Linear Quadratic Gaussian / Loop Transfer Recovery

LQG/LTR

An advantage of pole placement (Bass Gura) is you can place the poles wherever you like. Selecting a specific response is likewise easy. A problem with pole-placement is it tends to give very large feedback gains. This is due to trying to force a system to behave way it doesn't want to.

An advantage of LQG control is you tend to get smaller feedback gains and more robust control designs. A problem is it can be difficult to select Q and R to get a specific response.

LQG/LTR is a way to use LQG control and get a specific response.

Here's the idea: instead of applying the set-point directly to the plant, run the set point through a reference model first. This model defines how the plant is *supposed* to behave.

$$sX_m = A_m X_m + B_m R$$

$$Y_m = C_m X_m$$

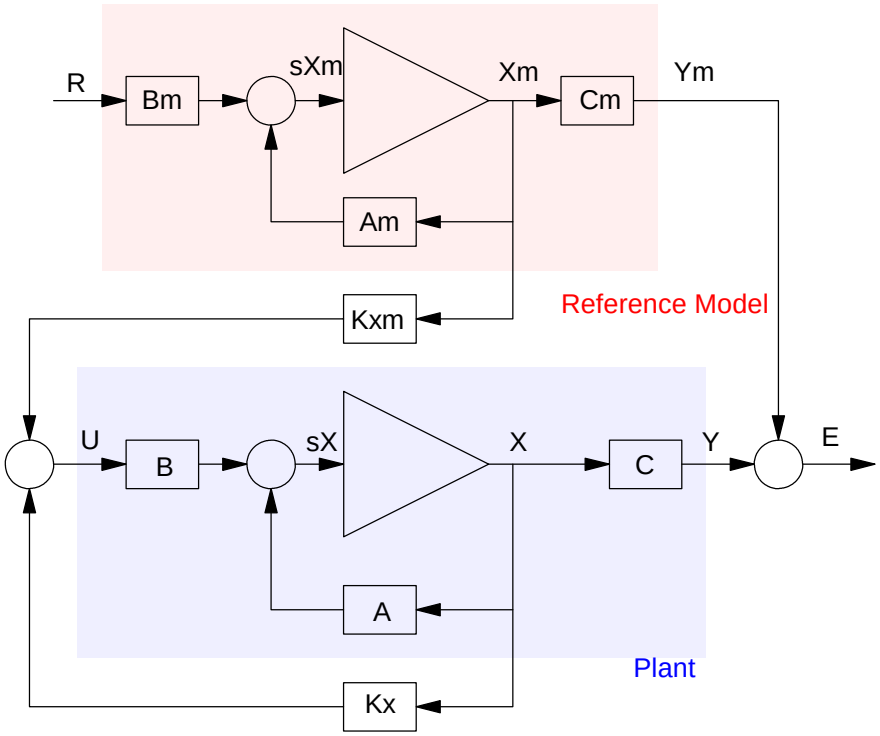
Next, design a LQG controller for this augmented system. If you define the output to be the error between the plant and the reference model

$$E = Y - Y_m$$

and define a set of full-state feedback gains which force E to be small (meaning Q is large), the plant will behave like the reference model.

LQG/LTR Formulation

Define two systems: the plant and the reference model. Define the output to be the difference between the two outputs:



LQG/LTR Formulation: Define a plant and a reference model which defines how the plant should behave. The output is the difference between the two

In state-space, the augmented system is:

$$s\begin{bmatrix} X \\ X_m \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & A_m \end{bmatrix}\begin{bmatrix} X \\ X_m \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix}U + \begin{bmatrix} 0 \\ B_m \end{bmatrix}R$$

$$E = Y - Y_m$$

$$E = \begin{bmatrix} C & -C_m \end{bmatrix}\begin{bmatrix} X \\ X_m \end{bmatrix}$$

Design with LQR controller where

$$Q = \alpha\begin{bmatrix} C & -C_m \end{bmatrix}^T\begin{bmatrix} C & -C_m \end{bmatrix}$$

R = 1

and α is a large number (as $\alpha \to \infty$, E $\to$ 0). The full-state feedback gains will be the gains time X and Xm

$$U = -\begin{bmatrix} K_x & K_m \end{bmatrix}\begin{bmatrix} X \\ X_m \end{bmatrix}$$

Example: Force a metal bar (4-pole heat equation) to behave as

$$Y_m = \left(\frac{10}{s^2+2s+10}\right) R$$

Solution: Define the reference model to be

$$sX_m = \begin{bmatrix} 0 & 1 \\ -10 & -2 \end{bmatrix} X_m + \begin{bmatrix} 0 \\ 10 \end{bmatrix} R$$

$$Y_m = \begin{bmatrix} 1 & 0 \end{bmatrix} X_m$$

Define an augmented system with the plant and reference model:

```
-->A = [-2,1,0,0;1,-2,1,0;0,1,-2,1;0,0,1,-1]
-->B = [1;0;0;0]
-->C = [0,0,0,1];

-->Am = [0,1;-10,-2]
-->Bm = [0;10]
-->Cm = [1,0]

-->A6 = [A, zeros(4,2);zeros(2,4),Am]

          plant          ref model
   -2    1    0    0 :   0     0
    1   -2    1    0 :   0     0
    0    1   -2    1 :   0     0
    0    0    1   -1 :   0     0
   - - - - - - - - - - - - - - - -
    0    0    0    0 :   0     1
    0    0    0    0 : -10    -2

-->B6 = [B;0*Bm]

    1.
    0.
    0.
    0.
    0.
    0.

-->C6 = [C,-Cm]

    0.    0.    0.    1.  - 1.    0.
```

To force the error to zero, define $Q = C^T C$

```
-->Q = C6'*C6
-->R = 1;
```

Case 1: $Q = 10^6\ C^TC$

```
-->Q6 = 1e6 * C6'*C6

-->K6 = lqr( A6,  B6,  Q6, R )

    8.7795468    56.099315    248.67487    685.44677  -246.06529  -188.6547
```

The poles of the closed-loop system are

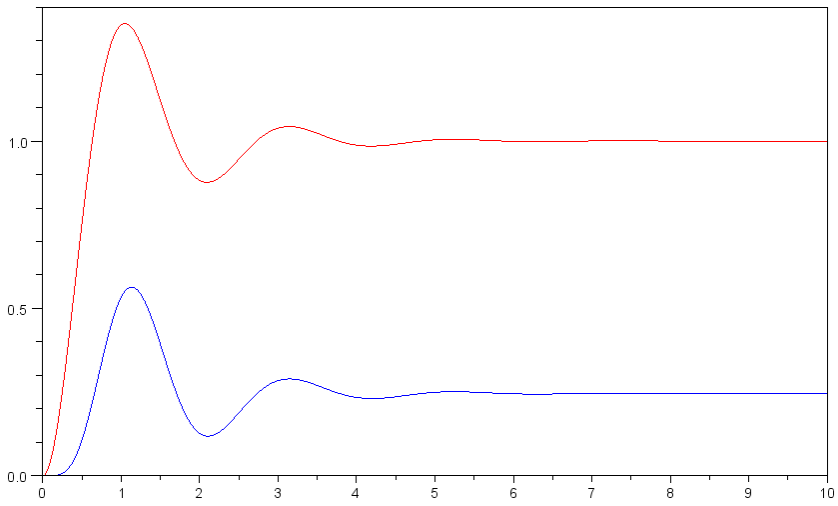
```
-->eig(A6-B6*K6)

  - 2.2976458 + 4.8124056i       Dominant Pole
  - 2.2976458 - 4.8124056i
  - 5.5921276 + 1.9727695i
  - 5.5921276 - 1.9727695i
  - 1. + 3.i                     Reference Model's Poles
  - 1. - 3.i
```

Note

- Two poles are at $-1 \pm j3$ - which is the reference model
- The other poles are a little faster - meaning it will sort of track the reference model. They're not that much faster, however, so tracking will be poor.

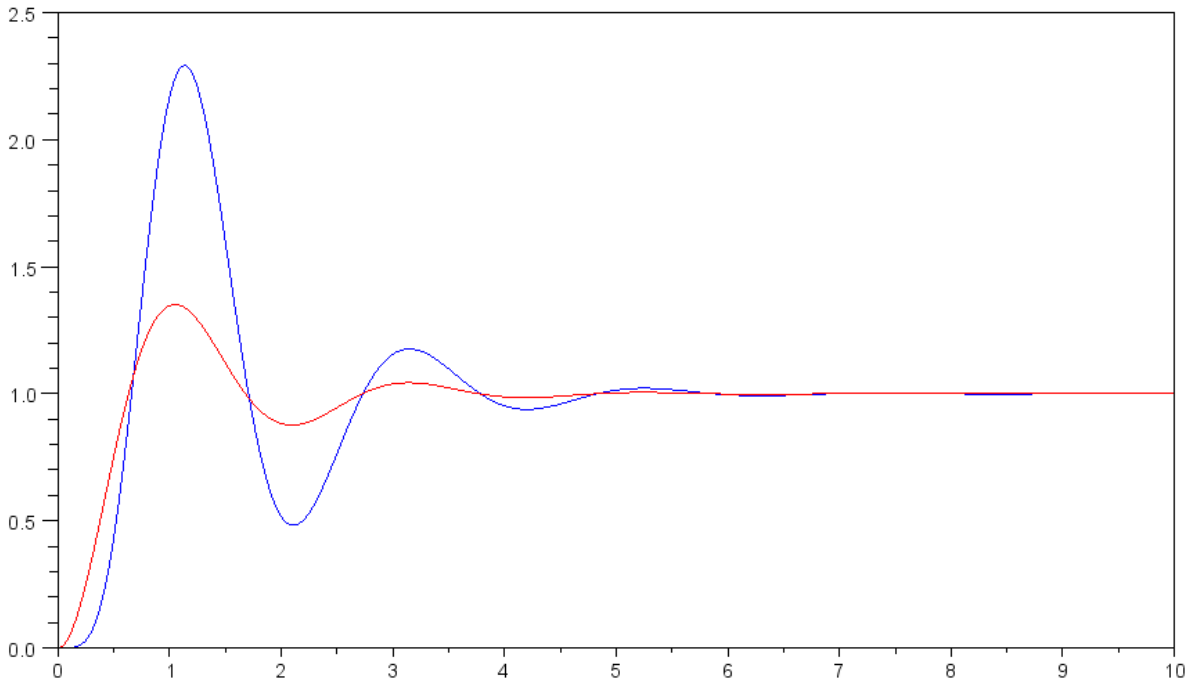
You can check this by plotting the step response of the closed-loop system from R

```
-->Br = [0*B;Bm]
-->C6 = [ Cy ; Cm ]

   0 0 0 1 0 0       plant output, y
   0 0 0 0 1 0       reference model output

D6 = [ 0 ; 0 ]

   0
   0

-->G = ss(A6-B6*K6,Br,C6,D6)
-->t = [0:0.01:10]';
-->Y = step(G,t);
-->plot(t,Y)
```



Step Response of the Reference Model (red) and Plant (blue) for $Q = 10^6\, C^T C$

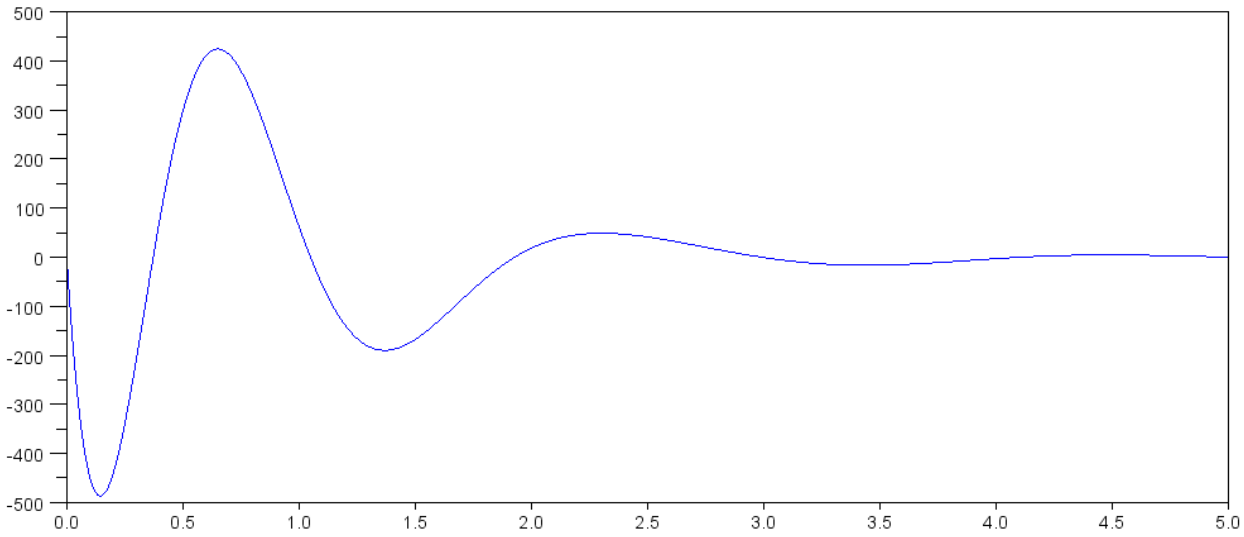
This isn't a fair comparison since the DC gain is so different. You can adjust this by adding a pre-scalar to make the DC gain of the system equal to one



Response to a Step Input in R for the Plant (blue) and Reference Model (red)

The resulting input is:

```
-->Gu = ss(A6-B6*K6,Br/DC,K6,0);
-->U = step(Gu,t);
```



Input vs. Time for Step Input to R

Note that the response is sort of following the reference model. It's not great, however.

The response is the "optimal" response, however, for the Q and R selected.

To get a better response, increase the weighting on Q to 10^{12} (penalize the error more and more)

```
-->Q6 = 1e12 * C6'*C6;
-->R6 = 1;
-->K6 = lqr(A6, B6, Q6, R)

    75.830444    3026.789    73170.323    923726.06  -967507.21  -75408.799

-->eig(A6-B6*K6)

  - 12.13016 + 29.146288i
  - 12.13016 - 29.146288i
  - 29.285062 + 12.072677i
  - 29.285062 - 12.072677i
  - 1. + 3.i
  - 1. - 3.i
```

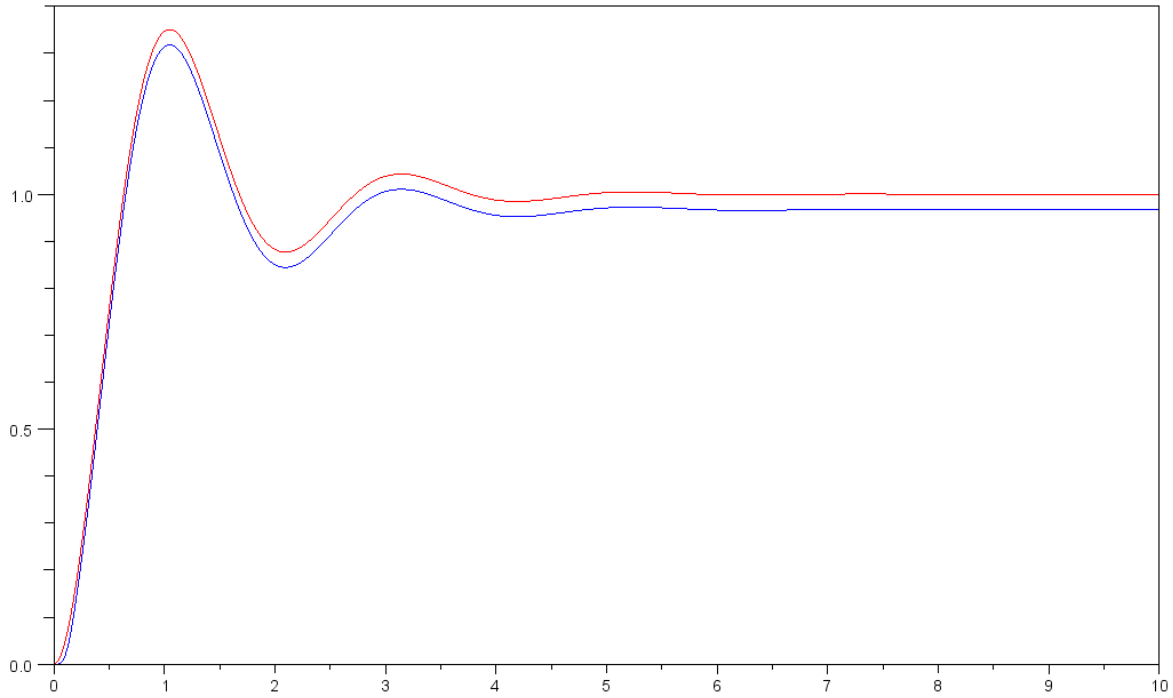
Note that with larger Q,

- The feedback gains are much larger. It takes larger gains to force the desired response.

- The closed-loop poles are more reasonable. The pole at $-1 \pm j3$ is the reference model. The other poles are the plant - which are much faster than the reference model (meaning it should be able to track the model)

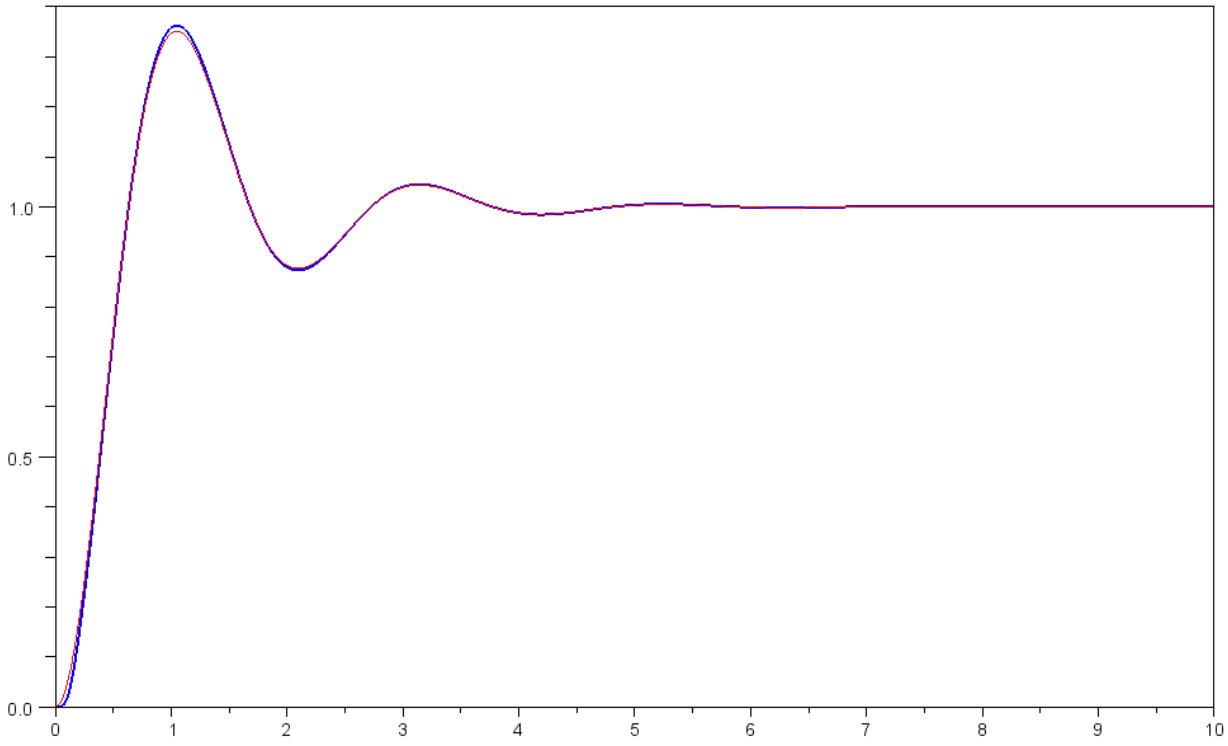
Checking by plotting the response of the closed-loop system:

```
-->DC = -Cy*inv(A6-B6*K6)*Br

    0.9675072

-->G = ss(A6-B6*K6,Br,C6,D6)
-->Y = step(G,t);
-->plot(t,Y)
```



Step Response of the Plant (blue) and Reference Model (red) for $Q = 10^{12}\, C^T C$

or adjusting to make the DC gain one



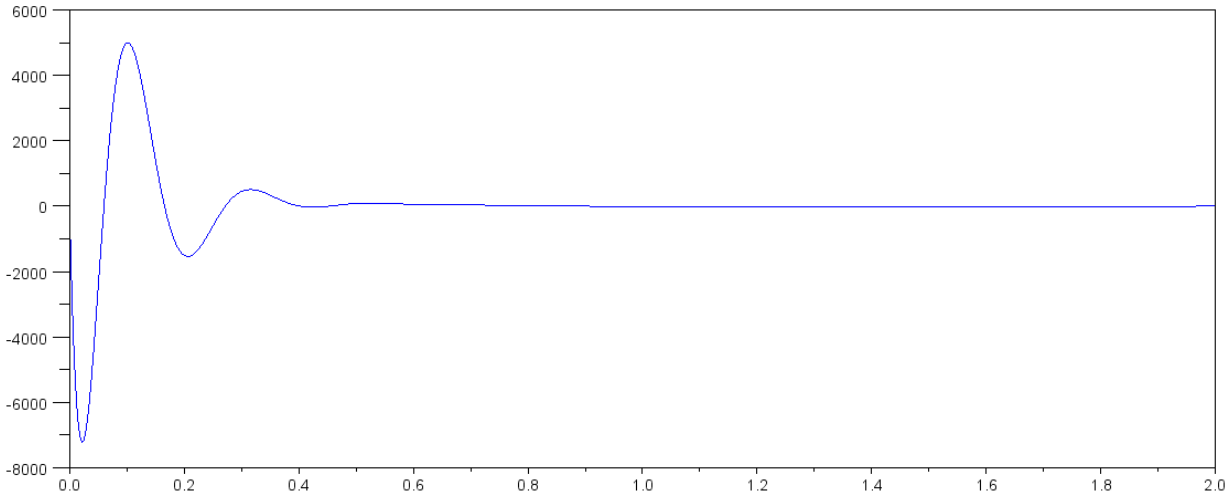
Step Response of the Plant (blue) and Reference Model (red) for $Q = 10^{12}\, C^T C$ with a DC gain of one

Note

- The plant behaves almost identical to the reference model - even though you are trying to make a heat equation oscillate.
- This comes at a cost.

The input is quite large:

```
-->Gu = sspack(A6-B6*K6,Br,K6,0);

-->U = step(Gu,t) / DC;

-->plot(t,U)
```



Input (U) for $Q = 10^{12}\, C^T C$ with a DC gain of one