Topic #14

16.31 Feedback Control

State-Space Systems

- **Full-state Feedback Control**
- How do we change the poles of the state-space system?
- Or, even if we can change the pole locations.
- Where do we change the pole locations to?
- **How well does this approach work?**

Reference Inputs

- So far we have looked at how to pick K to get the dynamics to have some nice properties (*i.e.* stabilize A)

- The question remains as to how well this controller allows us to track a reference command?

  – Performance issue rather than just stability.

- Started with

$$\begin{aligned} \dot{x} &= Ax + Bu \quad y = Cx \\ u &= r - Kx \end{aligned}$$

- For **good tracking performance** we want

$$y(t) \approx r(t) \text{ as } t \to \infty$$

- Consider this performance issue in the frequency domain. Use the final value theorem:

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} sY(s)$$

  Thus, for good performance, we want

$$sY(s) \approx sR(s) \text{ as } s \to 0 \quad \Rightarrow \quad \left.\frac{Y(s)}{R(s)}\right|_{s=0} = 1$$

- So, for good performance, the transfer function from $R(s)$ to $Y(s)$ should be approximately 1 at DC.

- **Example #1:** Consider:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{aligned}$$

  – Already designed $K = \begin{bmatrix} 14 & 57 \end{bmatrix}$

  – Then the closed-loop system is

$$\dot{x} = (A - BK)x + Br \quad y = Cx$$

  – Which gives the transfer function

$$\begin{aligned} \frac{Y(s)}{R(s)} &= C\left(sI - (A - BK)\right)^{-1} B \\ &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s+13 & 56 \\ -1 & s-2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{s-2}{s^2+11s+30} \end{aligned}$$

- Assume that $r(t)$ is a step, then by the FVT

$$\left.\frac{Y(s)}{R(s)}\right|_{s=0} = \frac{-2}{30} \neq 1 \text{ !!}$$

  – So our step response is quite poor.

- One obvious solution is to scale the reference input $r(t)$ so that

$$u = \bar{N}r - Kx$$

  - $\bar{N}$ is an extra gain used to scale the closed-loop transfer function

- Now we have

$$\dot{x} = (A - BK)x + B\bar{N}r \ , \quad y = Cx$$

  so that

$$\frac{Y(s)}{R(s)} = C\left(sI - (A - BK)\right)^{-1} B\bar{N}$$

  If we had made $\bar{N} = -15$, then

$$\frac{Y(s)}{R(s)} = \frac{-15(s-2)}{s^2 + 11s + 30}$$

  so with a step input, $y(t) \to 1$ as $t \to \infty$.

- So the steady state step error is now zero, but is this OK?
  - See plots – big improvement in the response, but the transient is a bit weird.

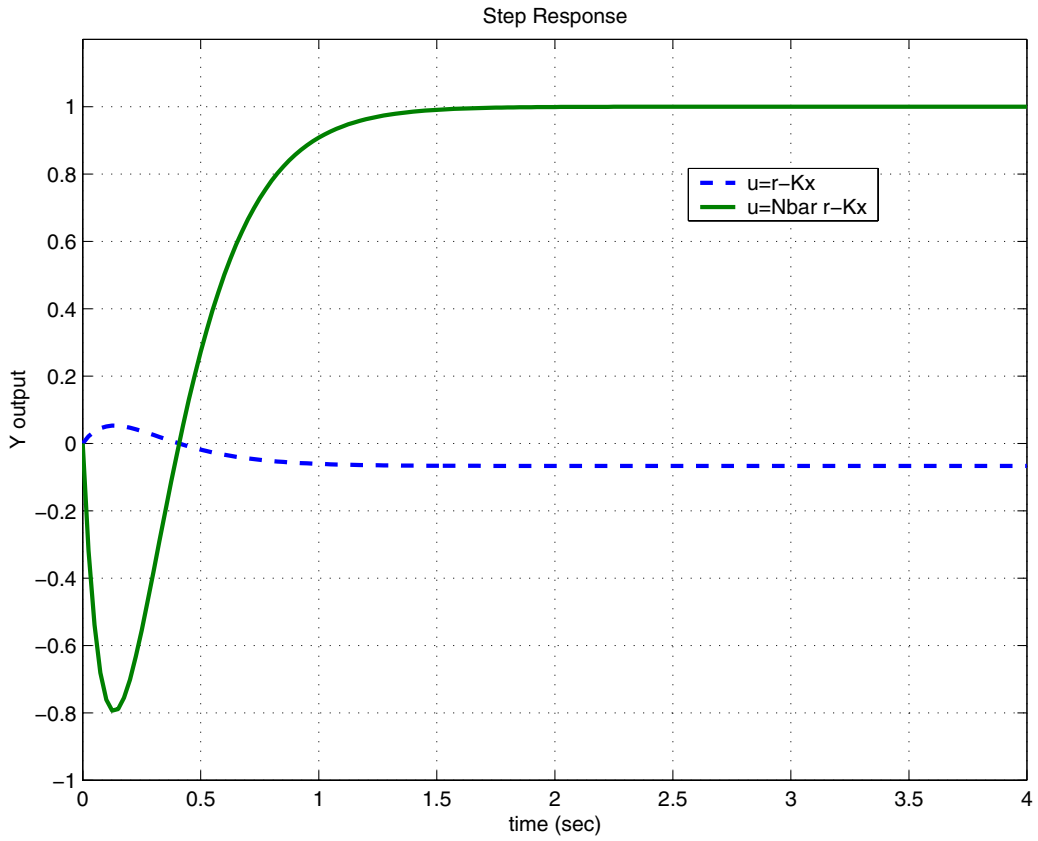


Figure 1: Response to step input with and without the $\bar{N}$ correction.

- Formal way to compute $\bar{N}$ is to change the form of the control input.
  - Consider the analysis for the response to a step input $r = r_{ss}\mathbf{1}(t)$

- At steady state $\dot{x} = 0$, so we have

$$\left.\begin{array}{l} \dot{x} = Ax + Bu \\ y = Cx \end{array}\right\} \quad \begin{array}{r} 0 = Ax_{ss} + Bu_{ss} \\ y_{ss} = Cx_{ss} \end{array}$$

  and if things are going well, then $y_{ss} = r_{ss}$.

$$\begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_{ss} \\ u_{ss} \end{bmatrix} = \begin{bmatrix} 0 \\ r_{ss} \end{bmatrix}$$

  which can be easily solved for x_{ss} and u_{ss}.

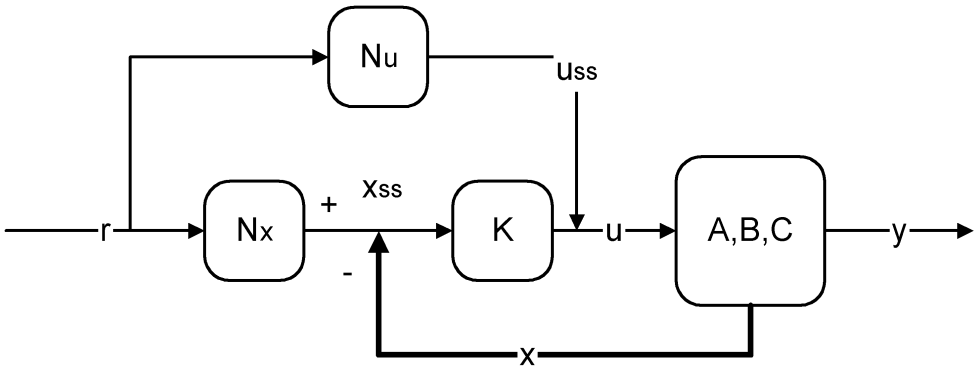
- For purposes of scaling, define:

$$x_{ss} \equiv N_x r_{ss} \qquad u_{ss} \equiv N_u r_{ss}$$

- We would then implement the control in the new form

$$\begin{aligned} u &= \bar{N}r - Kx = (N_u + KN_x)r - Kx \\ &= N_u r + K(N_x r - x) \\ &= u_{ss} + K(x_{ss} - x) \end{aligned}$$

  which can be visualized as:



  – Use N_x to modify the reference command r to generate a **feed-forward** state command to the system x_{ss}.

  – Use N_u to modify the reference command r to generate a **feed-forward** control input u_{ss}

- Note that this development assumed that r was constant, but it could also be used if r is a slowly time-varying command.

  – But as we have seen, the architecture is designed to give good steady-state behavior, and it might give weird transient responses.

- For our example,

$$\begin{bmatrix} x_{ss} \\ u_{ss} \end{bmatrix} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \left[\begin{array}{c} 1 \\ -0.5 \\ \hline -0.5 \end{array}\right]$$

so

$$x_{ss} = \begin{bmatrix} 1 \\ -0.5 \end{bmatrix}, \quad u_{ss} = \begin{bmatrix} -0.5 \end{bmatrix}$$

and

$$\bar{N} = N_u + KN_x = -0.5 + \begin{bmatrix} 14 & 57 \end{bmatrix} \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} = -15$$

as we had before.

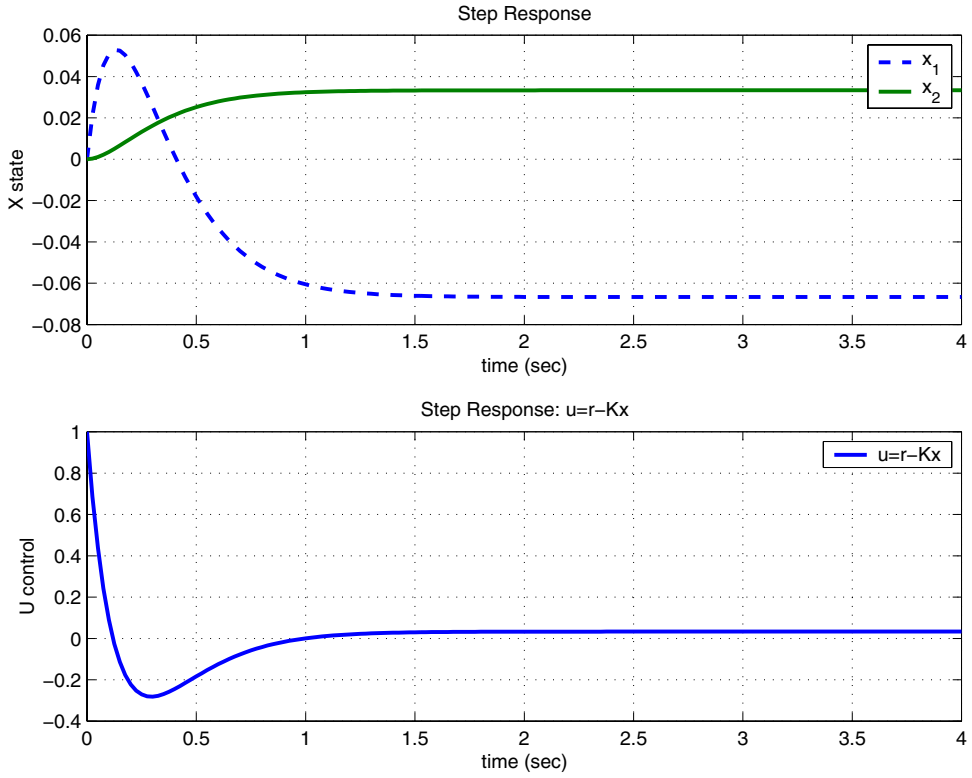


Figure 2: Response to step input **without** the $\bar{N}$ correction. The steady state x and u values are non-zero but they are not the values that give the desired y_{ss}.

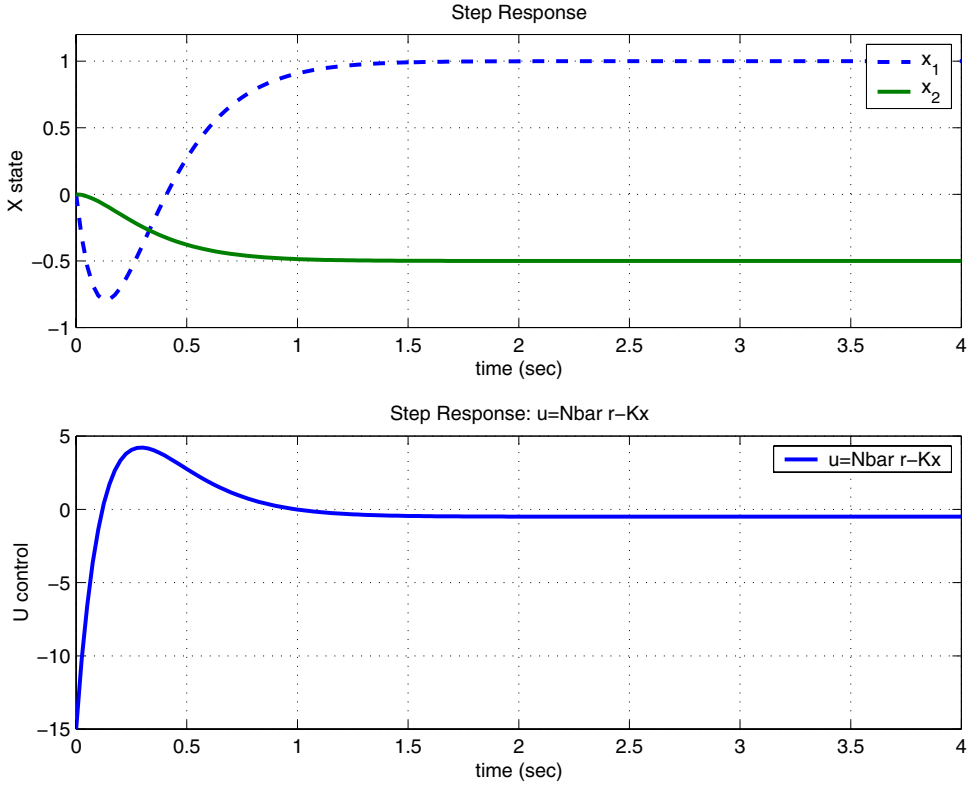


Figure 3: Response to step input **with** the $\bar{N}$ correction. Gives the desired steady-state behavior, but note the higher $u(0)$.

Examples

- Example from Problem set

$$G(s) = \frac{8 \cdot 14 \cdot 20}{(s+8)(s+14)(s+20)}$$

  – Target pole locations $-12 \pm 12j$, -20

```
% system
[a,b,c,d]=tf2ss(8*14*20,conv([1 8],conv([1 14],[1 20])))
% controller gains to place poles at specified locations
k=place(a,b,[-12+12*j;-12-12*j;-20])
TT=[a b;c d];
% find the feedforward gains
N=inv(TT)*[zeros(1,length(a)) 1]';
Nx=N(1:end-1);Nu=N(end);Nbar=Nu+k*Nx;
sys1=ss(a-b*k,b,c,d);
sys2=ss(a-b*k,b*Nbar,c,d);
t=[0:.01:1];
[y,t,x]=step(sys1,t);
[y2,t2,x2]=step(sys2,t);
```

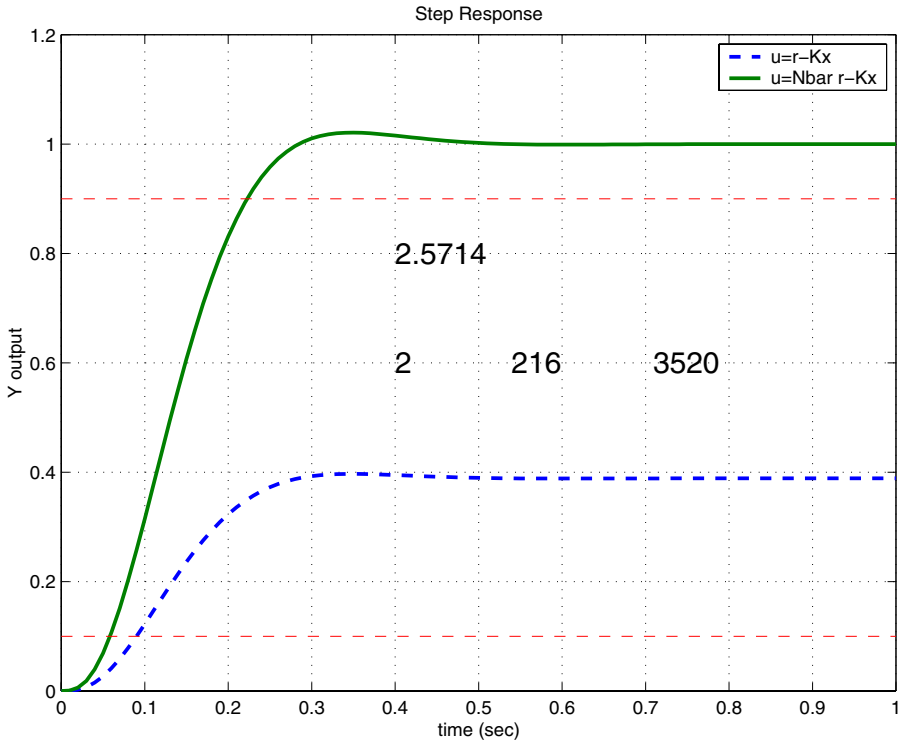


Figure 4: Response to step input with and without the $\bar{N}$ correction. Gives the desired steady-state behavior, with little difficulty!

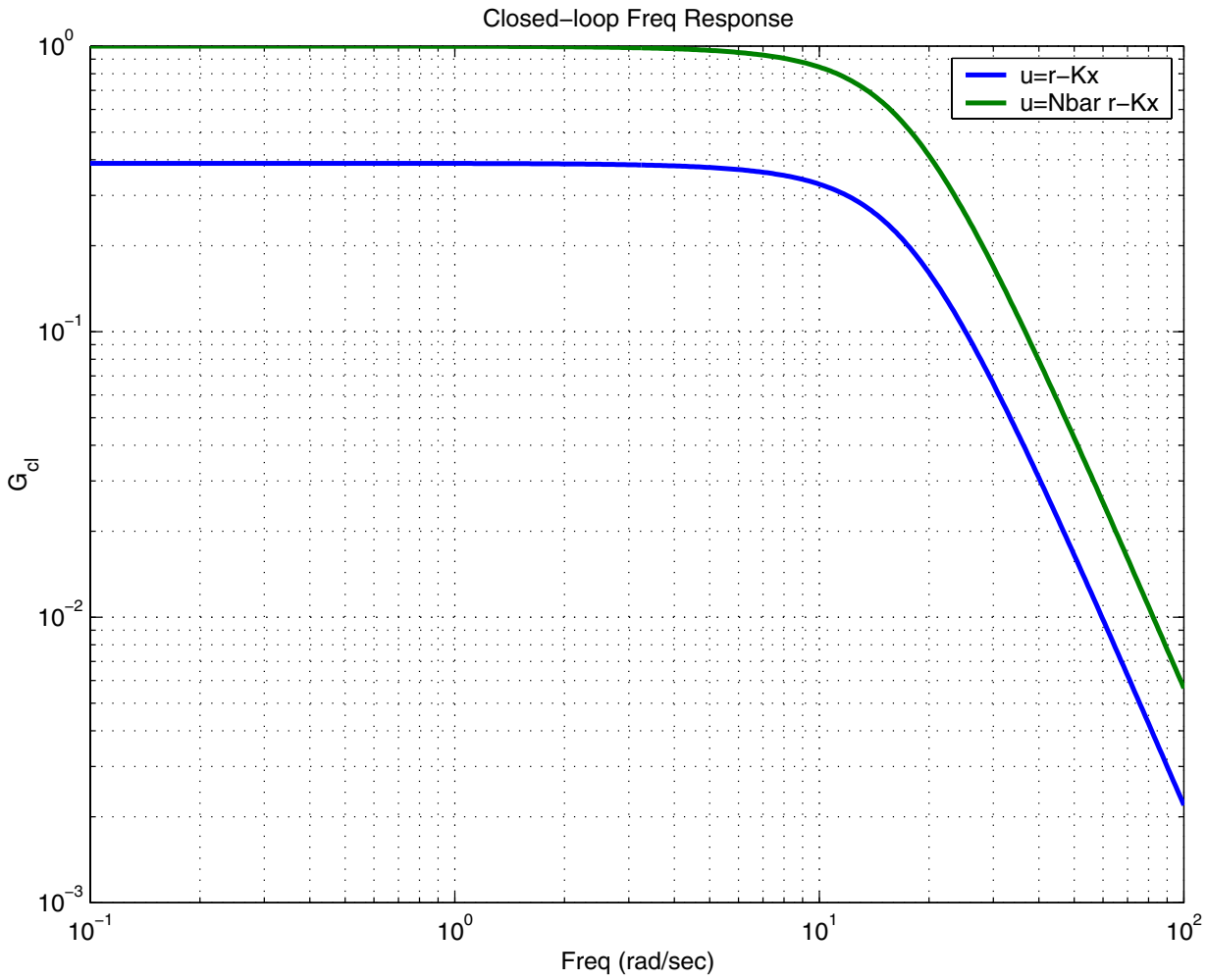


Figure 5: Closed-loop frequency response. Clearly shows that the DC gain is unity

- Example from second problem set

$$G(s) = \frac{0.94}{s^2 - 0.0297}$$

  – Target pole locations $-0.25 \pm 0.25j$

```
[a,b,c,d]=tf2ss(.94,[1 0 -0.0297])
k=place(a,b,[-1+j;-1-j]/4)
TT=[a b;c d];
N=inv(TT)*[zeros(1,length(a)) 1]';
Nx=N(1:end-1);Nu=N(end);Nbar=Nu+k*Nx;
sys1=ss(a-b*k,b,c,d);
sys2=ss(a-b*k,b*Nbar,c,d);
t=[0:.1:30];
[y,t,x]=step(sys1,t);
[y2,t2,x2]=step(sys2,t);
```

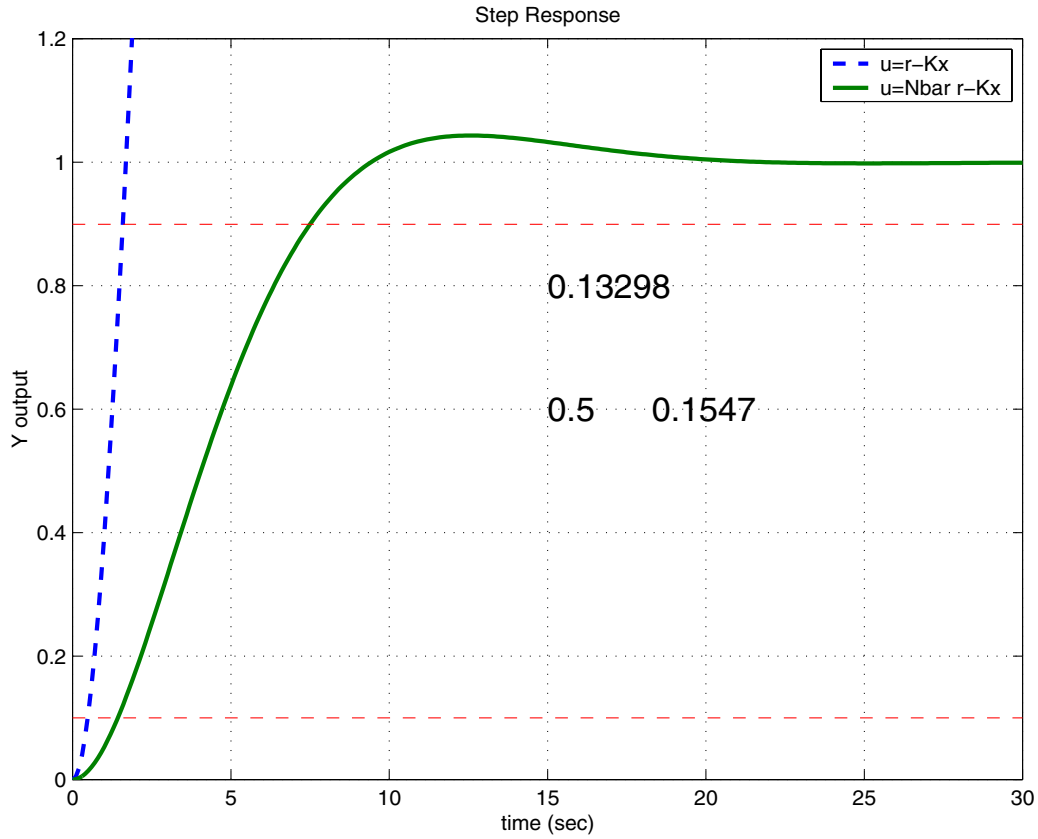


Figure 6: Response to step input with and without the $\bar{N}$ correction. Gives the desired steady-state behavior, with little difficulty!

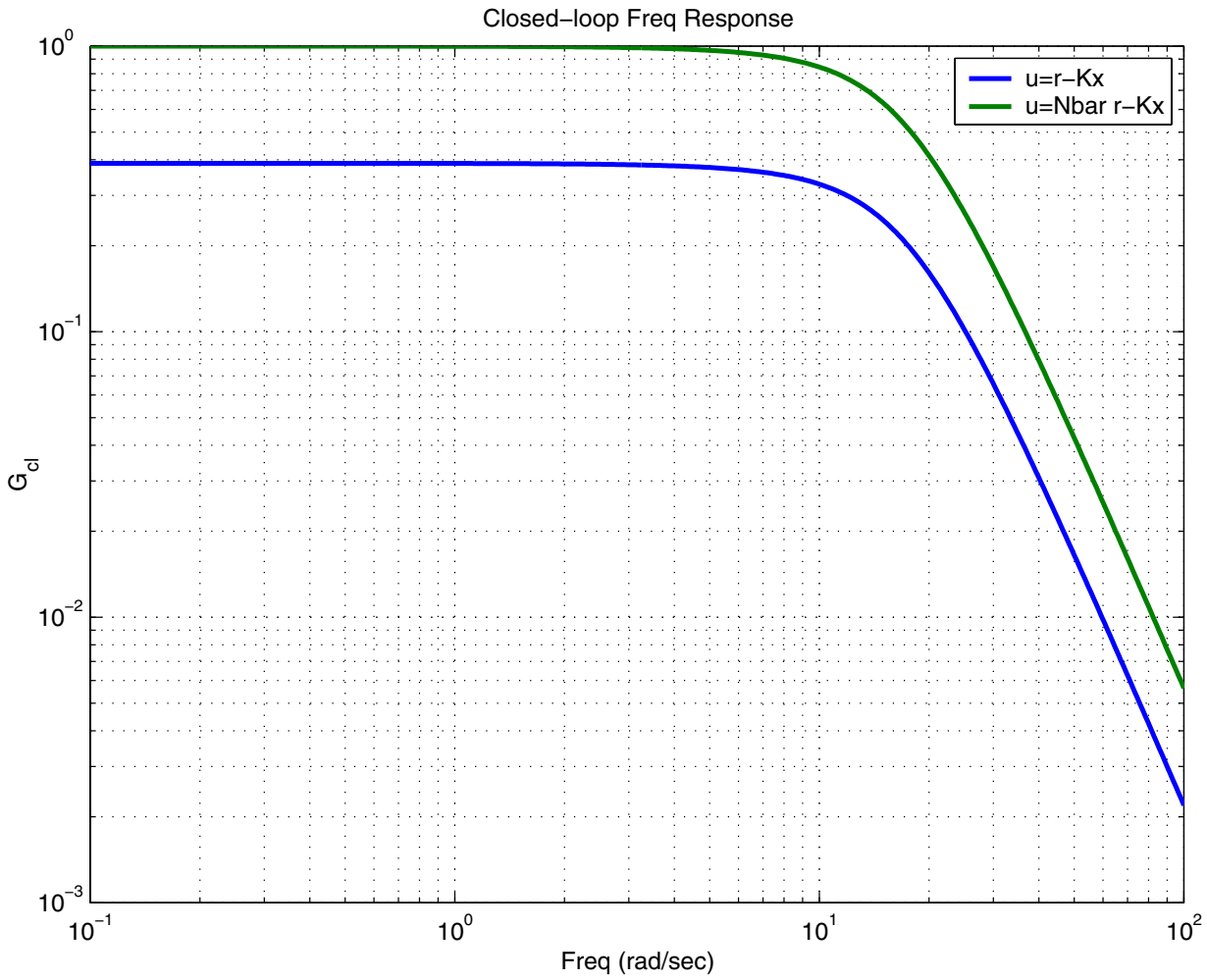


Figure 7: Closed-loop frequency response. Clearly shows that the DC gain is unity

- Ok, so let's try something very challenging...

$$G(s) = \frac{8 \cdot 14 \cdot 20}{(s-8)(s-14)(s-20)}$$

  – Target pole locations $-12 \pm 12j$, -20

```
[a,b,c,d]=tf2ss(8*14*20,conv([1 -8],conv([1 -14],[1 -20])))
k=place(a,b,[-12+12*j;-12-12*j;-20])
TT=[a b;c d];
N=inv(TT)*[zeros(1,length(a)) 1]';
Nx=N(1:end-1);Nu=N(end);Nbar=Nu+k*Nx;
sys1=ss(a-b*k,b,c,d);
sys2=ss(a-b*k,b*Nbar,c,d);
t=[0:.01:1];
[y,t,x]=step(sys1,t);
[y2,t2,x2]=step(sys2,t);
```

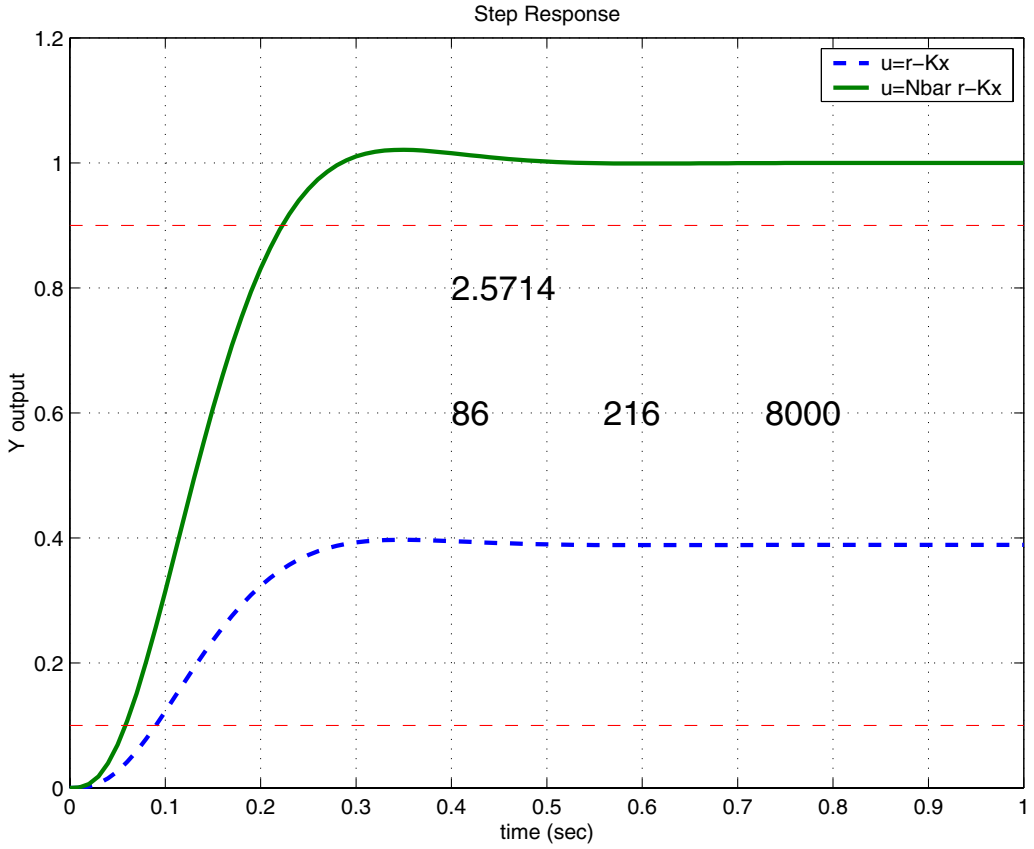


Figure 8: Response to step input with and without the $\bar{N}$ correction. Gives the desired steady-state behavior, with little difficulty!

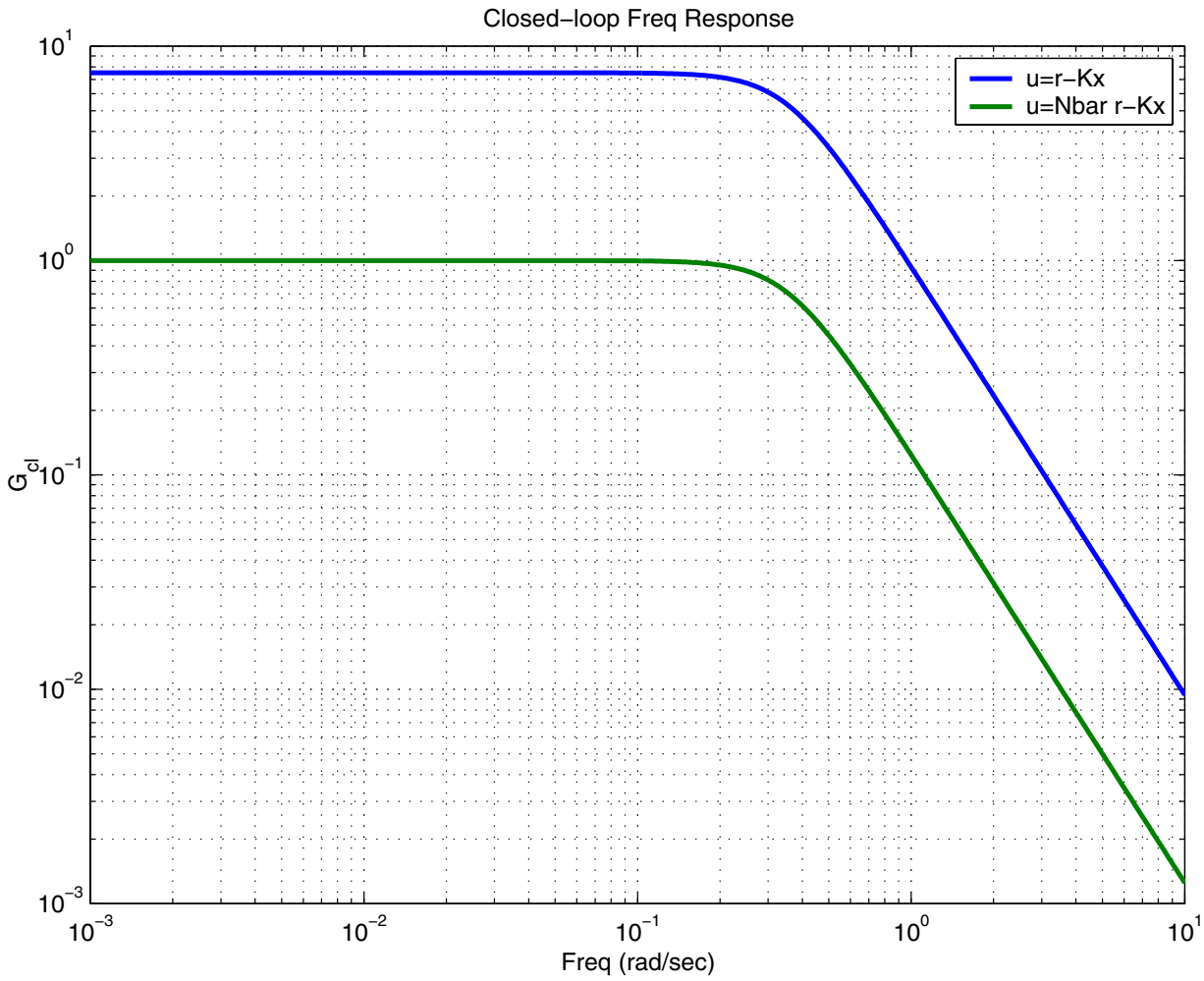


Figure 9: Closed-loop frequency response. Clearly shows that the DC gain is unity

- The worst possible...

$$G(s) = \frac{(s-1)}{(s+1)(s-3)}$$

  - Unstable, NMP!!
  - Target pole locations $-1 \pm j$

```
[a,b,c,d]=tf2ss([1 -1],conv([1 1],[1 -3]))
k=place(a,b,[[-1+j;-1-j]])
TT=[a b;c d];
N=inv(TT)*[zeros(1,length(a)) 1]';Nx=N(1:end-1);Nu=N(end);Nbar=Nu+k*Nx;
sys1=ss(a-b*k,b,c,d);
sys2=ss(a-b*k,b*Nbar,c,d);
t=[0:.01:10];[y,t,x]=step(sys1,t);[y2,t2,x2]=step(sys2,t);
```

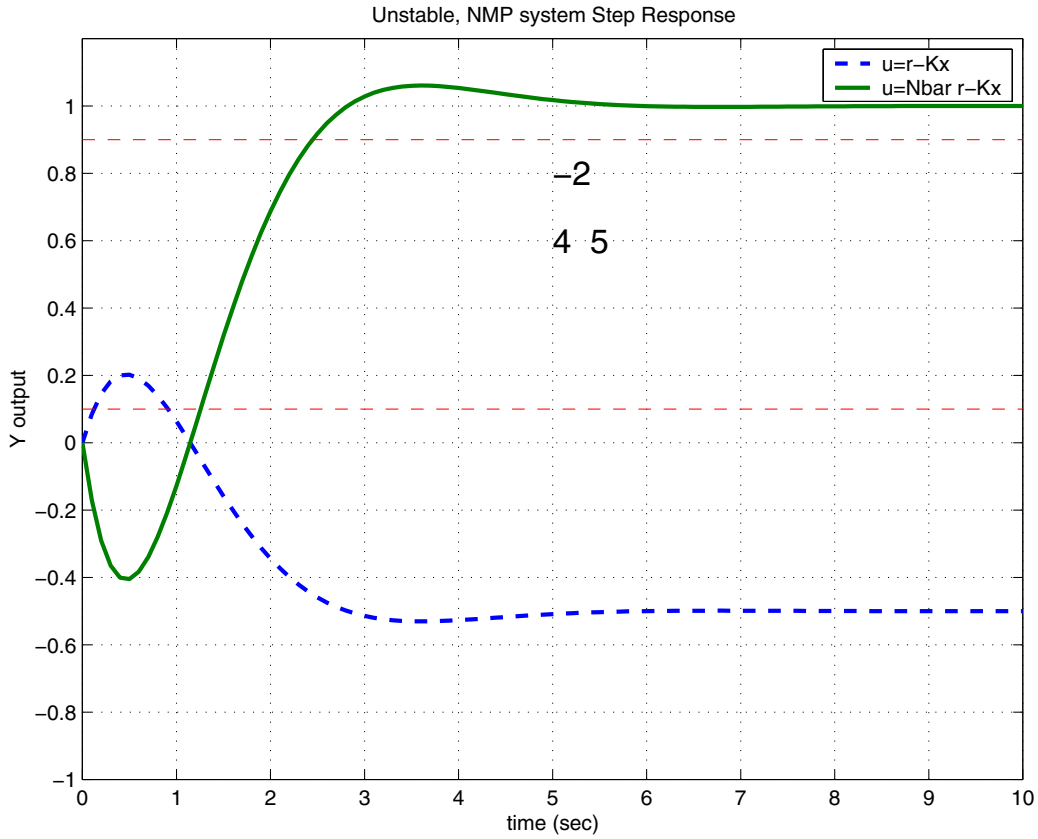


Figure 10: Response to step input with and without the $\bar{N}$ correction. Gives the desired steady-state behavior, with little difficulty!

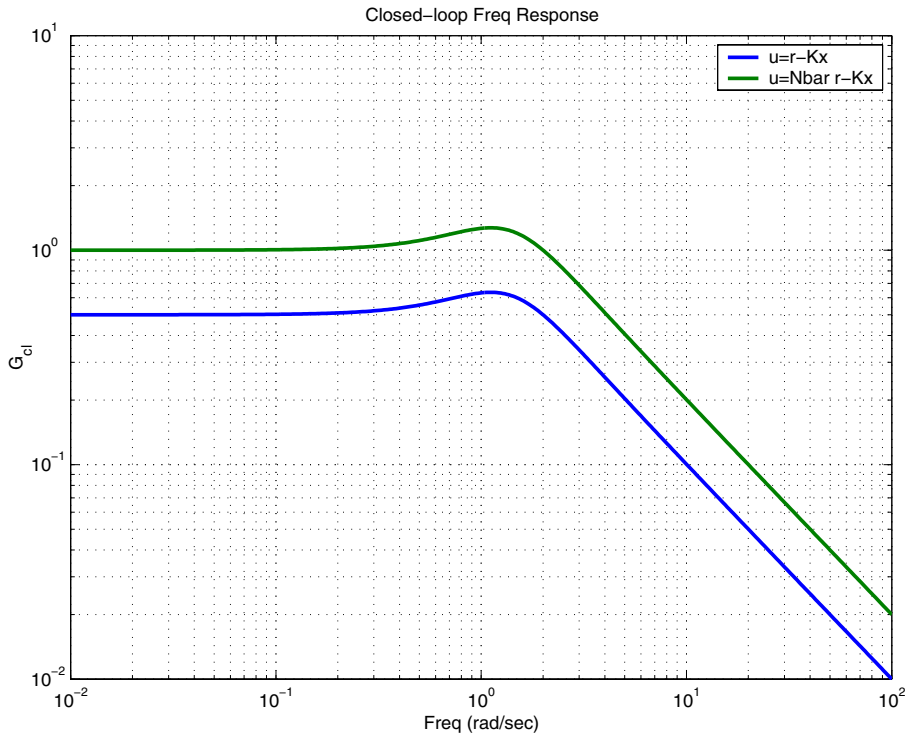


Figure 11: Closed-loop frequency response. Clearly shows that the DC gain is unity

- Full state feedback process is quite simple as it can be automated in Matlab using `acker` and/or `place`

- With more than 1 actuator, we have more than n degrees of freedom in the control $\rightarrow$ we can change the eigenvectors as desired, as well as the poles.

- The real issue now is where to put the poles...

- And to correct the fact that we cannot usually measure the state $\rightarrow$ develop an estimator.