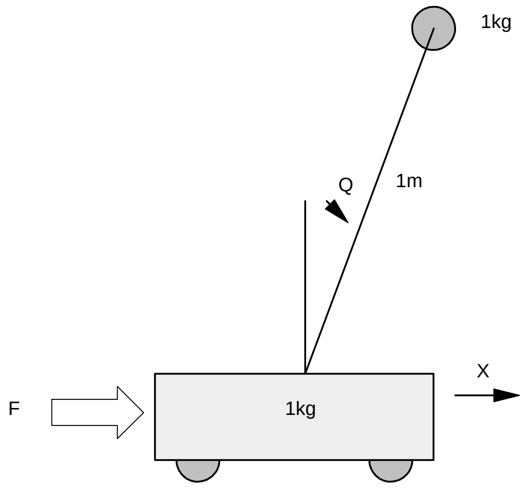
MIMO LQG Control

(work in progress)

Multi-Input, Multi-Output Systems

One problem with Bass Gura (pole placement) is that there is only a closed-form solution for single-input systems. When you have multiple inputs, you have to either turn off some inputs (making it a single-input system) or comine the inputs (which may be sub-optimal).

For example, consider the problem of stabilizing a cart and pendulum where you can apply a torque to the pendulum:



$$s\begin{bmatrix} x \\ \theta \\ sx \\ s\theta \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -19.6 & 0 & 0 \\ 0 & 29.4 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \theta \\ sx \\ s\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} F + \begin{bmatrix} 0 \\ 0 \\ -2 \\ 3 \end{bmatrix} T$$

Bass-Gura is unable to deal with two inputs since this results in eight feedback gains:

$$\begin{bmatrix} F \\ T \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \end{bmatrix}\begin{bmatrix} x \\ \theta \\ sx \\ s\theta \end{bmatrix}$$

with four constraints (the four closed-loop poles of A - BKx).

With LQR control, in contrast, multiple inputs can be dealt automatically: the algorithm finds the feedback gains to minimize the cost function. If there are more gains available, the additional gains allow you to reduce the cost.

Effect of the Weightings on R:

R tells you how much cost is associated with each input. By adjusting each term, you can share the control effort or shift the control effort from one input to the other.

Example: Find the optimal feedback gains for

$$Q = C^T C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
>> Q = diag([1,0,0,0]);
>> R = diag([1,1]);
>> Kx = lqr(A, B, Q, R)

    0.8816   -4.6251    2.0757    0.4243     force input
    0.4720   18.4399    1.3875    4.2661     torque input

>> eig(A - B*Kx)

  -5.4250 + 0.1943i
  -5.4250 - 0.1943i
  -0.4124 + 0.4031i
  -0.4124 - 0.4031i
```

Note that Kx is a 2x4 matrix. Both inputs are being used (the rows). The gains are comperable - each input has the same weighting. The closed-loop poles are the 'optimal' place to put the poles with this cost function.

Repeat for

$$R = \begin{bmatrix} 1000 & 0 \\ 0 & 1 \end{bmatrix}$$

```
>> R = diag([1000,1]);
>> Kx = lqr(A, B, Q, R)

    0.0316    0.0147    0.4352    0.2889      force input: weight = 1000
   -0.0473   19.5890    0.2619    3.7911      torque input: weight = 1

>> eig(A - B*Kx)

  -5.4253 + 0.1843i
  -5.4253 - 0.1843i
  -0.0725 + 0.0725i
  -0.0725 - 0.0725i
```

Note that the first input (force) has smaller feedback gains due to the large weighting (cost) associated with this input. The system is still stable - but slower (due to using less input).

Repeat for

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1000 \end{bmatrix}$$

```
>> R = diag([1,1000]);
>> Kx = lqr(A, B, Q, R)

   -0.9755  -68.2469   -2.7818  -14.3206     force input: weight = 1
    0.0070    0.2406    0.0184    0.0558     torque input: weight = 1000

>> eig(A - B*Kx)

  -5.4218 + 0.0618i
  -5.4218 - 0.0618i
  -0.4129 + 0.4036i
  -0.4129 - 0.4036i
```

Note here that the second input (with a weighting of 1000) has much smaller gains. With a cost of 1000x more than the first input, the control law tries to minimize these gains.

Also note that the system is again stable. For the cost to be finite, the system can't go to infinity (i.e. must be stable).

One of the beauties of LQR control is, if you have multiple inputs, you can use any and all of them. You can also adjust how much control effort comes from each input through the weightings of R.

Effect on the Weightings of Q

Q penalizes the states: higher weightings force the corresponding state to be close to zero (or its final value).

Example: Design a control law so that

- The cart position (x) has a 2% settling time less than 3 seconds
- With less than 5% overshoot, and
- The angle remains less than 20 degrees (0.35 radians)

Start with Q weighting the cart position (x):

$$Q = \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

First, find the feedback gains, Kx:

```
>> Q = diag([1,0,0,0]);
>> R = diag([1,1]);
>> Kx = lqr(A, B, Q, R)

    0.8816   -4.6251    2.0757    0.4243
    0.4720   18.4399    1.3875    4.2661
```

This system has two inputs (force and torque). The DC gain to two outputs (position and angle) are:

```
>> DC = -Cxq*inv(A-B*Kx)*B

    0.8816    0.4720
   -0.0482    0.0900
```

If you want to drive the two outputs separately, Kr is the inverse of the 2x2 DC gain:

```
>> Kr = inv(DC)

    0.8816   -4.6251
    0.4720    8.6399
```

meaning:

$$\begin{bmatrix} F \\ T \end{bmatrix} = K_r \begin{bmatrix} x_{ref} \\ \theta_{ref} \end{bmatrix}$$

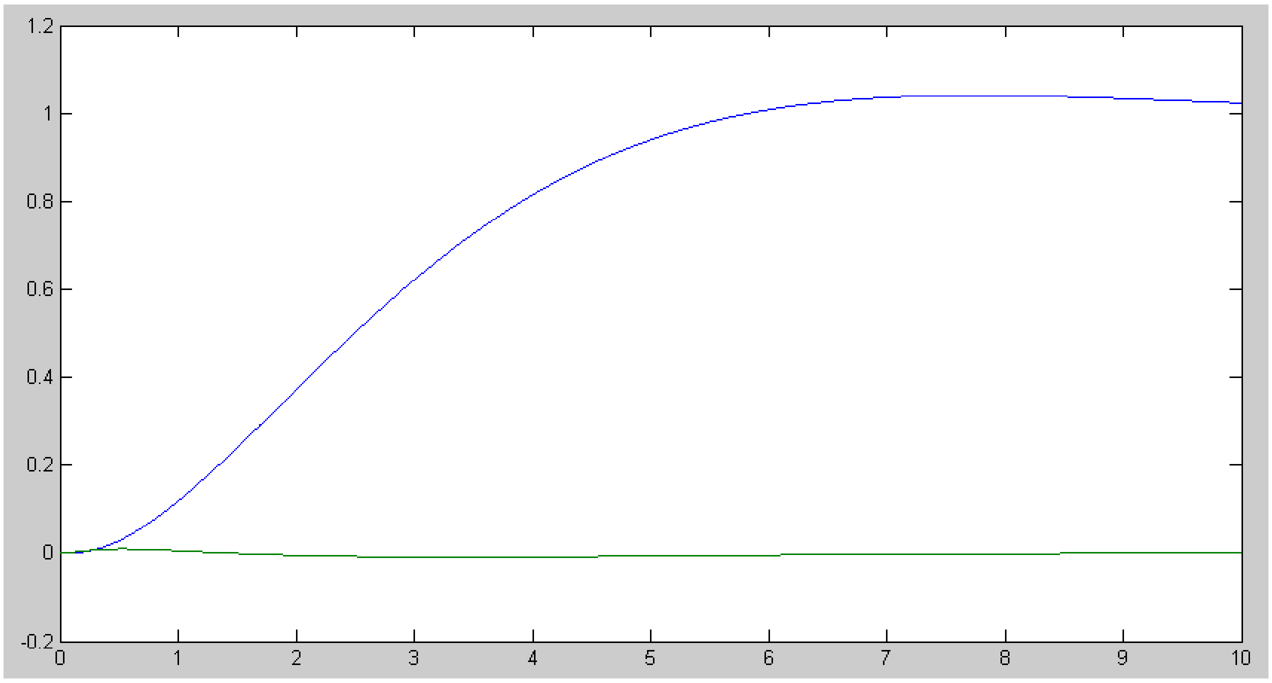
Assuming angle is always supposed to be zero, you can simplify this by using only the first column of Kr:

$$\begin{bmatrix} F \\ T \end{bmatrix} = K_r \begin{bmatrix} x_{ref} \\ 0 \end{bmatrix}$$

```
>> Kr = Kr(:,1)

    0.8816
    0.4720
```

The step response from Xref is then:

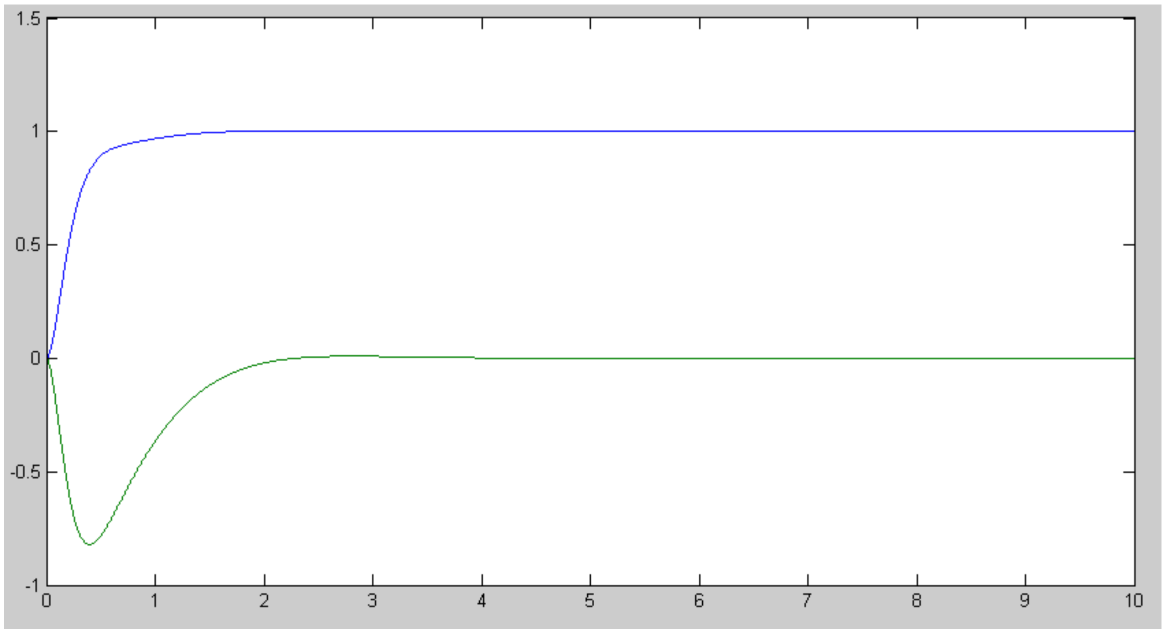
```
>> D = zeros(2,1);
>> G = ss( A - B*Kx,  B*Kr,  Cxq, D);
>> y = step(G,t);
>> plot(t,y)
```



Step Response to Xref: Position (blue) and Angle (green). Q = diag(1 0 0 0) R = diag(1 1)

This is a bit slow. To speed it up, increase the weighting on position (x). Repeating the design process:

```
>> Q = diag([1000,0,0,0]);
>> R = diag([1,1]);
>> Kx = lqr(A, B, Q, R)

   25.9388    5.6056   10.9015    5.9345
  -18.0881   17.8385    0.7840    4.7834

>> DC = -Cxq*inv(A-B*Kx)*B

    0.0259   -0.0181
    0.0584    0.0837

>> Kr = inv(DC)

   25.9388    5.6056
  -18.0881    8.0385

>> Kr = Kr(:,1)

   25.9388
  -18.0881

>> G = ss( A - B*Kx,  B*Kr,  Cxq, D);
>> y = step(G,t);
>> plot(t,y)
```

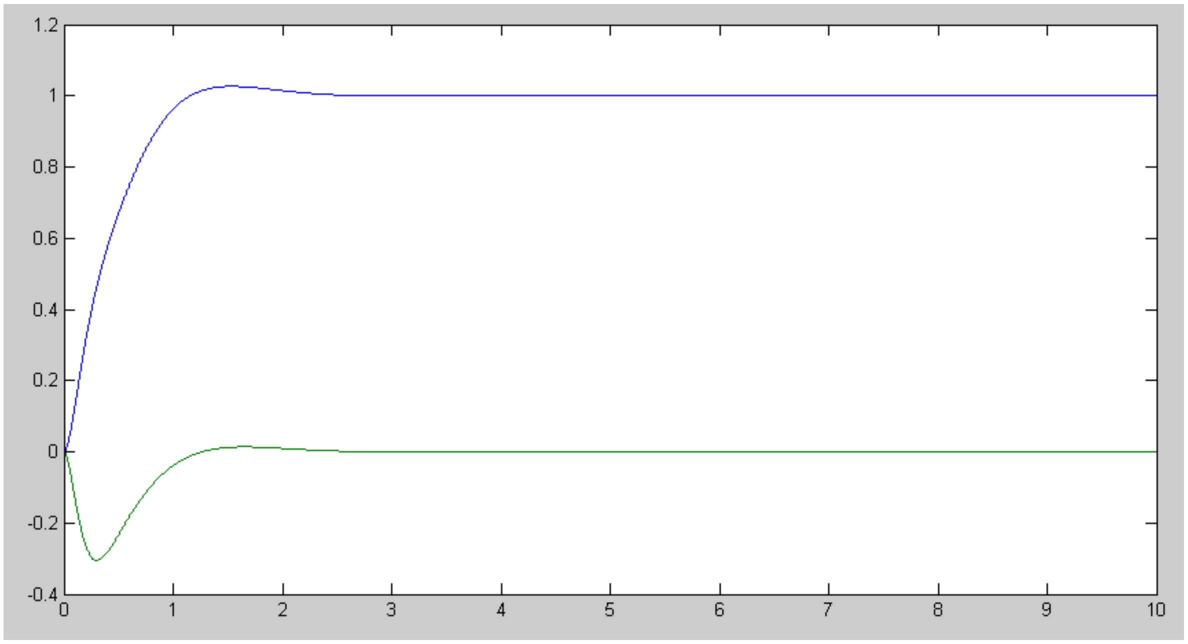


Step Response to Xref: Position (blue) and Angle (green). Q = diag(1000 0 0 0) R = diag(1 1)

On the good side, the speed is less than 3 seconds. On the bad side, the angle reaches -0.8 radians (45 degres). This may be excessive.

To reduce the angle, increase the weighting on the second state (angle).

```
>> Q = diag([1000,1000,0,0]);
>> R = diag([1,1]);
>> Kx = lqr(A, B, Q, R)

   31.1386    5.7709   12.8486    7.1439
   -5.5123   42.3996    3.2744    7.5397

>> DC = -Cxq*inv(A-B*Kx)*B

    0.0311   -0.0055
    0.0053    0.0297

>> Kr = inv(DC)

   31.1386    5.7709
   -5.5123   32.5996

>> Kr = Kr(:,1)

   31.1386
   -5.5123

>> G = ss( A - B*Kx,  B*Kr,  Cxq, D);
>> y = step(G,t);
>> plot(t,y)
```



Step Response to Xref: Position (blue) and Angle (green). Q = diag(1000 1000 0 0) R = diag(1 1)

Now the system is reasonable:

- The position reaches its final value in about 3 seconds
- With 2.6% overshoot
- The maximum angle is -0.3 radians (17 degrees)

Example: Design a feedback controller for a gantry system such that

- The step response for a 10m step input is as quick as possible, and
- The maximum swing in less than 10 degrees

The net control law is:

```
Kx =     31.1386    5.7709   12.8486    7.1439
         -5.5123   42.3996    3.2744    7.5397


Kr =     31.1386
         -5.5123
```

$$\begin{bmatrix} F \\ T \end{bmatrix} = K_r R - K_x X$$