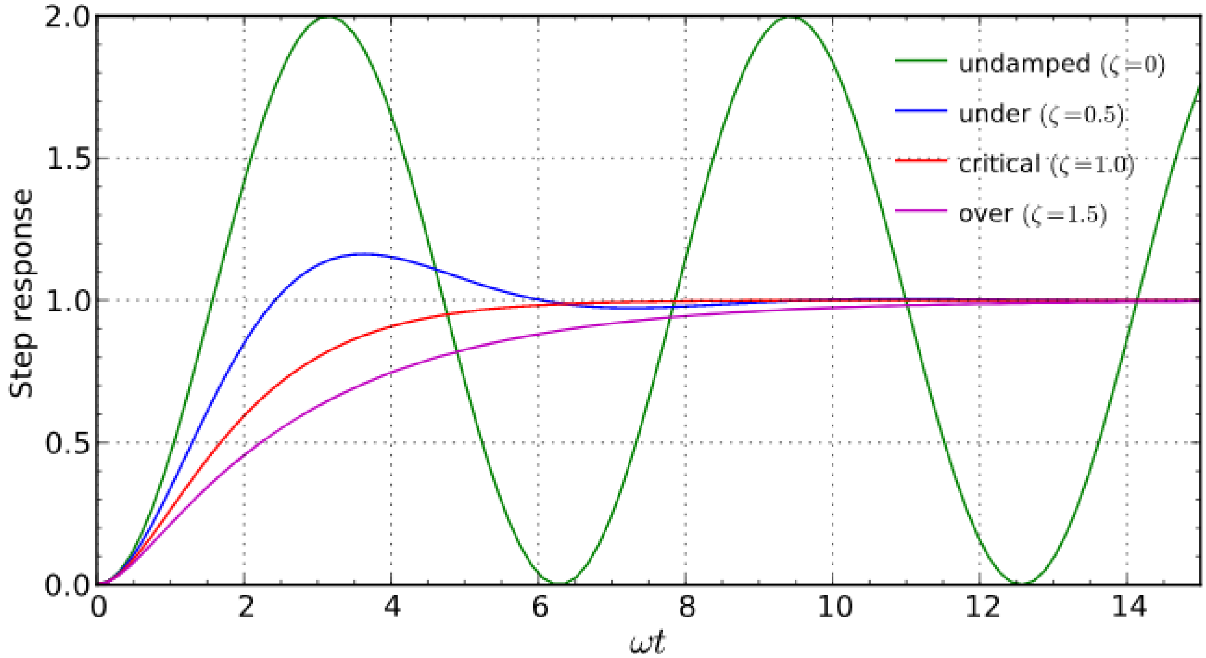
FL. 2018 ESE 447.02 Robotics Lab

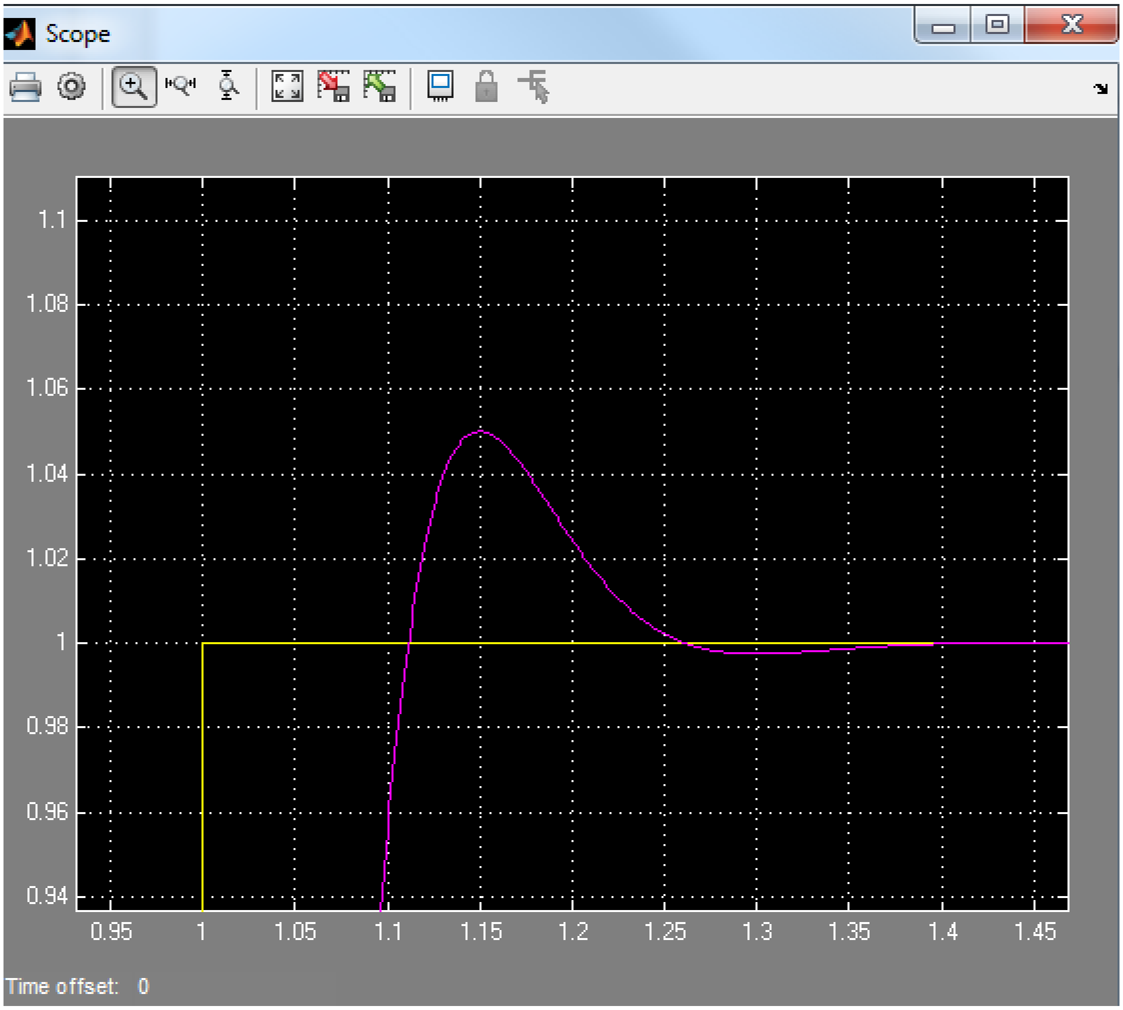
Lab Journal 18

11/27/2018

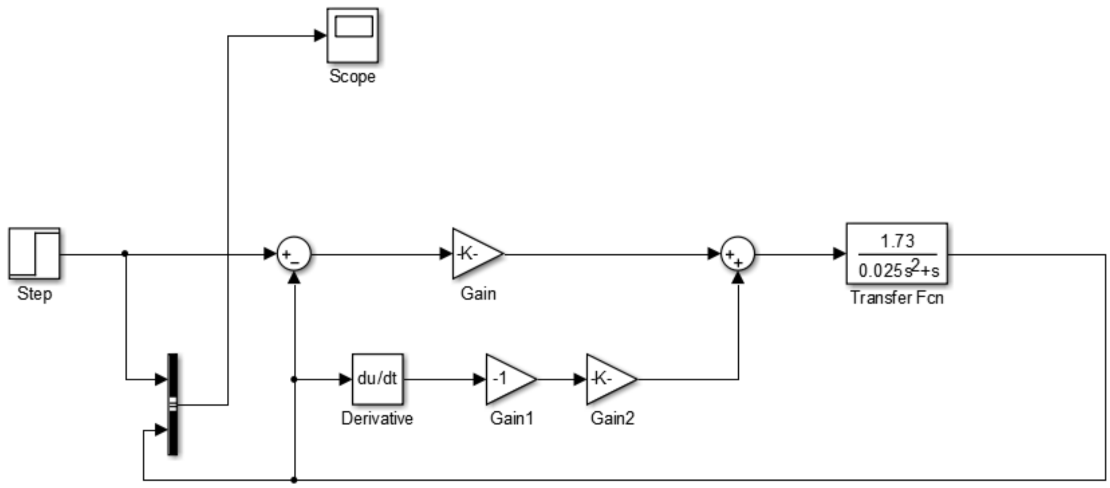
1. Back onto the swing up controller.
   The control system used for the swing-up controller was identified as a second order system.The general expression of transfer function of a second order control system is given, and the dynamic behavior of the second order system is described in terms of two parameters: the damping ratio and the natural frequency. If the damping ratio is between 0 and 1, the system poles are complex conjugates and lie in the left-half s plane. The system is then called underdamped, and the transient response is oscillatory. If the damping ratio is 1, the system is called critically damped, and when the damping ratio is larger than 1 we have overdamped systems.



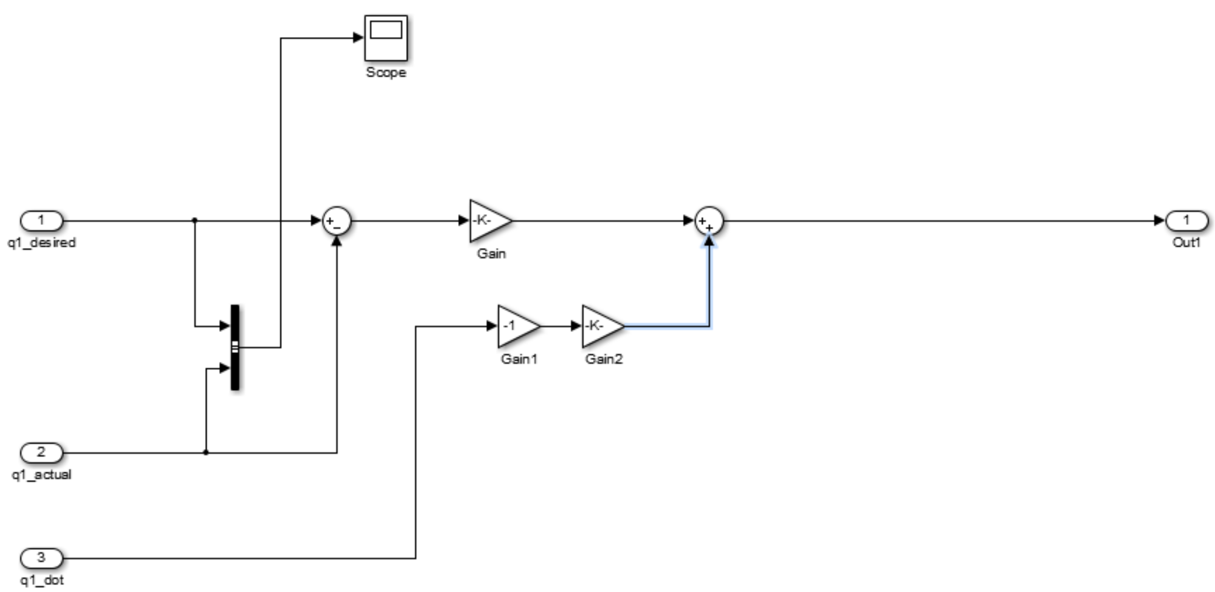
We derived our Kv and Kp values so that the system response looks like this with a 5 % overshoot:



Implementing the gains, the separate swing up controller was built as follows:



This model takes in a step input and gives a step response based on the transfer function, where the transfer function comes from the real system we were using. However to use the swing up controller for our simulated robot system, we need to get rid of the transfer function and combine the swing up controller with our whole system.
Recall that the purpose of the swing up controller is to swing q2 up by controlling q1, and q1 is a function of q2 and q2_dot. Therefore the swing up controller would give an input of the desired q1 and output the voltage needed to control the motor. Therefore when combining the controller to the larger system, we replaced the transfer function and the feedback with our actual feedback q1 and q1_dot, and used the controller as a subsystem shown below:



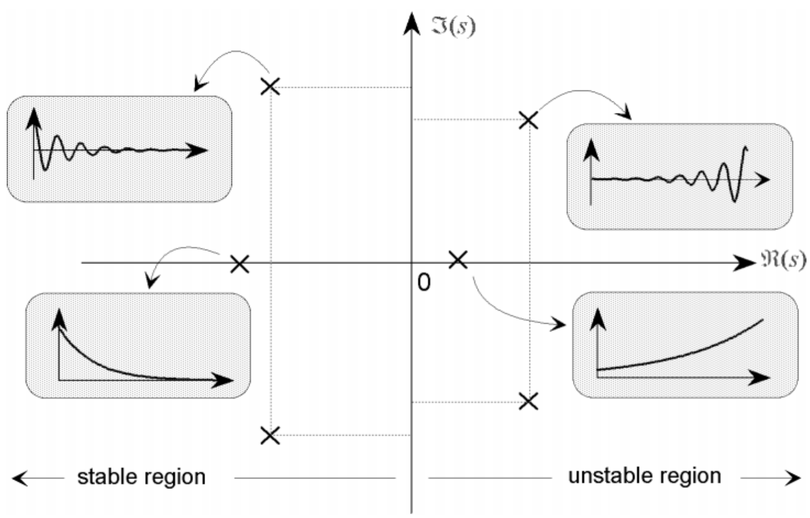
The output is then fed to the first port of the multi-switch block as the swing up signal.

2. Create the Balance controller using the k values we got from previous lab.
   As stated in the previous lab, the balance controller is to be achieved by using a state-space closed-loop control approach.

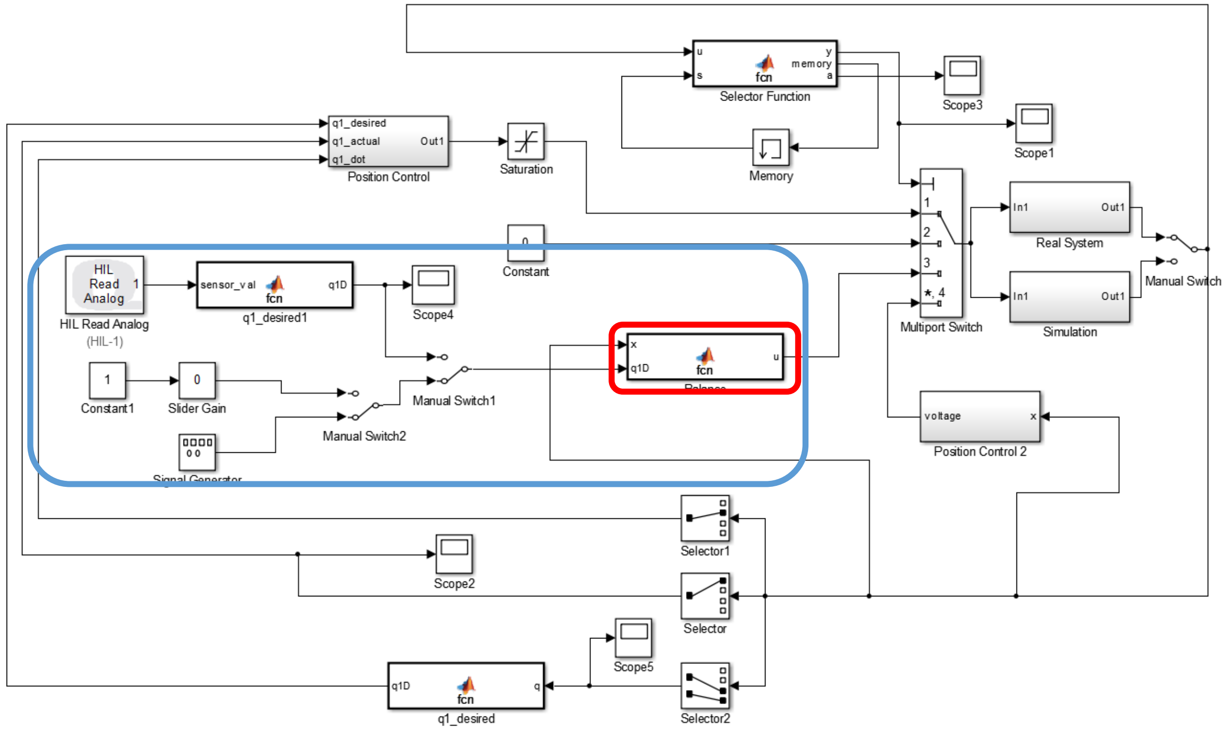
   The system was linearized to the form

   $$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$

   where x-dot is the state vector, A is the state matrix, and B is the input matrix. The method for feedback for the system was to multiply the output by a matrix K and setting u(t)=Kx as the input to the system. The values of the poles we used were real negative numbers and they defined a proper decay performance in the homogeneous response. In order for a linear system to be stable, all of its poles must have negative real parts, that is they must all lie within the left-half of the s-plane. The K matrix was put into the balance function and the closed-loop was then formed. The closed-loop control system will therefore control the four state variables based on different gains for each variable.



   We have already gotten the constants for the k parameter as
   k = [-0.9825, -1.3165, -15.6590, -2.2058]
   We just need to feed back the u=-kx to the input to form the closed loop, where x is the vector of state variables.
   Circled below in blue is the part for the balance function, and in red is the main balance code, the rest in the blue circle is different inputs for desired q1.



The code in the function block is as follows:

```
function u = fcn(x,q1D)
q1A = x(1);
q2 = x(3);

q2 = atan2(sin(q2),cos(q2));
x(3) = q2;
k = [-0.9825 -1.3165 -15.6590 -2.2058];
x(1)

x(1) = q1A-q1D;
x(1)
u = -k * x;
```

This code uses the difference of actual q1 and desired q1 as the variable to be controlled, therefore it can affect the balance state to make link1 rotate to a desired position while keeping link2 balanced.