Problem 1 (Numerical Optimal Control Algorithms) *An ice skater is skating around a pond that has several weak spots in the ice. The equations of motion are:*

$$\dot{v} = u_1, \tag{1}$$

$$\ddot{\theta} = u_2. \tag{2}$$

where

$$\dot{x} = -v\cos\theta, \tag{3}$$

$$\dot{y} = -v\sin\theta. \tag{4}$$

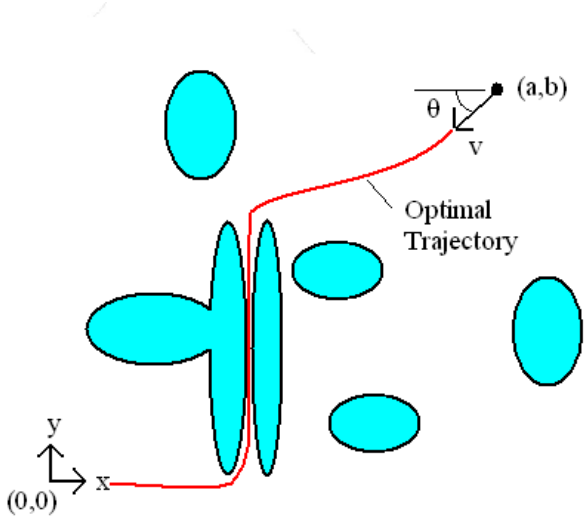
Note that this enforces a nonholonomic "skating" constraint which effectively prevents the skate from slipping sideways across the ice.

The red line gives the optimal trajectory when they are trying to get from point (a,b) to (0, 0) while minimizing a quadratic cost $g(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}$.

a) *Is value iteration likely to find a policy such that when simulated from* (a, b)*, the trajectory it produces closely approximates the optimal trajectory? Does it depend on the coarseness of the mesh, and if so, how? How many dimensions must you tile over for this problem?*

b) *Is a shooting method likely to find a trajectory that closely approximates the optimal trajectory? What issues might you run into if you initialize the method with a random trajectory?*

c) *Would direct collocation likely find a trajectory that closely approximates the optimal trajectory? How does it depend on the spacing between state/action pairs? How does it depend on the initial trajectory?*

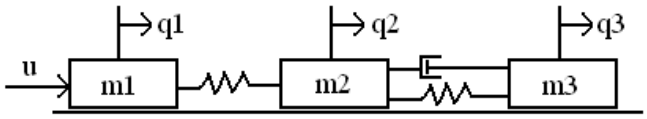


Problem 2 (PFL) *Please answer the following questions about partial feedback linearization.*

Three masses are attached by springs and dampers as shown in the accompanying figure, and have the following equations of motion:

$$\ddot{q}_1 = u - kq_1 + kq_2, \tag{5}$$
$$\ddot{q}_2 = kq_1 - 2kq_2 + kq_3 - b\dot{q}_2 + b\dot{q}_3, \tag{6}$$
$$\ddot{q}_3 = kq_2 - kq_3 + b\dot{q}_2 - b\dot{q}_3. \tag{7}$$

a) *What is the expression for the collocated PFL input $u = f(\mathbf{x}, \ddot{q}_{1desired})$ in which you attempt to linearize the control of mass 1?*



b) *Is it possible to perform a noncollocated PFL of mass 3? Why or why not? If it is possible, give the $u = f(\mathbf{x}, \ddot{q}_{3desired})$ that accomplishes the PFL.*

c) *What if you wish to control $h(\mathbf{x}) = q_3 - q_1$? Is this possible, and if it is, give the $u = f(\mathbf{x}, \ddot{h}_{desired})$ that accomplishes the PFL.*

Problem 3 (Dynamic Programming with Continuous Actions) *In this problem we'll explore a variant of the discrete dynamic programming algorithm, where we avoid discretizing the action space (but still discretize state and time). Consider a mechanical system described by the manipulator equations:*

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}(\mathbf{q})\mathbf{u},$$

and an quadratic regulator instantaneous cost function:

$$g(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}.$$

a) Rewrite the discrete-time dynamics of the system in the form

$$\mathbf{x}[n+1] = \mathbf{f}(\mathbf{x}[n], \mathbf{u}[n]),$$

where the discretization is done by Euler integration ($z[n+1] = z[n] + \dot{z}dt$).

b) The discrete-time dynamic programming equation is

$$J^*(\mathbf{x}, n) = \min_u \left[g(\mathbf{x}, \mathbf{u}) + J^*(\mathbf{f}(\mathbf{x}, \mathbf{u}), n+1)\right],$$
$$\mathbf{u}^*(\mathbf{x}, n) = \arg\min_u \left[g(\mathbf{x}, \mathbf{u}) + J^*(\mathbf{f}(\mathbf{x}, \mathbf{u}), n+1)\right].$$

Assuming only that $J^*(\mathbf{x}, n)$ is smoothly differentiable, compute the update for $\mathbf{u}^*$ analytically in terms of $\mathbf{H}, \mathbf{C}, \mathbf{G}, \mathbf{B}, \mathbf{Q}, \mathbf{R}, J, \frac{\partial J}{\partial \mathbf{x}}$ and dt (your answer should not have any min *in it).*

c) Using your answer from part (b), compute the continuous-state, continuous action update

$$J(\mathbf{x}, n) = ?.$$

Again, your answer should be in terms of $\mathbf{H}, \mathbf{C}, \mathbf{G}, \mathbf{B}, \mathbf{Q}, \mathbf{R}, J, \frac{\partial J}{\partial \mathbf{x}}$ and dt, and should not have any min *in it.*

Note: *The final step, which we omit here for brevity, is discretizing $\mathbf{x}$ into a finite set of states $\mathbf{s}$, and converting the update equations appropriately. The result is a numerical variant of dynamic programming that (for some problems) avoids discretizing the action space.*

6.832 Underactuated Robotics
Spring 2009