Value Iteration (Double Integrator)

Value Iteration (Double Integrator)

0.0/30.0 points (graded)

In this problem, we'll consider the optimal control problem for the double integrator (unit mass brick on ice), described by

$$\ddot{q} = u, \ |u| \leq 1$$

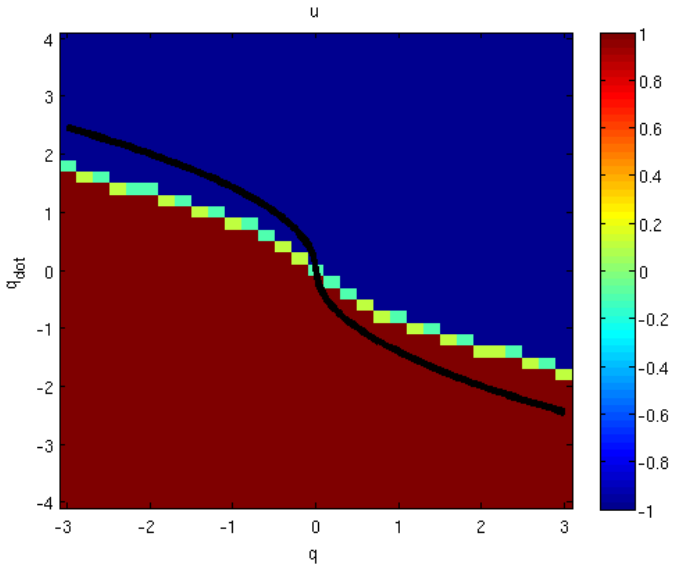
using the Value Iteration algorithm. An implementation of that algorithm is available for you in Drake (see the runValueIteration function in examples/DoubleIntegrator.m). This is a complete implementation of the algorithm with discrete actions and volumetric interpolation over state.

(a) Run the value iteration code for the double integrator to compute the optimal policy and optimal cost-to-go for the minimum-time problem. Compare the result to the analytical solution we found in lecture (also available in Example 9.2 in the course notes) by answering the following questions.

1) Find an initial condition of the form $(2, \dot{q}_0)$ such that the value iteration policy takes an action in exactly the wrong direction from the true optimal policy. Type in your value of $\dot{q}_0$ below:

Answer: -1.7

Explanation



The controller obtained from value iteration is plotted above. The black curve is the switching surface for the analytically derived optimal controller (i.e., where the control input switches from -1 to +1). The value iteration controller has the wrong sign at all points lying in the "band" between the switching surface for the analytical controller and the switching surface for the value iteration controller. One such point is $(2, -1.7)$.

2) What is the true optimal time-to-go from this state (i.e., for the optimal bang-bang controller derived in class)?

Answer: 2.0

Explanation

We derived the optimal cost-to-go function in class. For our point $(2, -1.7)$, the optimal cost is given by $2\sqrt{(\dot{q}_0^2)/2 + q_0} + \dot{q}_0$, which is approximately 2.01.

3) What is the time-to-go from this state estimated by value iteration?

Answer: 4.1390

Explanation

This can be read off from the plot generated by the value iteration code (the "index" field in the cost-go-go plot). It is approximately 4.14.

4) When implementing value iteration, one needs to be wary of several implementation details. Find a setting of the discretization (i.e., the variable xbins in DoubleIntegrator.m) that causes the code to NOT converge. The maximum distance between points in the q and $\dot{q}$

directions should still be at most 0.2, and the grid must still contain the square with sides of length 2 centered about the origin. (Hint: it might help to see how the minimum-time cost function is implemented).

```
q_bins =
qdot_bins =
xbins = {q_bins,qdot_bins};

```

Unanswered

```
% Any solution where the grid does not contain the origin will cause
% the value iteration to not converge. This is because the cost-to-go
% is never 0 at any point. So, one possible answer is:
q_bins = linspace(-3,3,40);
qdot_bins = linspace(-3,3,40);
xbins = {q_bins,qdot_bins};
```

Run Code

(b) Change the cost-to-go function to a combination of the quadratic regulator problem and the minimum-time problem:

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

where $c(q, \dot{q})$ is 0 when $(q, \dot{q}) = (0, 0)$ and 1 otherwise. Use $Q_p = Q_d = 1, R = 10$.

1) What is the cost-to-go from the point $(1.0, 1.0)$ estimated by the value iteration?

Answer: 31.14

Explanation
This can be read off from the plot generated by the value iteration code (the "index" field in the cost-go-go plot). It is approximately 31.14.

2) When implementing controllers on real robots, we have to be mindful of the fact that our models seldom capture all aspects of the behavior of the system. Supposing that in addition to the constraints imposed on the maximum and minimum control input (i.e., $|u| \leq 1$), our real physical brick "robot" also had constraints on the derivative of the control input (let's say $|\dot{u}| \leq 1$). Assuming that you only cared about stabilizing the system to the origin, which controller would you prefer to implement?

- Minimum-time (cost from part (a))
- Quadratic cost + minimum-time (cost from part (b)) ✔

Explanation
As discussed in class, the minimum-time cost typically yields controllers that change very sharply from -1 to 1 along the switching surface. This would violate the constraint $|\dot{u}| \leq 1$. The quadratic cost adds a penalty to the actions and causes them to get smoothed out (as you can see from the controller found by value iteration in part (b)).

Submit You have used 0 of 3 attempts

Answers are displayed within the problem