16-299 Assignment: Controlling a TWIP

Introduction

This assignment focuses on the modeling and control of a two-wheeled inverted pendulum (TWIP) which is also the focus of the lab. Segways are TWIPs. The lab robot is a TWIP.

I want to make the point that this assignment and the lab are a good starting point for your project. There is an immense literature (assignments, papers, theses, web pages, videos, ...) on the web on modeling and controlling the cart-pole system. There is a more limited literature on TWIPs. You can use what you find to help you on this assigment as well as inspire a project. For example "sliding model control" is mentioned in these papers. Check them out for project ideas!

For example, here are controllers you can find by searching for "segway" on matlab.com:
`https://www.mathworks.com/matlabcentral/fileexchange/70498-lqr-and-kalman-f`
`https://www.mathworks.com/matlabcentral/fileexchange/95158-lego-ev3-balance`
`https://www.mathworks.com/matlabcentral/fileexchange/60322-gyroboy-self-bal`
`https://www.mathworks.com/matlabcentral/fileexchange/19147-nxtway-gs-self-b`
`https://www.mathworks.com/matlabcentral/fileexchange/88768-two-wheeled-self`
Here are some theses and papers on TWIPs, that cover how to do this assignment:
`https://dspace.mit.edu/bitstream/handle/1721.1/69500/775672333-MIT.pdf`
`https://smartech.gatech.edu/bitstream/handle/1853/44897/castro_arnoldo_201208_mast.pdf`
`https://www.researchgate.net/publication/338674336_Model-Based_LQR_Control_of_Two-Wheeled_Balancing_Robot`
`https://www.researchgate.net/publication/299282248_Controllers_Comparison_to_stabilize_a_Two-wheeled_Inverted_Pendulum_PID_LQR_and_Sliding_Mode_Control`
`https://zenodo.org/record/1125869/files/10005085.pdf`
`https://www.semanticscholar.org/paper/Position-Tracking-Controllers-for-Two`
`37423fa5538c48c29dfa7c29b9e0af5cafd6bcaf`
Google "control twip segway", "lqr control twip segway", "PID lqr control twip segway", or "*method* control twip segway" to get lots more. Try not be confused by different choices of notation, state vectors, and/or sign conventions.

Parameters and States

Parameters for a planar TWIP include:

- m_p = the mass of the inverted pendulum.
- l_p = the distance along the inverted pendulum between the wheel axis and the center of mass of the pendulum. The pendulum being straight up defines the zero pitch angle.

- $\mathbf{I}_p$ = the moment of inertia of the pendulum (about the COM or wheel axis?).
- m_w = the mass of the combined wheels.
- r_w = the radius of a wheel.
- $\mathbf{I}_w$ = the moment of inertia of the combined wheels (about the COM or wheel axis?).

Elements of the state vector of a planar TWIP are:

- x = the position of the ground contact point or wheel axis along the x axis.
- θ = the angle of the body/inverted pendulum with respect to vertical. Learning forward is positive, which is not the usual angle convention of positive angles are counter-clockwise.
- $\dot{x}$ = the forward velocity of the ground contact point or wheel axis.
- $\dot{\theta}$ = the angular velocity of the pendulum with respect to vertical.

Another state vector would swap wheel angle ϕ for x. These are functionaly equivalent since $x = r_w\phi$

Deriving the Dynamics

I have derived the forward dynamic equations for you, both in Matlab and in Mathematica. Look in the directory `http://www.cs.cmu.edu/~cga/controls-intro/twip/` for matlab.zip and twip.math, respectively. Feel free to try to rederive these dynamics, and/or check these derivations for errors. How could you check my work? One good way is to create a TWIP model in a simulation package like ODE, Gazebo, or Bullet and see if the resulting simulations matched. Only do this if you think it will be fun. Here is a similar derivation of Lagrangian dynamics for a twip:
`https://ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-co`
`idsc-dam/Lectures/System-Modeling/HS19/Lecture_Slides/LagrangeCaseStudy_`
`HS2019.pdf`

These derivations are impressively complicated (figure out why and they could make good projects):
`https://www.researchgate.net/publication/321730779_Nonlinear_reduced_`
`dynamics_modelling_and_simulation_of_two-wheeled_self-balancing_mobile_`
`robot_SEGWAY_system`
`https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6076349&casa_`
`token=wFmTdyqyA9wAAAAA:I9irHqfaoaK9hW6GX_7TYp1aBQaNLtMuorXd96BYwhNY8McnAYNz`

Key features of the dynamic model: The pendulum rotates about the wheel axis. The cart-pole is similar to a TWIP, but there is no rotation of the cart about a horizontal axis. This assignment focuses on a planar TWIP (no turning). The two wheels are combined into a single wheel. The motor is treated as a perfect torque source. The torque on the wheel also acts on the pendulum directly. The wheel is not deformable. We ignore the possibility of the wheel lifting off the ground (which can be done by swinging the pendulum down really fast). Wheel lifting off the ground and/or slip makes modeling the TWIP very complicated, so we will not allow wheels to slip.

Extra credit/a **project** is available if you augment the model to include varying ground contact force, slip, deformable tires, and liftoff, and your controller can handle these issues. Slip can be caused by the wheel/ground friction being small (ice or snow, liquid on ground, oily ground, soil mechanics, ...), or there is rough terrain or bouncing that cause the force between the wheel and the ground to become small or zero (such as a pothole or driving off a cliff).

The coordinate system we will use has x forward and y upward. Rotation of the pendulum forward is positive pitch. Positive torque pitches the pendulum forward (which would roll the wheels backwards if starting upright at rest). Extra credit/a **project** is available to make a 3D model and control and the lab robot to drive along paths and turn.

Other **projects** involve 1) driving over hilly terrain, curbs, and potholes, and 2) getting the robot to dance to music.

TWIP dynamics

The simplest way to express TWIP dynamics is using the equation $\mathrm{M}(\mathbf{x})\mathbf{a} = \mathbf{v}(\mathbf{x})$, where M is an inertial matrix, $\mathbf{a}$ is the vector of system accelerations, and $\mathbf{v}$ is a vector of the sum of centripetal and Coriolis fictitious forces (quadratic in velocities), gravity (look for a G), actuation (in our case τ), and external forces (none in our case):

$$\begin{pmatrix} \frac{I_w}{r_w^2} + m_p + m_w & l_p * m_p * \cos(\theta) \\ l_p * m_p * \cos(\theta) & I_p + l_p^2 * m_p \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} r_w * (-\tau + \dot{\theta}^2 * l_p * m_p * r_w * \sin(\theta)) \\ \tau + \mathrm{G} * l_p * m_p * \sin(\theta) \end{pmatrix} \quad (1)$$

Given M and $\mathbf{v}$, we can solve for the accelerations using $\mathrm{M}^{-1}\mathbf{v}$, which is what the dynamics subroutine **twip.m** in **twip-manual.zip** does.

The Actual Assignment

A. Try to design a controller manually

A1) Given the example Matlab program in **twip-manual.zip**, try to design satisfactory manual gains.
A2) Set the duration to 100.0 (100s). Try to minimize the one step criterion $dt * (x^2 + \theta^2 + \tau^2)$ by manually choosing gains. What are your best gains?

B. Linearize the dynamics

Linearize the TWIP dynamics in the straight up configuration. What are $\mathbf{A}$ and $\mathbf{B}$? Which of the symbolic terms in the dynamics in the $\mathbf{M}(\mathbf{x})\mathbf{a} = \mathbf{v}(\mathbf{x})$ equation above could you have ignored in the first place? Why?

C. Design an LQR controller

C1) Choose an optimization criterion ($\mathbf{Q}$ and $\mathbf{R}$) and design a corresponding LQR controller. Implement it for the simulation (and later for the lab). What is your $\mathbf{Q}$, $\mathbf{R}$, and corresponding $\mathbf{K}$? Evaluate the resulting control in simulation, and explain why you chose the optimization criterion you did.

C2) Vary the diagonal components of $\mathbf{Q}$ and $\mathbf{R}$ over several orders of magnitude (..., 1e-2, 1e-1, 1, 10, 100, ...) and generate a root locus plot for each component. When do which components of $\mathbf{Q}$ and $\mathbf{R}$ matter? What is the minimal set of $\mathbf{Q}$ and $\mathbf{R}$ components that give you good control of pole locations?

D. Design a Kalman filter

D1) Choose a noise model, design a Kalman filter based on measurement of only x and θ, and implement it for the simulation (and later for the lab). What is your $\mathbf{Q}_{\mathrm{KF}}$, $\mathbf{R}_{\mathrm{KF}}$, and corresponding $\mathbf{K}_{\mathrm{KF}}$? Evaluate the resulting control in simulation with added noise, and explain why you chose the optimization criterion you did.

D2) Explore two Kalman filter designs. D2a) Generate a Kalman filter design with time constants much faster than your LQR controller. Plot the pole locations for the combined system. D2b) Generate a Kalman filter design with time constants about the same as your LQR controller. Plot the pole locations for the combined system. What is the difference in command response between this KF+LQR combination, and the same LQR design with full state feedback.

E. Robustness

This is a separate assignment that will come later.

E1) Explore parametric uncertainty by varying the parameters and using the same LQR controller from above. Which parameters can you change by how much? An easy thing to do is put a can or other object on top of the actual robot and see what happens.

E2) Add actuator dynamics. Use a motor that has first order dynamics for torque generation with a 0.1s time constant. Note that this would be similar to taking into account the reaction time and other delays of a human driver. How much can you vary the actuator time constant and have your LQR controller still work. This could be turned into a **project** by trying to accurately model the lab robot including its actuation, and another TWIP we can provide.

E3) *DRAFT* Add a second order underdamped subsystem to the TWIP (like adding an antenna or fishing pole with a weight on the top to the TWIP). How much "jiggliness" can your LQR system take. We can characterize a jiggly subsystem by its natural frequency, damping ratio, and "mass" relative to the main system. A more complex load is a half empty bottle of water, so the water can slosh around.

F. Other approaches and designs

This is a separate assignment that might come later.

F1) Do you need the x in the state vector (removing it leads to a state vector (θ,

F2) Should the robot be short or tall? Tall makes it fall over more slowly, but short means that smaller torques can recover from the same θ errors. The same "impulse" (the integral of force over the duration of a force pulse) Which robot can handle bigger θ errors? Bigger parametric uncertainty? What happens if the center of mass of the pendulum is below the wheel axis?

F3) Apply "deep" reinforcement learning to learn a policy to control a TWIP. Make it work in simulation first, and then on the real robot.