

This problem set is due *by 11:59pm* on **Tuesday, March 3**.

**Problem 1 (Value Iteration on the Double Integrator)** *In this problem, we'll reconsider the optimal control problem on the double integrator (aka, unit mass brick on ice), described by*

$$\ddot{q} = u,$$

*using the Value Iteration algorithm. An implementation of that algorithm is available for you in the file `brick_vi.m`, which is available for download on the course website. This is a complete implementation of the algorithm with discrete actions and volumetric interpolation over state; you only need to define the mesh location, the action set, and the cost function.*

- a) *Use value iteration to compute the optimal policy and optimal cost-to-go for the minimum-time problem. Submit your code and the resulting plots. Is it different from the analytical solution we found in lecture? Explain.*
- b) *Use value iteration to compute the optimal policy and optimal cost-to-go for the quadratic regulator problem, using the cost function*

$$g(q, \dot{q}, u) = \frac{1}{2}Q_p q^2 + \frac{1}{2}Q_d \dot{q}^2 + \frac{1}{2}R u^2.$$

*You should appropriately set the gains  $Q_p, Q_d, R$  to achieve the task. Submit your code and the resulting plots. How does this compare to the analytical solution from lecture?*

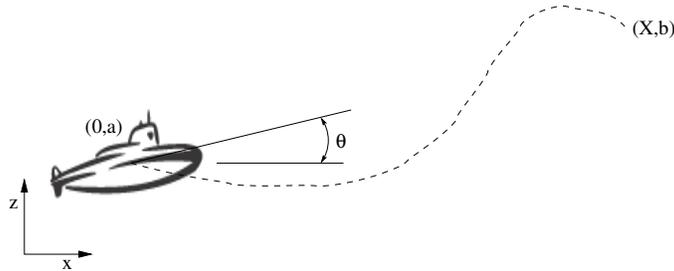
- c) *Design a new cost function which causes the optimally controlled brick to oscillate stably (in a stable limit cycle), rather than stabilizing a fixed point. Note that, because the value iteration solves infinite-horizon problems (and assumes that the optimal policy/cost-to-go do not depend on time), your cost function cannot depend on time. Submit your cost function, plots of the resulting policy and cost-to-go functions, and convincing trajectories from the simulation with the optimal policy.*

*Although we will not require it for this assignment, you should note that this code can be easily edited if you wish to examine optimal control solutions for the simple pendulum, or any other two-dimensional problem. Extending it to higher dimensions will require a slightly more general implementation of the volumetric interpolation.*

**Problem 2 (Pontryagin Minimum Principle)** Consider a 2D robotic submarine with a position given by  $(x, z)$ . Assume that the submarine is operating in a very viscous medium, yielding first-order dynamics, and that the control input is the pitch  $\theta$ , encoded such that  $u = \tan(\theta) = dz/dx$  (this encoding is used simply to make the mathematics more tractable). We will assume that the thrust is always a constant, and that the environment's resistance to motion (viscosity), varies inversely with  $z$ -position, and that gravitational effects are negligible, resulting in equations of motion:

$$\begin{aligned}\dot{x} &= z \cos(\theta) \\ \dot{z} &= z \sin(\theta) \\ \theta &= \tan^{-1}(u)\end{aligned}$$

Our goal for this problem is to obtain the minimum-time trajectory for the submarine to get from a position  $(0, a)$  to a position  $(X, b)$ .



The trick for solving the problem simply will be to reparameterize the trajectory in terms of  $x$ , instead of time. We will therefore assume that the robot will not backtrack ( $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ ). Let us proceed with the optimal control derivation using the following steps:

- Compute the time taken to travel from an  $x$ -position  $x_0$  to  $x_0 + dx$  as a function of  $u$  and  $z$  (you should be able to cancel all  $\theta$ 's). Then write the total time (this will be the minimum-time cost function) as a definite integral over  $x$  with limits of 0 and  $X$ . The integrand should be a function of only  $u$  and  $z$ .
- Compute the dynamics of the system  $z$  with respect to  $x$  (eg,  $\frac{\partial z}{\partial x} = \dots$ ).
- Now, give the Hamiltonian for this problem and the adjoint equation. **Hint:** To see things in the more standard form, you might literally replace the  $x$  in your dynamics equation and your cost function with a  $t$  (it's perfectly legal).
- Recalling that the Hamiltonian must be constant throughout the optimal trajectory for problems of this form (i.e., problems where the cost is not a function of the parameterizing coordinate, which is in this case  $x$ ), give a differential equation that defines the optimal trajectory (and thus the optimal policy  $u^*(x)$ ). You don't have to worry about the specific values of any constants.
- Either through simulating or examining the equation, what form of curve does this differential equation describe? Can you give the equation of this curve? Again, don't worry about the values of any constants which may appear.

**Problem 3 (Swing-Up and Balance for the Cart-Pole System)** *In this problem we'll implement the energy-based swing-up and LQR balancing controllers for the cart-pole system. You can start by downloading `cartpole.m` from the course website. For all parts, use the system parameters:  $m_c = 10\text{kg}$ ,  $m_p = 1\text{kg}$ ,  $l = 0.5\text{m}$ ,  $g = 9.8\text{m/s}^2$ .*

- a) **Controllability.** *Linearize the dynamics of the cart-pole system about the desired fixed point,  $\mathbf{x}^* = [0, \pi, 0, 0]^T$ ,  $\mathbf{u}^* = 0$ . Is this linear system controllable? Show your work.*
- b) **Balancing.** *Implement LQR to stabilize the desired fixed point. Submit the resulting LQR policy, and trajectories of the nonlinear system following this policy. Explore the basin of attraction of this controller and define a coarse (conservative) approximation of this basin that will define your switching surface (note we are not asking for any detailed basin of attraction analysis).*  
**Hint:** *In addition to using the `lqr` command in Matlab, you might consider using `dlqr` which will explicitly take into account the discreteness of your integration step.*
- c) **Swing-Up.** *Implement the energy-based swing-up controller described in lecture. Using the switching law you defined in part (b), have your controller switch to the LQR solution for balancing at the top. Submit your control code, and a plot of the trajectory which takes the system from  $\mathbf{x} = [0, 0, 0, 0]^T$  to  $\mathbf{x} = [0, \pi, 0, 0]^T$ .*

MIT OpenCourseWare  
<http://ocw.mit.edu>

6.832 Underactuated Robotics  
Spring 2009

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.