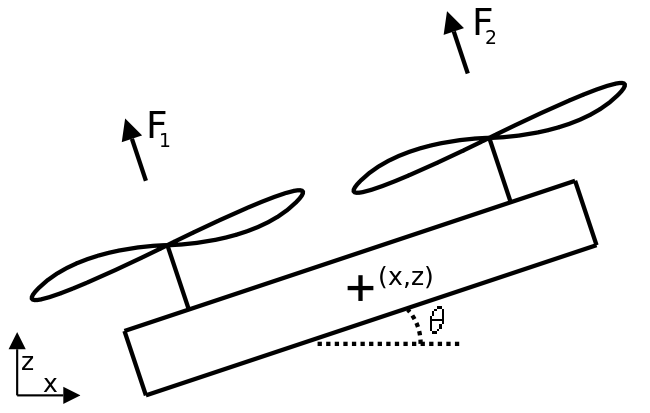
**Problem: Definition of Underactuated**

 Bookmark this page

Definition of Underactuated

The following problem explores the definition of underactuated as described in the lecture notes. You should not need to derived detailed equations of motion for any of these problems.

**Helicopter**



A helicopter with two rotors is constrained to move in a vertical plane. Assume gravity acting on the helicopter. The task is to control the position and pitch  by varying the thrust produced by the two rotors. Decide whether this system is fully-actuated (in all states), or if there are any states in which the problem is underactuated. Use the definition of underactuated provided in lecture.

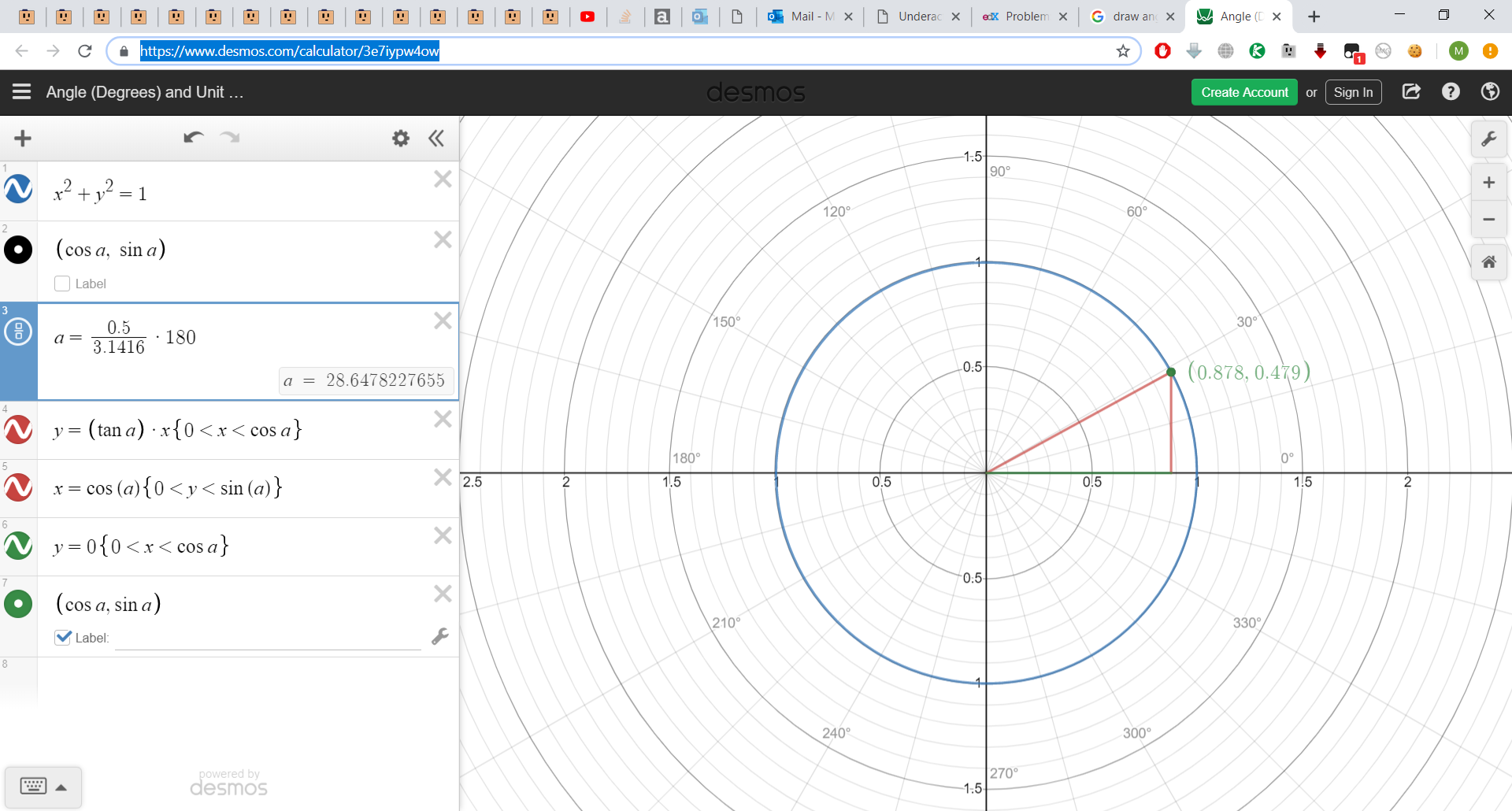
**Underactuated -> there are fewer actuators than degrees of freedom.**

If you said "Underactuated" above, then please provide an expression for an acceleration that cannot be instantaneously achieved by the system. Assume that  and  are unbounded (and can be negative) and that the current state of the system is  radians. Your answer should consist of three **numerical values**, in the order

**xdd**

**ydd**

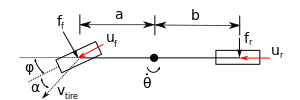
**thetadd**



# **If we want to accelerate in x and z while keeping theta = 0.5, we can’t rotate the angle theta simultaneously.**

**A control system is underactuated in state (q,q˙) at time t if it is not able to command an arbitrary instantaneous acceleration in q.**

**Bicycle Model**



Consider the simple model of a vehicle known as the bicycle model, illustrated above. Let  be the position of the vehicle in inertial coordinates and  be the heading angle. Lateral tire forces are typically modeled as being proportional to the lateral slip angle , which defines the angle between the angle of the tire and the velocity of the tire , which depends on the speed and angular velocity of the vehicle, giving .

Generating the equations of motion can be tedious, and we will often use a software package to do it automatically. For this problem, we have done it for you--but, as is often the case, the equations are pretty messy! One way to write the system of equations is:

For the purposes of this problem, assume that the driver has control over the steering angle  and has rear wheel drive. Treat the drive torque as a simple ground reaction force  acting at the tire and let . Is this system is fully-actuated or underactuated? Explain.

**Underactuated -> there are fewer actuators than degrees of freedom.**

Now, suppose the the driver has control of both the front and rear longitudinal tire forces  and  and so has 3 total control inputs. In general, do you think this system is fully-actuated or underactuated? Give an intuitive explanation.

**Underactuated**

**For example, we can’t move the vehicle to its right or left side without changing it’s direction.**

**This vehicle model is still underactuated, despite that there are three actuators. Intuitively, the  
rotation of the vehicle and lateral translation are intrinsically coupled--for instance, the car  
cannot translate sideways without rotating.**

Since these dynamics are not control affine, consider a simplified system of equations linearized about  and . For simplicity, without loss of generality, let . Recalling the general dynamics form,

 We can write:

Find the rank of  when .

****

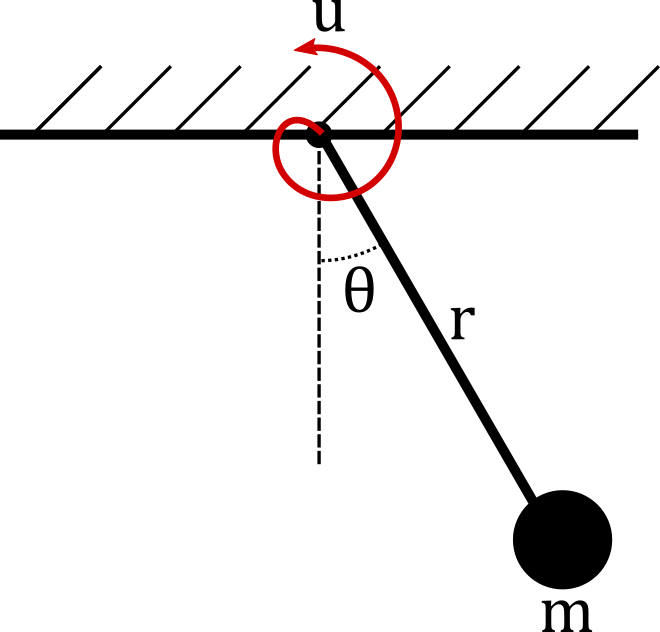
Is the system fully-actuated or underactuated?

**Underactuated**

Are there values for  for which the system is fully-actuated? If so, write a symbolic expression of the form  where  describes these states. If there are no such states, enter 0. For grading purposes, ensure that . For example, if the system is fully-actuated whenever , write .



**Pendulum Model**



Consider a pendulum with length , mass , and a single degree of freedom, . Let  be the torque applied to the pendulum, which will be our input to the system. Which of the following is/are true? (if none are true, then don't check any boxes).

**b. The system is underactuated if u is bounded.**

**If  is unbounded, then for any acceleration  we can find a corresponding torque  to produce that acceleration. That means that the pendulum is fully actuated when  is unbounded, regardless of the value of , so choices a, b, and d are all incorrect.**

**If  is bounded there will exist accelerations  that cannot be produced without violating the bounds on . That means that the pendulum is underactuated when  is bounded. Thus only choice b is correct.**

### Feedback Linearization

True or false: for any underactuated system of the form , one can choose  so that , where  is a new control input.

**False**

**Only when f2 is full row rank, it is invertible.**

Take a robot whose dynamics are given by the manipulator equations,  for . The robot starts in a given initial configuration  and with a given initial velocity . Suppose  is rank  for all . Which of the following statements are true for **any** twice-differentiable desired trajectory ?

1. **Feedback linearization can be used to make it so that  for all**

****

**In other words, if f1 and f2 are known and f2 is invertible, then we say that the system is "feedback equivalent" to q¨=u′q¨=u′.**

**Feedback linearization of a second-order system can create arbitrary accelerations ,  
however, it cannot change the simple fact that we are dealing with a second-order system.  
Position and velocity are still limited in important, fundamental ways--particularly, both must  
be differentiable with respect to time. On the other hand, has no such restriction, it can,  
and often will be, discontinuous in time.**

## Oscillating Pendulum

Consider an actuated pendulum, where the base is forced to oscillate in simple harmonic motion, . Then, the dynamics of the pendulum angle  are:

Even with the base shaking, we would like the pendulum to spin at a constant speed, . To achieve this, we should choose  to stabilize any velocity error. Use feedback linearization to find the control law such that . Write the variable  for .

Trigonometric functions and greek letters can be written out in English, and should format properly. For example, simply write "sin(omega\*t)" to form

**L^2 m ((C(ω)^2 sin(θ) sin(t ω))/L - (g sin(θ))/L - t\_d + 1) = u**

### Nonlinear Dynamics

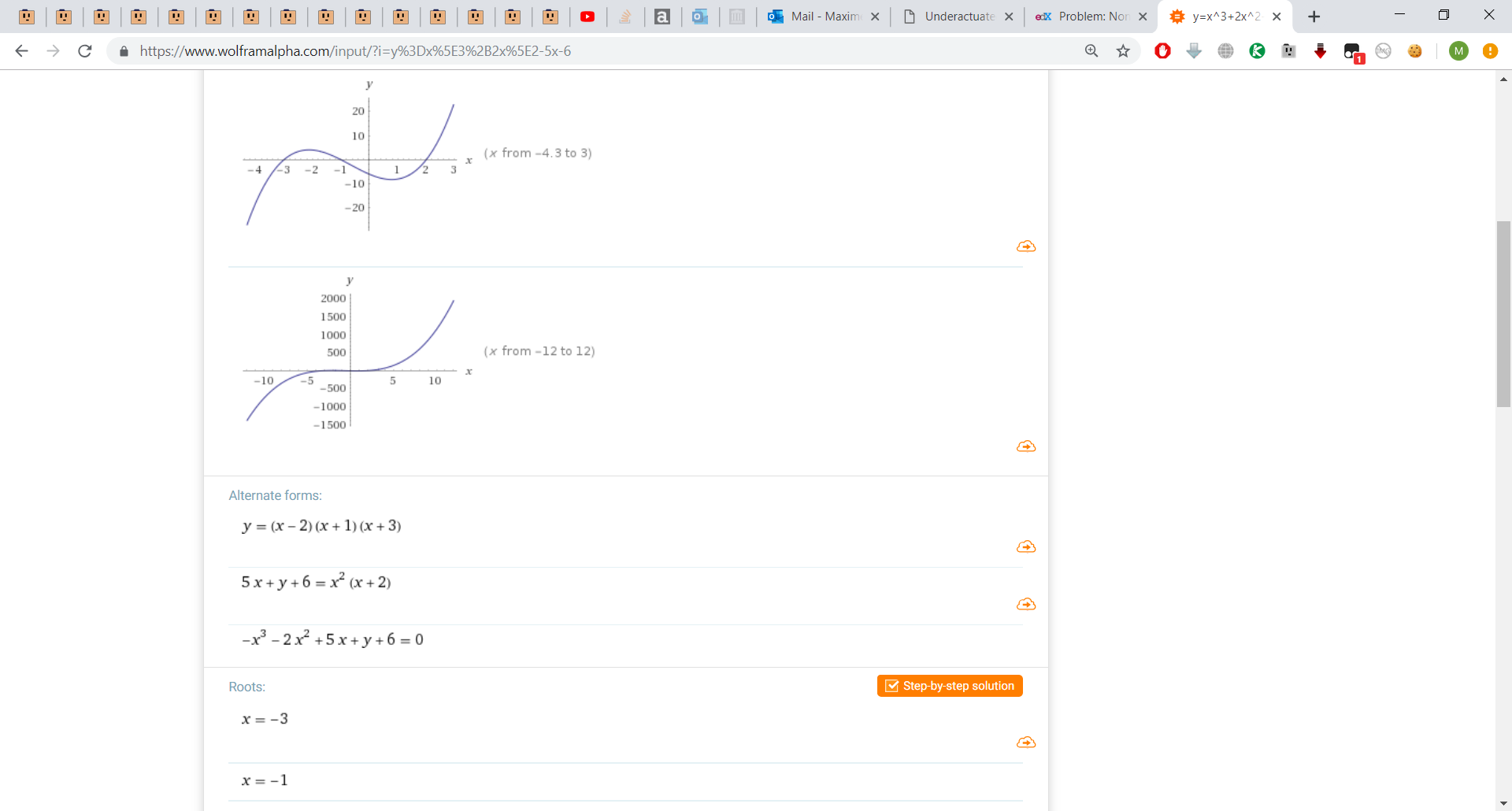
0.0/20.0 points (graded)

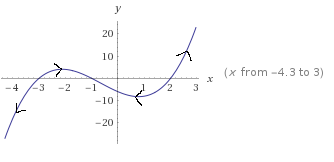
Consider a system with the dynamics given by

Fill in the MATLAB code below, which is supposed to:

* Plot a phase diagram  vs.  for this system and set the three equilibrium points. Ensure that all equilibrium points are included in the plot range.
* Set the variable "eq\_points" such that  is an equilibrium point.

Do not change the variable names x, xdot, and eq\_points.





Is the first equilibrium point (at the smallest value of ) stable, unstable, or marginally stable?

**Unstable**

Is the second equilibrium point stable, unstable, or marginally stable?

**Stable**

Is the third equilibrium point (at the largest value of /(x/)) stable, unstable, or marginally stable?

**Unstable**

There is an interval, containing the origin, that is a region of attraction for one of these points. Identify this interval, using standard notation of  for open intervals and  for closed intervals. Indicate a interval extending to infinity with "-inf" or "inf."



### Discrete Systems

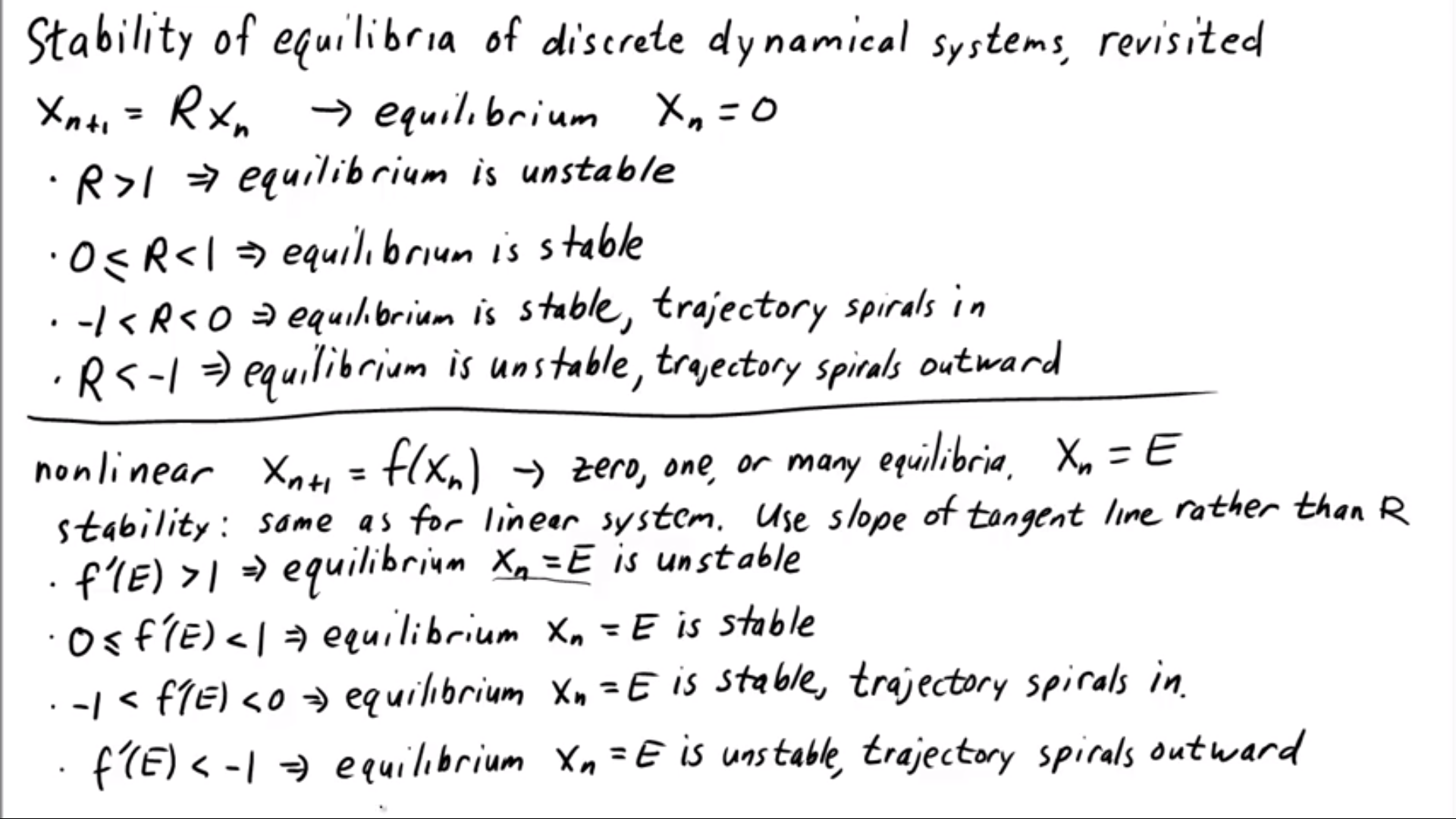
0.0/20.0 points (graded)

For a univariate dynamic system  we have seen via graphical analysis that  is a locally stable equilibrium if the following conditions hold

Otherwise stated, that  has a zero-crossing at  with negative slope.

Now, consider a simple discretization of this continuous system, where for some fixed time step  we have:

For arbitrary , the two conditions above are *not* sufficient for stability of the discrete system. Provide a counterexample demonstrating this by giving values for , , and  below.



**syms x**

**x\_star = 0**

**h = 1**

**f = -x^2**

**% df/dx= -2x**

**% 0.1+1(-0.01) stable but df/dx is not < 0**

**% One possible solution is described  
syms x  
% pick a favorite system which is stable in the continuous sense  
% any linear system will do. We'll also take the origin to be  
% the equilibrium, for simplicity.  
x\_star = 0  
f = -x  
% now, let's try some values for h and see what happens  
h = .1;  
% simulate 100 steps, from x\_0 close to 0  
x\_sim(1) = .01;  
for i=2:100,  
x\_sim(i) = x\_sim(i-1) + h\*-x\_sim(i-1); % our discrete update rule  
end  
% x\_sim(10) = 2.9e-7, so it looks like the discrete system is also stable  
% try again with a much larger h\*-x\_sim  
h = 10;  
x\_sim(1) = .01;  
for i=2:100,  
x\_sim(i) = x\_sim(i-1) + h\*-x\_sim(i-1); % our discrete update rule  
end  
% Now x\_sim(100) = -2.9e92, a massive number, so it's safe to say that this is unstable**

Find the upper bound  such that all  results in a stable discrete system. Write your answer in terms of , where

**Using graphical analysis for one dimensional continuous systems, we said that for an equilibrium to be stable, the dynamic flow had to point toward the equilibrium. Put another way, that the dynamics had to lead the system *closer* to the equilibrium. We saw this graphically by drawing arrows along the axis. Part 1 of this problem illustrated that if $h$ is too large, the system can become unstable. What does it mean for the discrete system to get closer? It is, surprisingly, easier to formulate this criteria in the discrete case. Closer just means that . For simplicity, we'll assume $x^\*=0$. Substituting and simplifying, we get:**

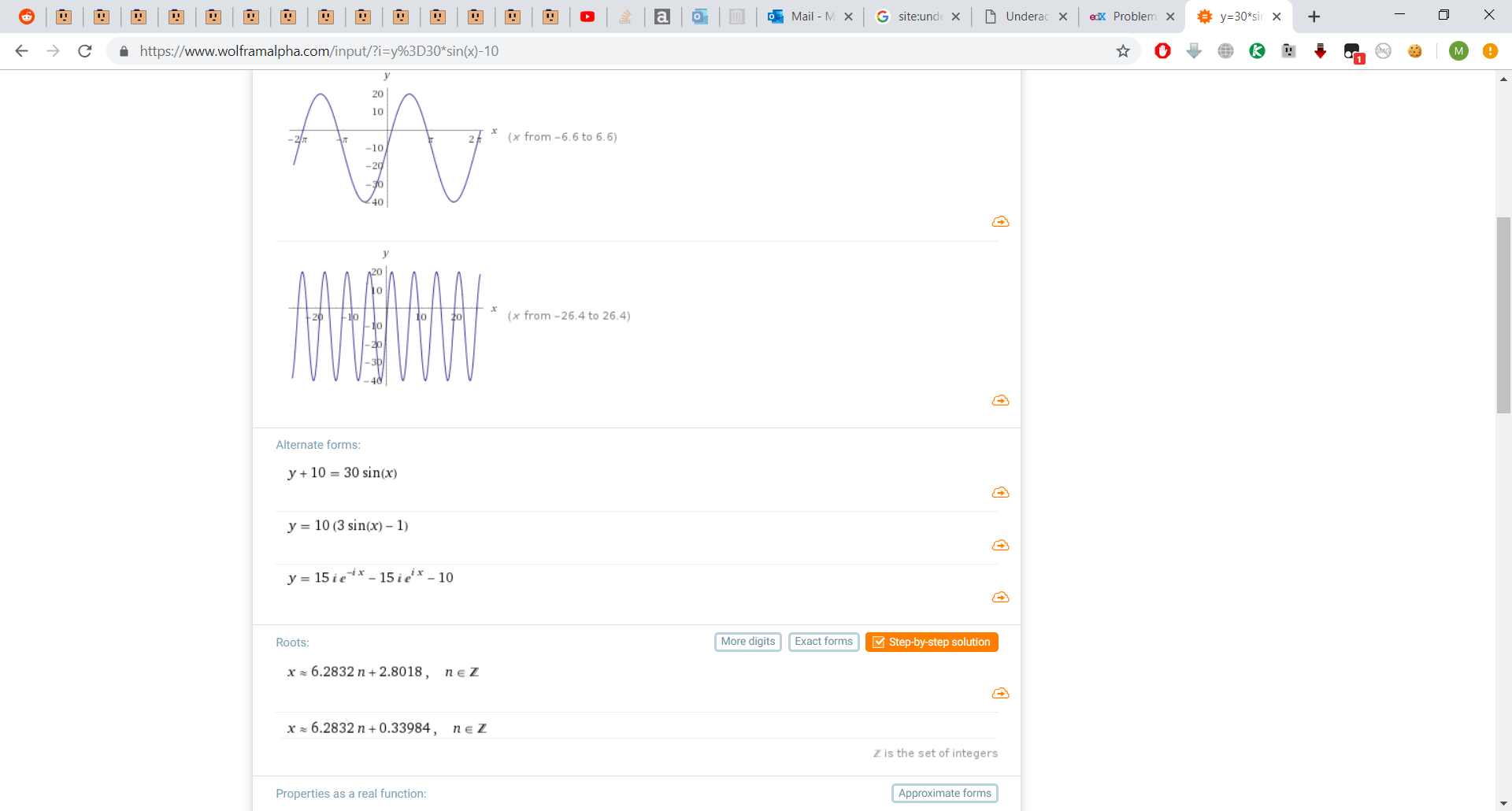
**This must hold as , so by linearization of  or through Taylor expansion, the right hand side simplifies to . The Taylor expansion argument is given below:**

Simple Pendulum

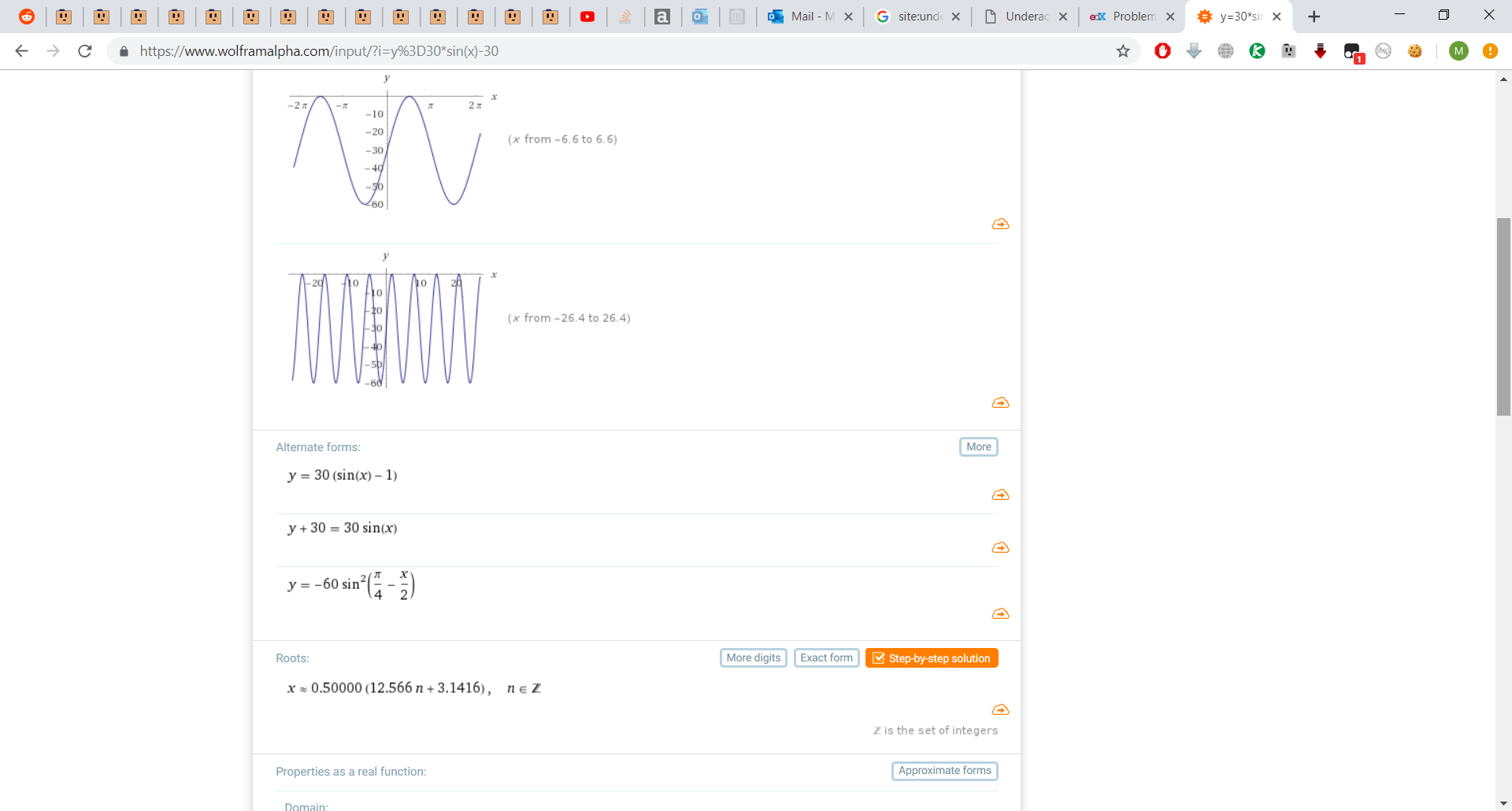
The lecture introduced the simple pendulum as a benchmark nonlinear system. Recall that the second-order dynamics of a damped pendulum are

Consider the case where the control input  takes on a constant value. Take the constants , , , and . Plot (but do not submit) the bifurcation diagram  showing the equilibrium point(s) for a fixed . Note what happens when  increases to 30 and above.

For , provide the equilibrium point(s) as a comma-separated list . Ensure that the number of equilibrium points is correct (no duplicate entries). The error tolerance for each element in the list is . Restrict your answers to the interval .

0.33984, 2.8018

Do the same for .

pi/2

Cost functions (Part a)

In this problem we will explore how to design cost functions that make the robot exihibit the kind of behavior we want. For this, we will consider the Dubins car model, which is a very simple model of a vehicle given by the following equations:

where  is the state of the system and consists of the states  (the x-position),  (the y-position) and  (the yaw angle of the vehicle). The control input is .

(a) For optimal control problems, it is often useful to have cost functions of the form:

where the first term depends only on where the robot ends up (the second term is simply the additive cost structure we saw in lecture). Here, is the final time and  is a symmetric positive semidefinite matrix. Suppose we want the robot to end up with its yaw angle close to 0, but do not care about the final  and  positions. What should we choose  to be (remember to make sure it is symmetric and positive semidefinite)?

1

**Qf = [0 0 0;0 0 0;0 0 1];**

**% Since we don't care about the final x and y positions,  
% Qf must have zeros everywhere except the (3,3) element.  
% Since we want to penalize deviations of the yaw angles  
% from 0 (and since our Qf must be p.s.d.), the (3,3) element  
% must be positive. So, one possible answer is:**

Cost functions (Part b)

0.0/5.0 points (graded)

(b) Now suppose we want the vehicle to end up close to the line , but we do not care exactly where on this line and what yaw angle it ends up in. What should we choose  to be (remember to make sure it is symmetric and positive semidefinite)?

**Qf = [1/4 -1/2 0; -1/2 1 0; 0 0 0]; simplify([x y z] \* Qf \* [x; y; z]-(0.5\*x-y)^2)**

**ans =**

**0**

**% Since we don't care about the yaw angle, the third row and column  
% must be all zeros. Hence, we only need to figure out the top left 2 by 2  
% block. One possible cost function x'\*Qf\*x for penalizing deviations from  
% the line y = 0.5 x is (y - 0.5x)^2. Thus, we can make x'\*Q\*x match this  
% function by picking the top left 2 by 2 block correctly. We find that the  
% following Qf achieves this**

Cost functions (Part c)

0.0/5.0 points (graded)

(c) Now suppose we want to end up close to the curve , and again do not care about the final yaw angle or where exactly on this curve we end up. Is it possible to achieve this given our setup?

**No**

**If there is no zero eigenvalue, the origin is the only equilibrium. If there is at least one zero eigenvalue, there are infinitely many equilibria, and they form a straight line or the whole plane.**

* **A line of stable equilibria (a negative eigenvalue and a zero eigenvalue)**
* **A line of unstable equilibria (a positive eigenvalue and a zero eigenvalue)**
* **A line of unstable equilibria and other solutions are parallel to this line (two zero eigenvalues with only one linearly indepedent eigenvector)**
* **A plane of equilibria (two zero eigenvalues with two linearly indepedent eigenvector)**

**One way to see it is to notice that  can only be 0 along eigenvectors corresponding to 0 eigenvalues. Hence, it can only be 0 at the origin, or along straight lines, or along a plane. Thus, it cannot be 0 on the curve .**

**Another way to see that this is not possible is to notice that  is a (positive semidefinite) quadratic function of . Level-sets of positive semidefinite quadratic functions are either ellipsoids or straight lines (which are just "degenerate" ellipsoids). However, we want the cost to be 0 along the curve  (which does not define an ellipsoid or a straight line).**

Linear Optimal Control

Consider the scalar equation

and the infinite horizon cost function

(a) Assume that the optimal cost-to-go function is of the form . What value of  satisfies the Hamilton-Jacobi-Bellman conditions for optimality?

**g=32x^2+u^2**

**dJ/dx=2\*p\*x**

**dJ/du=0**

**d/du(**

**g+dJ/dt=32x^2+u^2+2\*p\*x\*(-4\*x+2\*u)**

**)**

**2\*u+4\*p\*x=0**

**u=-2\*p\*x**

**J\* = 32x^2 + (-2\*p\*x)^2 + 2\*p\*x\*(-4\*x+2\*(-2\*p\*x))**

**=-4\*x^2\*(p^2 + 2\*p - 8) = 0**

**True for all x -> -4\*(p^2 + 2\*p - 8) = 0. P = -4, 2**

**P must be positive definite -> p=2**

What happens in the case where our system is not control affine or if we really do need to specify an instantaneous cost function on uu that is not simply quadratic? If the goal is to produce an iterative algorithm, like value iteration, then one common approach is to make a (positive-definite) quadratic approximation in uu of the HJB, and updating that approximation on every iteration of the algorithm.

(b) Given that the optimal feedback controller associated with  is , what is the value of ?

4

(c) Suppose we change our cost to the following:

Which of the following statements is true? (Select all that apply)

The optimal controller (K) gets multiplied by 3

The optimal controller (K) gets divided by 3

The optimal cost-to-go gets multiplied by 3

**g=96\*x^2+3\*u^2**

**dJ/dx=2\*p\*x**

**dJ/du=0**

**d/du(**

**g+dJ/dt=96x^2+3\*u^2+2\*p\*x\*(-4\*x+2\*u)**

**)**

**6\*u+4\*p\*x=0**

**u=-2/3\*p\*x**

**J\* = simplify(96\*x^2+3\*(-2/3\*p\*x)^2+2\*p\*x\*(-4\*x+2\*(-2/3\*p\*x)))**

**= -(4\*x^2\*(p^2 + 6\*p - 72))/3 = 0**

**True for all x -> (p^2 + 6\*p - 72) = 0. P = -12, 6**

**P must be positive definite -> p=6**

**The optimal cost-to-go gets multiplied by 3, but the controller does not change. This can be easily verified by going through the same calculations as in part (a) and (b). This is a feature of LQR (Linear Quadratic Regulator) problems in general. Scaling both the cost on state and action does not change the controller (but only changes the optimal cost-to-go).**

### Value Iteration (Double Integrator)

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

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  such that the value iteration policy takes an action in exactly the wrong direction from the true optimal policy. Type in your value of  below:

****

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

****

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

****

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  and  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).



**xbins = {[-2.1:.2:2.1],[-2.1:.2:2.1]};**

**% 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:**

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

where  is 0 when  and 1 otherwise. Use .

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



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., ), our real physical brick "robot" also had constraints on the derivative of the control input (let's say ). Assuming that you only cared about stabilizing the system to the origin, which controller would you prefer to implement?

**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 . 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)).**