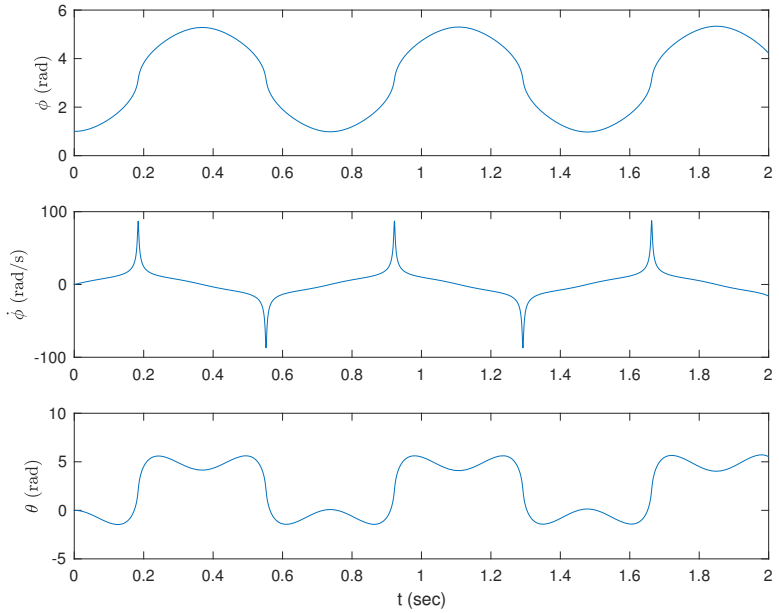

FIG. 2. MIP motion in the absence of motor torque ($\tau = 0$) given the initial condition $\phi_0 = 1$.

III. MIP CONTROL

If we want to control the MIP, an input torque τ has to be derived that produces the desired behaviour given the equations of motion. The MIP can be controlled in two ways - using a linear-quadratic regulator (LQR) generated automatically, or with a PID controller whose parameters must be either derived analytically or tuned by hand.

A. LQR Control

In general, given state variables x, our system is defined as a system of first-order differential equations for $\dot{x}$:

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \quad \dot{x} = f(x, \tau) \tag{7}$$

If we wish to linearize the equations of motion about upright ($\overline{x} = 0$), we can redefine our dependent variable x as $\tilde{x} = x - \overline{x}$. Then, linearizing using the Jacobian Df, the first-order differential equation for x becomes

$$\dot{\tilde{x}} \approx D_x f(\overline{x}, 0)\tilde{x} + D_\tau f(\overline{x}, 0)\tau = \mathbf{A}\tilde{x} + \mathbf{b}\tau \tag{8}$$

Given the $\mathbf{A}$ and $\mathbf{b}$ matrices, we can automatically generate a controller K using, for example, MATLAB's `lqr` function. Then our control input to the motor would be

$$\tau = Kx \tag{9}$$

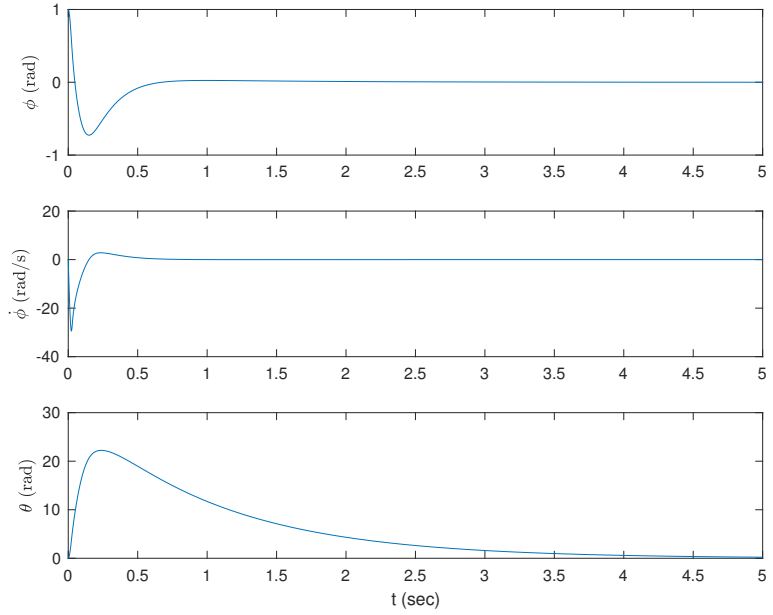

FIG. 3. The MIP stabilized about upright $\overline{x} = 0$ using an LQR controller

B. PID Control

The downside of an LQR controller is that the parameters are derived automatically for specific equations of motion with precisely-defined parameters. A more flexible alternative that allows for less-optimal control is the PID controller, whose input varies as the difference between the measured states and the desired states:

$$\tau = k_p(q_{des} - q) + k_d(\dot{q}_{des} - \dot{q}) + k_i \int_0^t q \, dt \tag{10}$$

The gain parameters k_d,k_p, and k_i can either be tuned by hand or derived from some tuning scheme.

IV. EXTENDED KALMAN FILTER

If we assume that the MIP gets its state information from an onboard accelerometer and gyroscope, the sensor measurement during the k-th time interval is approximately

$$z_k = h(x_k) = \begin{bmatrix} \sin\phi(t_k) \\ \cos\phi(t_k) \\ \dot{\phi}(t_k) \end{bmatrix}. \tag{11}$$

The first two entries, from the accelerometer, are the y- and z- components of the acceleration (gravity) projected on the body frame of the MIP. The third component is the rotation rate, from the gyroscope. Since the accelerometer measurements are nonlinear functions of the state variables, an extended Kalman filter (EKF) is needed. The algorithm can be summarized as follows for a linear state transition and nonlinear measurement function:

Given state $x_k = \begin{bmatrix} \phi(t_k) \\ \dot{\phi}(t_k) \end{bmatrix}$, the state transition is $x_{k+1} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} = A_k x_k$. The measurement is linearized using the Jacobian H of the measurement function h, in this case $H = Dh = \begin{bmatrix} \cos\phi(t_k) & 0 \\ -\sin\phi(t_k) & 0 \\ 0 & 1 \end{bmatrix}$.

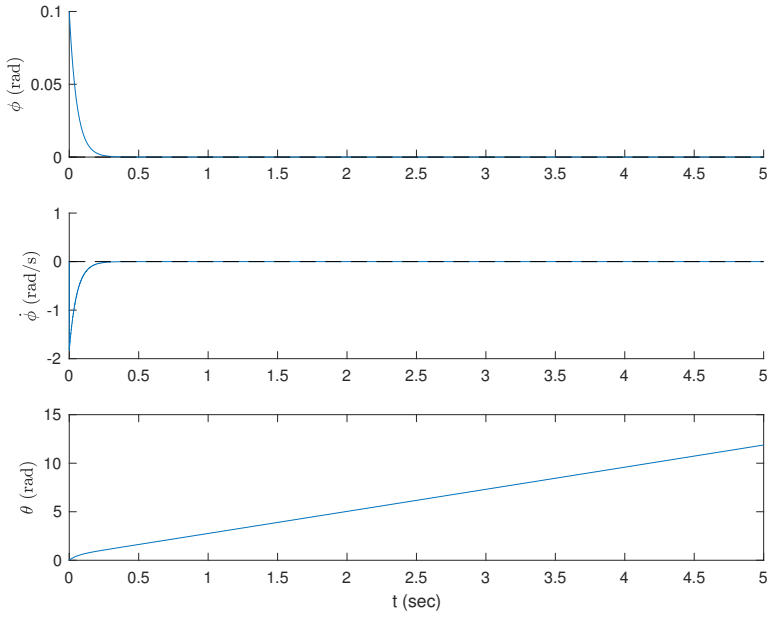

FIG. 4. The MIP stabilized about upright ($\phi, \dot{\phi} = 0$) using a PID controller. The roll angle θ is not controlled.

We also have the state covariance P_k, the measurement covariance R, and the state transition covariance Q. In this instance we assume that R and Q are constant. Then the algorithm for the EKF is

$$\begin{aligned} \overline{x}_k &= Ax_{k-1} \\ \overline{P}_k &= AP_{k-1}A^\top + Q \\ K_k &= P_k H_k (H_k P_k H_k^\top + R)^{-1} \\ x_k &= \overline{x}_k + K_k(z_k - h(\overline{x}_k)) \\ P_k &= \overline{P}_k - K_k H_k \overline{P}_k \end{aligned} \tag{12}$$

Given some noisy test data, we can tune Q to get a filter output that tracks the data and thereby gives an estimation of the state variables (Fig. 5).

A. EKF for MIP Control

Now that we have a way of estimating the state variables $\phi, \dot{\phi}$ from noisy sensor data, we can use this to make a PID controller (Eq. 10) that can command the MIP to follow an arbitrary trajectory. First, we check that our method allows us to control the body angle about upright ($\phi, \dot{\phi} = 0$). Rather than numerically solving the equations of motion directly, the differential equations are solved for a single time bin at a time, with the values of the state variables from the previous time bin as initial conditions. This allows us to add random noise to ϕ and $\dot{\phi}$ and pass them to the measurement function $h(x)$ (Eq. 11). The EKF (Eq. 12) estimates the original variables and passes them to a PID controller to obtain the wheel torque. Then the equations of motion can be solved given the noisy state variables and the estimated desired torque to see if the MIP can be stabilized (Fig. 6).

B. EKF for MIP Motion

One advantage of a PID controller is that it is easy to input and modify a desired trajectory for the MIP. Instead of stabilizing about upright, we test two trajectories - a step trajectory where the MIP is commanded to move from one point to another (Fig 7), and a sine trajectory where the desired location is a constantly-varying sine function of time (Fig 8).

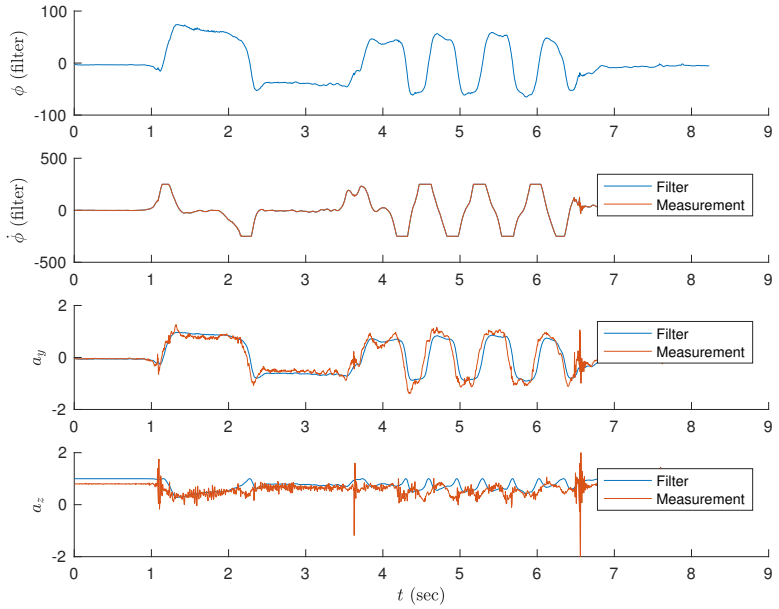

FIG. 5. The EKF estimation of the state variables $\phi, \dot{\phi}$ and accelerations a_y, a_z given noisy test data.

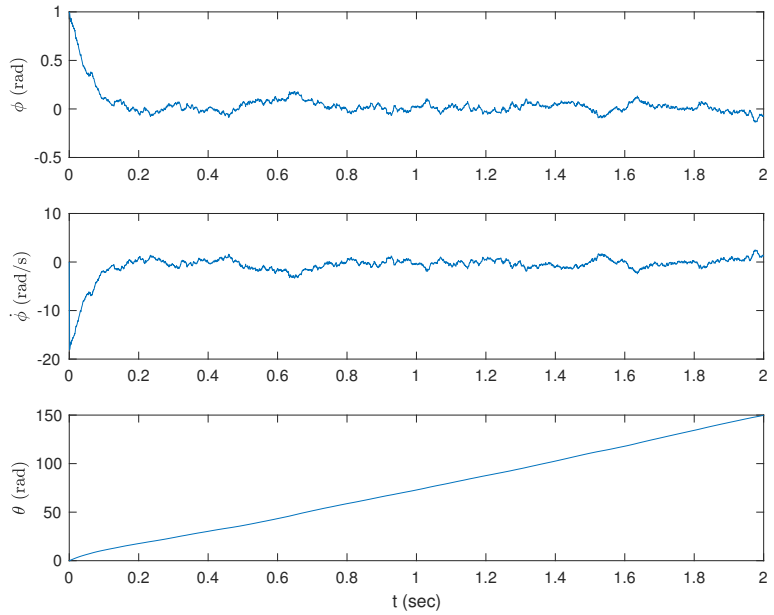

FIG. 6. The MIP stabilized about upright ($\phi, \dot{\phi} = 0$). Given simulated noisy sensor measurements, an EKF estimates the state and provides the input to a PID controller. The roll angle θ is not controlled.

Since the x-location is not one of the state variables of the MIP, we control for it by cascading an x-controller into a ϕ-controller. Given a goal trajectory in x and that $x = r(\theta + \phi)$ in the absence of slipping, we can define errors in x and $\dot{x}$. Then the goal ϕ, ϕ_{des}, is equal to the sum of the x-errors and the output motor torque is the output of a PD controller for ϕ.

Animations of the MIP following the step and sine trajectories can be found in the same folder as this document.

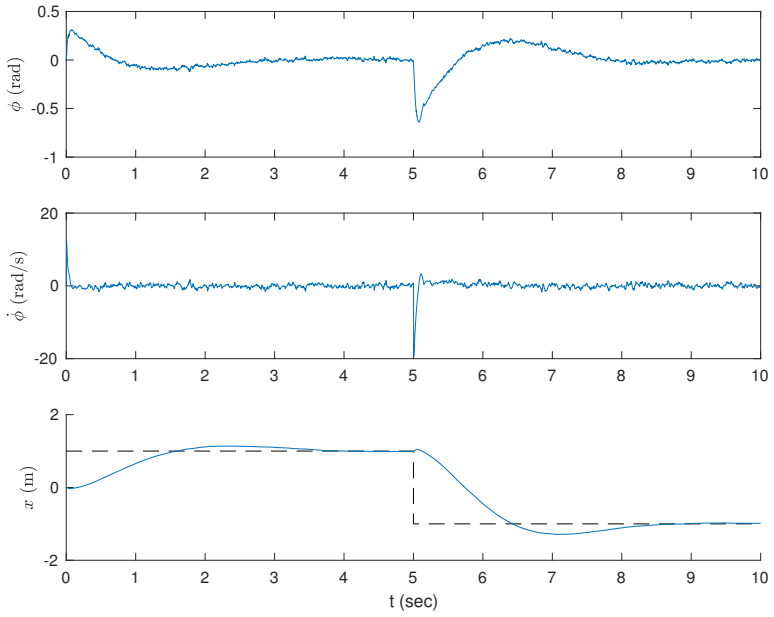

FIG. 7. The MIP following a step trajectory from $x = 1$ to $x = -1$. The desired x-position is shown in the bottom panel by the dotted line. An animation of the MIP following this trajectory can be found in the same folder as this document.

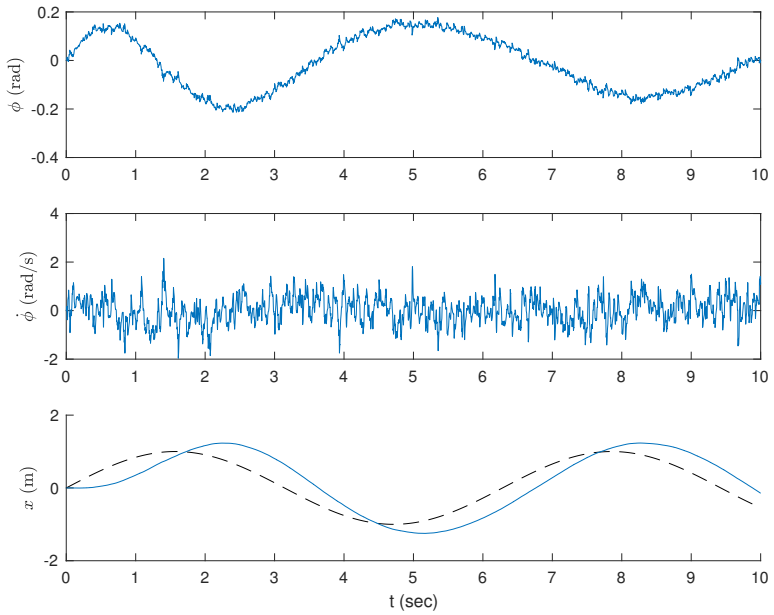

FIG. 8. The MIP following a sinusoidal trajectory. The desired x-position is shown in the bottom panel by the dotted line. An animation of the MIP following this trajectory can be found in the same folder as this document.