Final Assignment Report – COMP 417

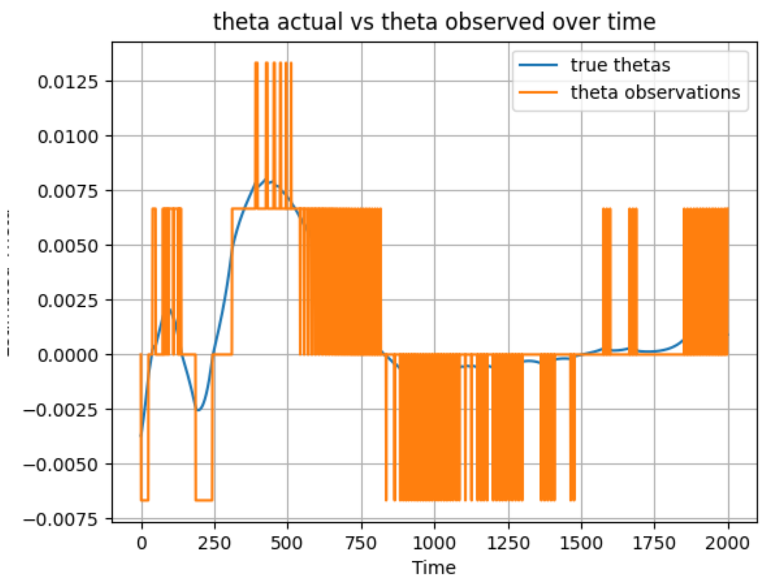
Ben Hepditch

Dec. 9, 22

Part 1) Visual Tracking

To estimate the angle (radians) between the pole and the cart at each time step I used a combination of methods available in OpenCV. Specifically, my program uses color masking and image thresholding to convert all but the pixel values of the red pole into white while the rest of the image is converted into black. The pixels are then converted to greyscale, at which point the program attempts to find the rectangular contours in the image, which is just the pole. Finally, the angle of the pole with the cart can easily be obtained using the cv2.minAreaRect() method, at which point the program merely converts the angle to negative or positive depending on the quadrant the pole is in (ie, negative if it's leaning left, positive if it's leaning right). *To view the results of the object detection live, un-comment line 45 in cart_pole_angle_detector.py.*

Results



Part 2) Kalman Filter

Using the following equations, I was able to obtain highly-optimal results with a Kalman filter. These equations are implemented in PID_Controller.py, with a custom KalmanFilter class.

Propagation (motion model):

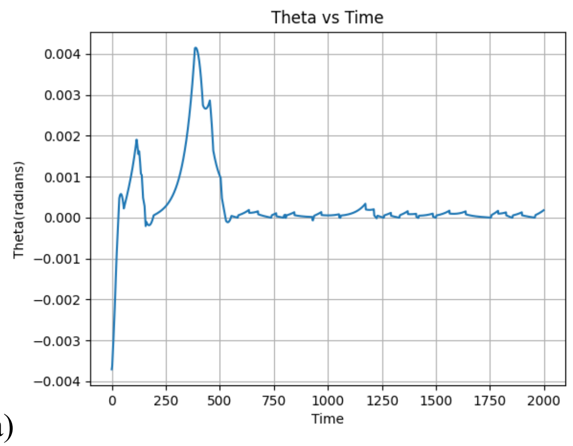
$\hat{x}_{t+1/t} = F_t \hat{x}_{t/t} + B_t u_t$	- **State** estimate is updated from system dynamics
$P_{t+1/t} = F_t P_{t/t} F_t^T + G_t Q_t G_t^T$	- **Uncertainty** estimate GROWS

Update (sensor model):

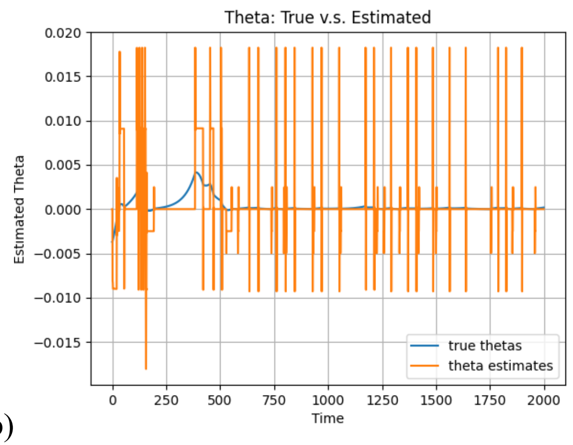
$\hat{z}_{t+1} = H_{t+1} \hat{x}_{t+1/t}$	- Compute expected value of sensor reading
$r_{t+1} = z_{t+1} - \hat{z}_{t+1}$	- Compute the difference between expected and "true"
$S_{t+1} = H_{t+1} P_{t+1/t} H_{t+1}^T + R_{t+1}$	- Compute covariance of sensor reading
$K_{t+1} = P_{t+1/t} H_{t+1}^T S_{t+1}^{-1}$	- Compute the Kalman Gain (how much to correct est.)
$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$	- Multiply residual times gain to correct state estimate
$P_{t+1/t+1} = P_{t+1/t} - P_{t+1/t} H_{t+1}^T S_{t+1}^{-1} H_{t+1} P_{t+1/t}$	- Uncertainty estimate SHRINKS

I found that setting the values of my Q and R matrices such that Q > R, was the most effective tuning mechanism. As shown in Figure b), the estimated values of theta were accurate to within ~+/-0.01 radians, well within a range required to balance the pole. Figures c) and d) clearly demonstrate that there's a strong covariance between the estimated values of theta and the actual values of theta, as well as between the estimated values of theta and theta dot. Overall, using the filter yielded much more stable results.

Results

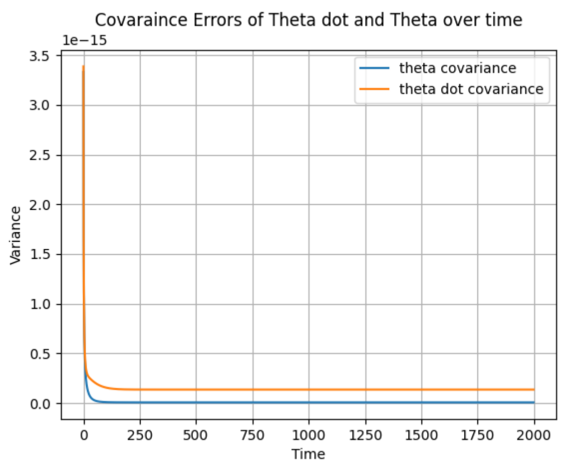


a)

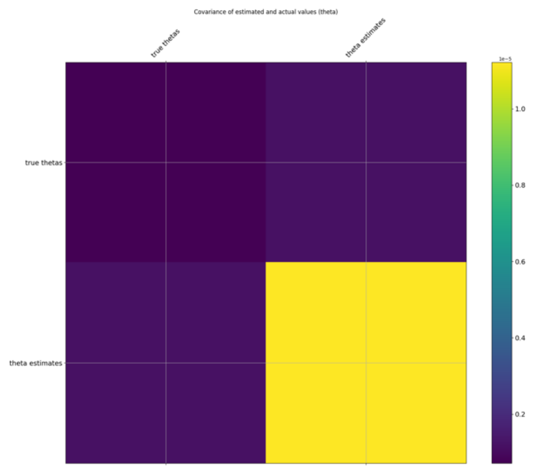


b)

c)



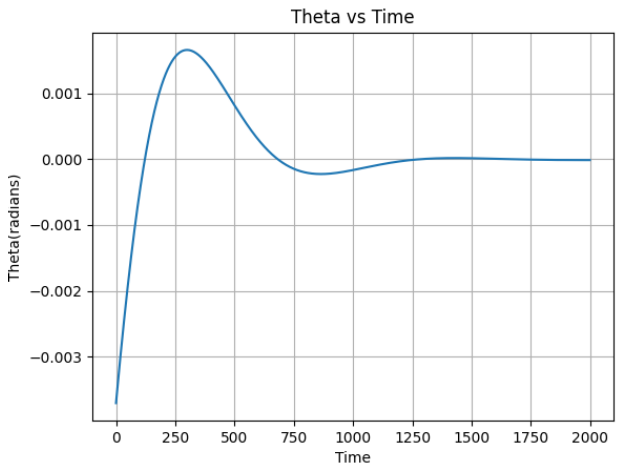
d)



Part 3) PID Controller

As the results in a) show, using the internal state values of theta and theta dot, I was able to obtain a highly-stable pole. To tweak the gains I used the Ziegler-Nichols method to tune my PID. Theta dot was calculated using the time delta of 0.005, as is used in the game.

a)


Theta vs Time
Theta(radians)
Time