RENSSELAER MECHATRONICS

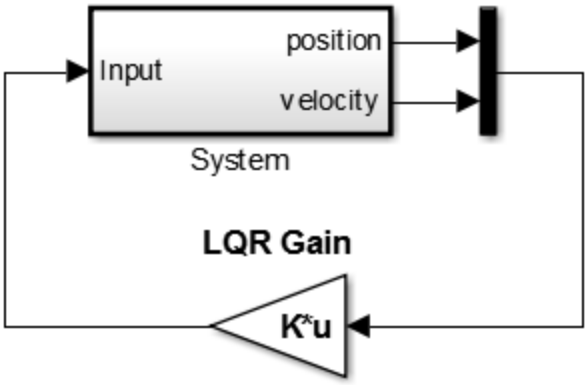
Basic Balancing Lab

Objectives:

- Balance the MinSeg using a full state feedback controller, in particular a linear quadratic regulator or LQR

Background Information:

Most single input, single output (SISO) systems can be adequately analyzed and controlled using basic transfer function techniques and PID type controllers. For systems that have multiple inputs and multiple outputs (MIMO) a more suitable type of control framework is called full state feedback. A full state feedback controller measures or estimates all of the system "states" and uses this "state" multiplied by a gain matrix to control the system. For a simple one degree of freedom mechanical system the "states" of the system could be the position and velocity. Typically states of the system can be those variables that are used to describe the energy of the system. For mechanical systems this would be kinetic and potential energy, which correspond to velocity and position "states".



A full state feedback controller has the control law $u = Kx$. For a mechanical system with one input with two states like position and velocity the control law is:

$$u = Kx = [k_1\ k_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = k_1 x_1 \quad k_2 x_2 = k_1 \ position \quad k_2 \ velocity$$

In other words the input is the sum of the position and velocity, each scaled by a gain. We could also write this as:

$$u = \quad k_1 \quad position \quad k_2 \quad \frac{d}{dt}(position)$$

And in this way we see it applies a input proportional to position and proportional to the derivative of position, or it *appears* as a proportional derivative controller (PD controller).

A LQR (linear quadratic regulator) controller is a special type of feedback controller that chooses the gain matrix K in an optimal way to minimize controller effort and state error. This is the type of control that will be used to balance the MinSeg.

The general steps to implement a LQR controller are:

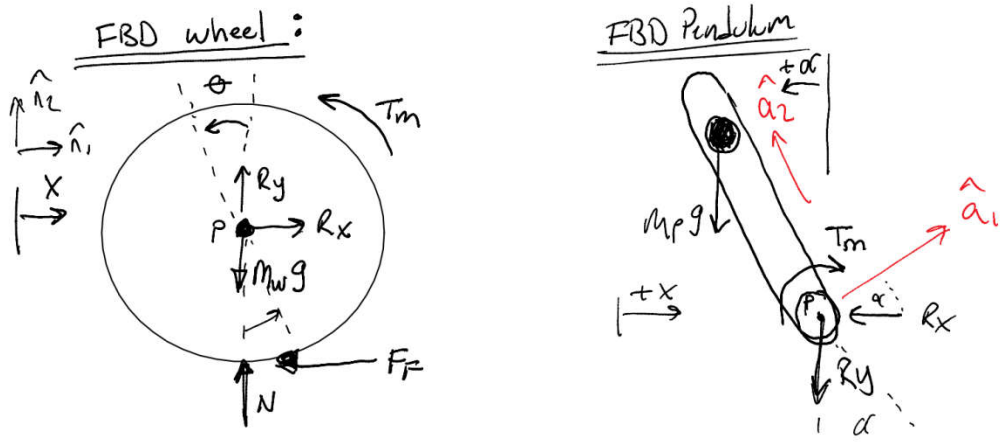
- Model the system and put the equations of motion in state-space form
- If the model is nonlinear linearize it about the operating point to obtain state space matrices A, B
- Choose weighting matrices R and Q. R is the weighting for the input – it describes how important it is to minimize the amount of effort used to control the system. Q is a weighting matrix for the states – it describes the importance it is to minimize the error in each state.
- Use the Matlab command K=lqr(A,B,Q,R) to compute the controller grain matrix K. This is the "brain" of the system and requires the model knowledge A and B.
- Finally estimate the states of the system, form the state vector, and multiply this by K to obtain the input to the system

This lab will walk through these general steps for the MinSeg system

Calculation of Gain Matrix K

- Determine system matrices A and B. If they have been provided they can be loaded with the command `load MINSEG_AB`.
    - They system states for the MinSeg are
        - x – the position of the center wheel
        - x – the velocity of the center of the wheel
        - α – the angular positoin of the body (pendulum) part of the MinSeg
        - α – the angular velocity of the body (pendulum) part of the MinSeg

      These coordinates correspond to the following Free Body diagrams:



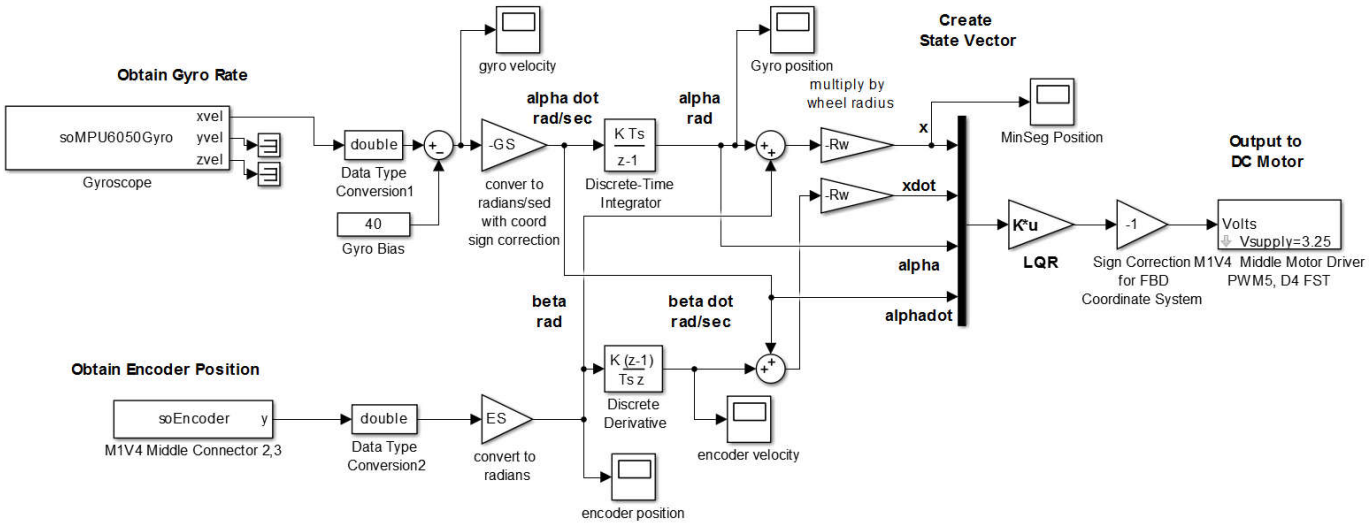
- Define weighting matrix for the input R. Since there is only one input R will be a scalar and it can be initially chosen as 1
- Define weighting matrix for the states. Initially weight all the states the same, but emphasize that keeping the system balanced ($\alpha, \dot{\alpha}$ states zero) is more important than minimizing the control input by making the weights 100 on each element corresponding to the pendulum angle

```
% load MINSEG_AB  % load system model matrices A, B
load Segway_AB

R=1; % weight for control voltage
Q= 100*[ 1 0 0 0;
         0 1 0 0;
         0 0 1 0;
         0 0 0 1;];  % weight each state by 100
KLQR=lqr(A,B,Q,R)
Rw=.0216;  % radius of wheel in m (small wheel)
```

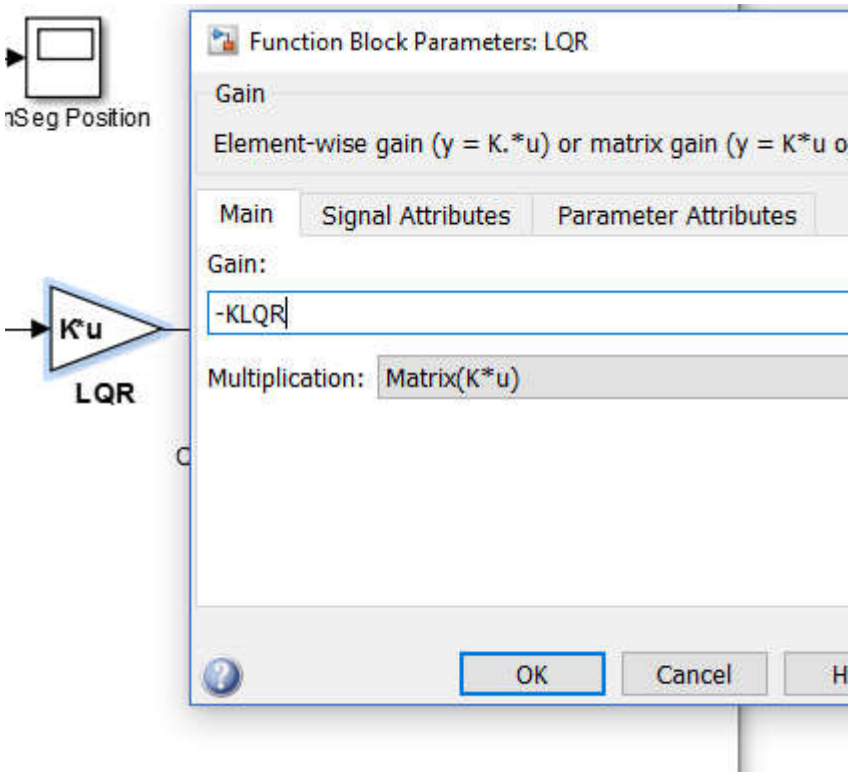
- Use commang lqr to compute the gain matrix
- Define the wheel radius – this is used to find x

Implementing the Controller in Simulink



Overview:

- Use an initial solver size of .03 seconds – so you can run the code in external mode if necessary
- Ensure all blocks that use sample times have the same sample time as the solver size
- Make sure the Discrete-Time Integrator has an inherited sample time of "-1", by default might be 1 second, which will not work.
- The angular rate of the pendulum α is obtained directly from the gyroscope and integrated to obtain α.
    - Make sure you use the correct Gyro bias! You will have to calculate this! You can do this in a separate diagram (like in lab 3).
    - This conversion constant GS is multiplied by a negative – this is to match the sign convention in the Free body diagram used to derive the model
    - You must choose the correct conversion constant to convert to radians/sec (see gyroscope lab)
- To obtain the positon of the center of the wheel x the wheel angle β must be added to the pendulum position α and multiplied by the wheel radius R_w:
    - $x = \quad (\beta + \alpha)R_w$
    - You will have to use the right conversion constant for the encoder!
- Finally create the state vector with the mux block and use the matrix multiply to get the command signal:

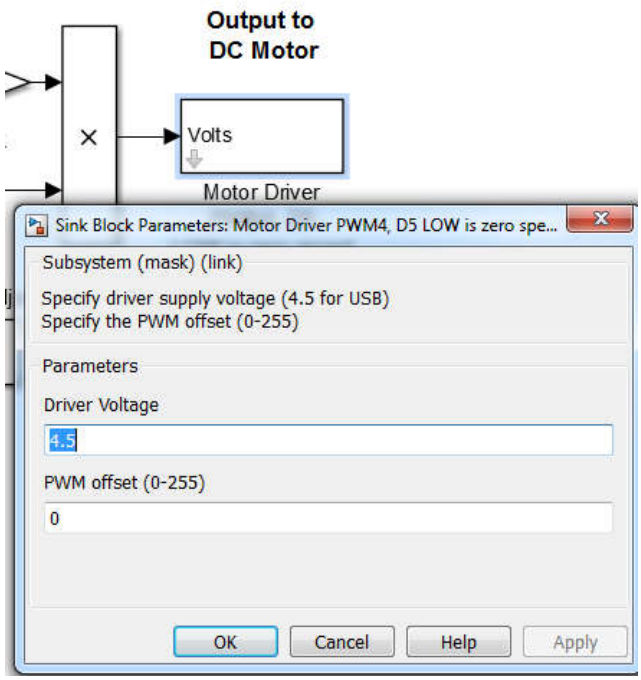


- The output is multiplied by -1 – this is to match the sign convention in the Free Body Diagram. You must verify all the signs in your system match the signs in the Free Body Diagrams.

Balancing the MinSeg

To balance the MinSeg:

- Download the code
- Hold the reset button, then place the MinSeg upright. Hold it in a place as balanced as possible.
- Release the reset button – the current position is “zero” and the controller will attempt to keep the pendulum there (If it works)
- The MinSeg should just barely be able to balance tethered with the USB cable
  - If batteries are added don’t forget to update the driver voltage in the driver output block



- If it won't stay upright try larger values on the weighting on Q
- If it is too jittery you can double click the Gyroscope block and set the value of DLFPmode to 5. This will create a low pass filter on the gyroscope that can "smooth" out the gyro signal which can sometime make the unstable.
- If they gyro is filtered and it is still to jittery, or doesn't' try hard enough you can try decreasing/increasing the overall value of the KLQR gain marix.

Questions:

Explain the purpose of every block in the block diagram