LINEAR QUADRATIC REGULATOR

8.1: Introduction to optimal control

- The engineering tradeoff in control-system design is

Fast response		Slower response
Large intermediate states	*versus*	Smaller intermediate states
Large control effort		Smaller control effort

EXAMPLE: Consider

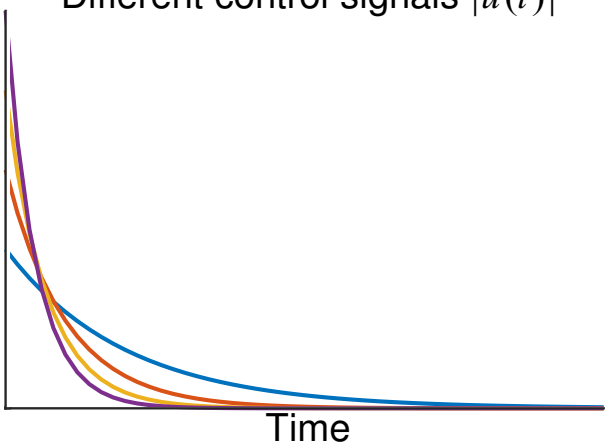
$$\dot{x}(t) = x(t) - u(t)$$

with state feedback $u(t) = -kx(t), \quad k \in \mathbb{R}$.

$$\dot{x}(t) = (1+k)x(t).$$

- Eigenvalue at $1+k$. Can make as negative (fast) as we want, with large negative k and corresponding large input $u(t)$.
- Suppose $x(0) = 1$, so $x(t) = e^{(1+k)t}$ and $u(t) = -ke^{(1+k)t}$.
- As $k \to -\infty$ (*i.e.,* large) $u(t)$ looks more and more like $\delta(t)$, the input we found earlier that (instantaneously) moves $x(t)$ to 0!

Different control signals $|u(t)|$



- To see this, note that

$$\int_0^\infty u(t)\,\mathrm{d}t = \frac{(-k)}{(-k)-1}.$$

- For $-k$ large, $\int u(t)\,\mathrm{d}t = 1$ and $u(t)$ is "bunching up" near $t = 0$.

- In general, as we relocate our eigenvalues farther and farther to the left, so that the closed-loop system is faster and faster, our plant input begins to look like the impulsive inputs we considered earlier.
- Once again, the tradeoff is speed versus gain/ size of input.

Cost functions (switch to discrete time)

- To avoid large inputs, we consider the cost function:

$$J = \sum_{k=0}^{\infty} \left(x[k]^T x[k] + \rho u[k]^2 \right).$$

- We will find the K such that $u[k] = -Kx[k]$ minimizes this cost.
  - We make ρ large if we don't want large inputs ("high cost of control");
  - We make ρ small if we want fast response and don't mind large inputs ("cheap control").

EXAMPLE: Consider (where $x[k]$ is a scalar)

$$x[k+1] = x[k] + u[k]$$

with

$$u[k] = -Kx[k].$$

- Thus $x[k] = (1-K)^k x[0]$ so

$$J = \sum_{k=0}^{\infty} (x[k]^2 + \rho u[k]^2) = \begin{cases} x[0]^2 \dfrac{1+\rho K^2}{1-(1-K)^2}, & 0 < K < 2; \\ \infty, & \text{otherwise.} \end{cases}$$

- Thus, $J = px[0]^2$ where

$$p = \frac{1+K^2\rho}{K(2-K)}.$$

- We can solve for the optimal K for any given ρ by

$$\frac{\mathrm{d}p}{\mathrm{d}K} = \frac{K(2-K)(2K\rho) - (1+K^2\rho)(2-2K)}{[K(2-K)]^2} = 0$$

$$K^2\rho(2-K) = (1+K^2\rho)(1-K)$$

$$2\rho K^2 - \rho K^3 = 1 + \rho K^2 - K - \rho K^3$$

$$\rho K^2 + K - 1 = 0.$$

- So, (the other solution is a maximum, not a minimum)

$$K_{\text{opt}} = \frac{-1+\sqrt{1+4\rho}}{2\rho}.$$

- The optimal cost is

$$J = \frac{\rho(\sqrt{1+4\rho}-1)}{2\rho - \sqrt{1+4\rho}+1}.$$

- For low cost ("cheap") control, let $\rho \to 0$. Then $K_{\text{opt}} \to 1$ since

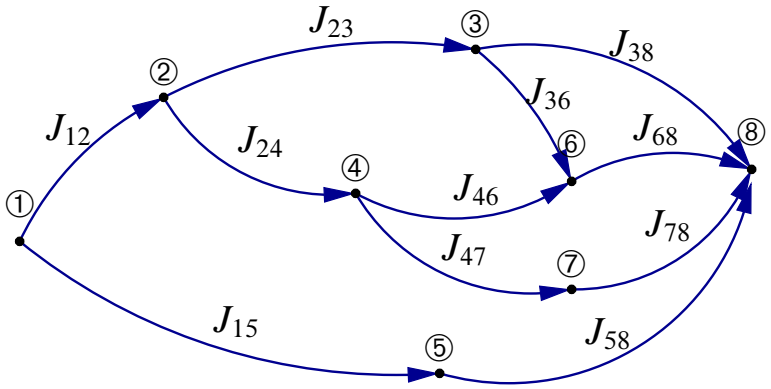
$$\lim_{\rho\to 0} \frac{-1+\sqrt{1+4\rho}}{2\rho} = \lim_{\rho\to 0} \frac{2(1+4\rho)^{-1/2}}{2} = 1,$$

  which is *deadbeat control;* closed-loop eigenvalues at 0.

- For high cost ("expensive") control, let $\rho \to \infty$ then $K_{\text{opt}} \to \dfrac{1}{\sqrt{\rho}}$, which is a *small* (as expected) feedback which just barely stabilizes the system, but plant input is small. Closed loop eigenvalue at $1 - \dfrac{1}{\sqrt{\rho}}$ which is < 1.

8.2: Dynamic programming: Bellman's principle of optimality

- We will want to minimize a more general cost function J— minimization is a topic in optimization theory.
- We will use a tool called dynamic programming.
- Consider the task of finding the lowest-cost route from point x_o to x_f, where there are many possible ways to get there.



- Then

$$J_{18}^* = \min\{J_{15} + J_{58}, J_{12} + J_{24} + J_{46} + J_{68}, \ldots\}.$$

- We need to make only one simple observation:

  In general, if x_i is an intermediate point between x_o and x_f *and* x_i *is on the optimal path,* then

$$J_{of}^* = J_{oi} + J_{if}^*.$$

- This is called Bellman's principle of optimality.

Quadratic forms

- In the cost function

$$J = \sum_{m=0}^{\infty} (x[m]^T x[m] + \rho u[m]^2),$$

all components of $x[m]$ are weighted evenly. Often, some components or some linear combination of components are more critical than others.

EXAMPLE: It is critical that $x_1[k]$ be brought to zero quickly; $x_2[k]$ doesn't matter so much. We might take

$$J = \sum_{m=0}^{\infty} \left(x[m]^T \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix} x[m] + \rho u[m]^2 \right).$$

- More generally, we use the quadratic form

$$x^T[k] Q x[k]$$

where Q is an $n \times n$ weighting matrix.

PROPERTY I: We may assume that $Q = Q^T$. Why?

$$\underbrace{(x^T Q x)^T}_{\text{a scalar}} = x^T Q x.$$

- Therefore $x^T Q^T x = x^T Q x$ and $x^T Q x = x^T Q_{\text{sym}} x$ where $Q_{\text{sym}} = \frac{1}{2}(Q + Q^T)$ is the symmetric part of Q.

PROPERTY II: J should always be ≥ 0. That is, we require

$$x^T Q x \geq 0 \qquad \forall\, x \in \mathbb{R}^n$$

then Q is *positive semi-definite* (we write $Q \geq 0$, and $\lambda(Q) \geq 0$).

- If $x^T Q x > 0$ for all $x \neq 0$ then Q is *positive definite* (we write $Q > 0$, and $\lambda(Q) > 0$).

Vector derivatives

- In the following discussion we will often need to take derivatives of vector/ matrix quantities.

- This small dictionary should help: $a(x)$ and $b(x)$ are $m \times 1$ vector functions with respect to the vector x, y is some other vector and A is some matrix.

1. $\dfrac{\partial}{\partial x}\left[a^T(x)b(x)\right] = \left[\dfrac{\partial a(x)}{\partial x}\right]^T b(x) + \left[\dfrac{\partial b(x)}{\partial x}\right]^T a(x),$
2. $\dfrac{\partial}{\partial x}(x^T y) = y,$
3. $\dfrac{\partial}{\partial x}(x^T x) = 2x,$
4. $\dfrac{\partial}{\partial x}(x^T A y) = Ay,$
5. $\dfrac{\partial}{\partial x}(y^T A x) = A^T y,$
6. $\dfrac{\partial}{\partial x}(x^T A x) = (A + A^T)x,$
7. $\dfrac{\partial}{\partial x}\left[a^T(x)Qa(x)\right] = 2\left[\dfrac{\partial a(x)}{\partial x}\right]^T Qa(x)$, where Q is a symmetric matrix.

- This brings us back to our problem. . .

8.3: The discrete-time linear quadratic regulator problem

- Most generally, the discrete-time LQR problem is posed as minimizing

$$J_{i,N} = x^T[N]Px[N] + \sum_{k=i}^{N-1} \left[x^T[k]Qx[k] + u^T[k]Ru[k] \right],$$

  which may be interpreted as the total cost associated with the transition from state $x[i]$ to the goal state 0 at time N.

- $x^T[N]Px[N]$ is the penalty for "missing" the desired final state.
- $x^T[k]Qx[k]$ is the penalty on excessive state size.
- $u^T[k]Ru[k]$ is the penalty on excessive control effort. ($R = \rho$ if SISO).
- We require $P \geq 0$, $Q \geq 0$ and $R > 0$.
- To find optimum $u[k]$, we start at last step and work backwards.

$$J_{N-1,N} = x^T[N]Px[N] + x^T[N-1]Qx[N-1] + u^T[N-1]Ru[N-1].$$

- We express $x[N]$ as a function of $x[N-1]$ and $u[N-1]$ via the system dynamics

$$\begin{aligned} J_{N-1,N} &= (Ax[N-1] + Bu[N-1])^T P (Ax[N-1] + Bu[N-1]) \\ &\quad + x^T[N-1]Qx[N-1] + u^T[N-1]Ru[N-1] \\ &= x^T[N-1]A^T PAx[N-1] + u^T[N-1]B^T PBu[N-1] \\ &\quad + x^T[N-1]A^T PBu[N-1] + u^T[N-1]B^T PAx[N-1] \\ &\quad + x^T[N-1]Qx[N-1] + u^T[N-1]Ru[N-1]. \end{aligned}$$

- We minimize over all possible inputs $u[N-1]$ by differentiation

$$\begin{aligned} 0 = \frac{\partial J_{N-1,N}}{\partial u[N-1]} &= 2B^T PBu[N-1] + 2B^T PAx[N-1] + 2Ru[N-1] \\ &= 2\left(R + B^T PB\right) u[N-1] + 2B^T PAx[N-1]. \end{aligned}$$

- Therefore, $u^*[N-1] = -\underbrace{\left(R + B^T P B\right)^{-1} B^T P A}_{\text{Constant!}} x[N-1]$.
- The exciting point is that the optimal $u[N-1]$, with no constraints on its functional form, turns out to be a linear state feedback! To ease notation, define

$$K_{N-1} = \left(R + B^T P B\right)^{-1} B^T P A$$

such that

$$u^*[N-1] = -K_{N-1} x[N-1].$$

- Now, we can express the value of $J^*_{N-1,N}$ as

$$\begin{aligned} J^*_{N-1,N} = \Big[&\Big(Ax[N-1] - BK_{N-1}x[N-1]\Big)^T P \Big(Ax[N-1] \\ &-BK_{N-1}x[N-1]\Big) \\ &+x^T[N-1]Qx[N-1] + x^T[N-1]K_{N-1}^T R K_{N-1} x[N-1]\Big] \\ = x^T[N-1]\Big[&(A-BK_{N-1})^T P (A-BK_{N-1}) + Q \\ &+K_{N-1}^T R K_{N-1}\Big] x[N-1]. \end{aligned}$$

- Simplify notation once again by defining

$$P_{N-1} = (A-BK_{N-1})^T P (A-BK_{N-1}) + Q + K_{N-1}^T R K_{N-1},$$

so that

$$J^*_{N-1,N} = x^T[N-1] P_{N-1} x[N-1].$$

- To see that this notation makes sense, notice that

$$J_{N,N} = J^*_{N,N} = x^T[N] P x[N] \triangleq x^T[N] P_N x[N].$$

- Now, we take another step backwards and compute the cost $J_{N-2,N}$

$$J_{N-2,N} = J_{N-2,N-1} + J_{N-1,N}.$$

- Therefore, the optimal policy (via dynamic programming) is

$$J^*_{N-2,N} = J_{N-2,N-1} + J^*_{N-1,N}.$$

- To minimize this, we realize that $N-1$ is now the goal state and

$$J_{N-2,N-1} = (Ax[N-2] + Bu[N-2])^T P_{N-1} (Ax[N-2] + Bu[N-2])$$
$$+x^T[N-2]Qx[N-2] + u^T[N-2]Ru[N-2].$$

- We can find the best result just as before $u^*[N-2] = -K_{N-2}x[N-2]$ where $K_{N-2} = \left(R + B^T P_{N-1} B\right)^{-1} B^T P_{N-1} A$.
- In general, $u^*[k] = -K_k x[k]$ where $K_k = \left(R + B^T P_{k+1} B\right)^{-1} B^T P_{k+1} A$ and

$$P_k = (A - BK_k)^T P_{k+1}(A - BK_k) + Q + K_k^T R K_k,$$

- This difference equation for P_k has a starting condition that occurs at the final time, and is solved recursively backwards in time.

EXAMPLE: Simulate a feedback controller for the system

$$x[k+1] = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} x[k] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[k], \qquad x[0] = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$$

such that the cost criterion

$$J = x^T[10] \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} x[10] + \sum_{k=1}^{9} \left(x^T[k] \begin{bmatrix} 2 & 0 \\ 0 & 0.1 \end{bmatrix} x[k] + 2u^2[k] \right)$$

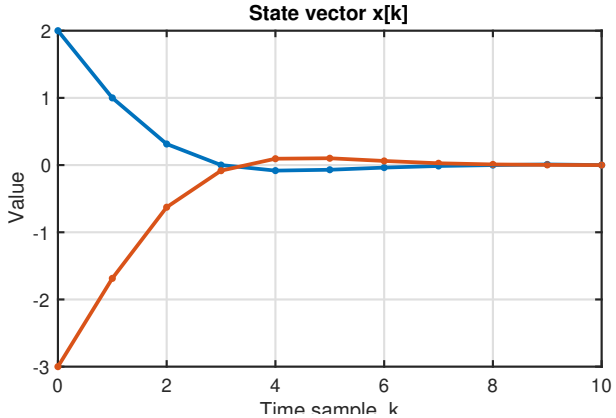
is minimized.

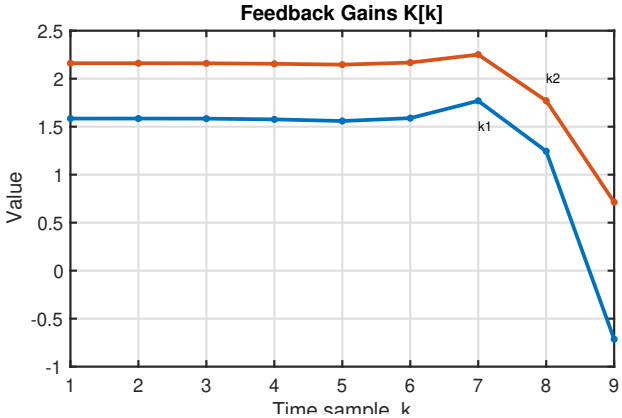
- From the problem, we gather that

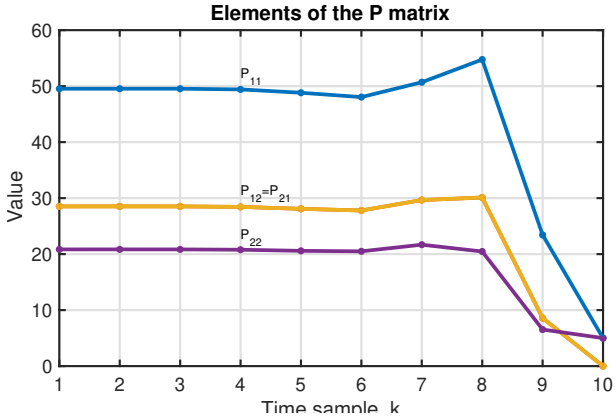
$$P_{10} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, \qquad Q = \begin{bmatrix} 2 & 0 \\ 0 & 0.1 \end{bmatrix}, \qquad R = [2].$$

- Iteratively, solve for K_9, P_9, K_8, P_8 and so forth down to K_1 and P_1. Then, $u[k] = -K_k x[k]$.

```
A=[2 1; -1 1]; B=[0; 1]; x0=[2; -3]; P=zeros(2,2,10); K=zeros(1,2,9);
x=zeros(2,1,11); x(:,:,1)=x0;
P(:,:,10)=[5 0; 0 5]; R=2; Q=[2 0; 0 0.1];
for i=9:-1:1,
  K(:,:,i)=inv(R+B'*P(:,:,i+1)*B)*B'*P(:,:,i+1)*A;
  P(:,:,i)=(A-B*K(:,:,i))'*P(:,:,i+1)*(A-B*K(:,:,i))+ ...
    Q+K(:,:,i)'*R*K(:,:,i);
end
for i=1:9,
  x(:,:,i+1)=A*x(:,:,i)-B*K(:,:,i)*x(:,:,i);
end
```







8.4: Infinite-horizon discrete-time LQR

- If we let $N \to \infty$, then P_k tends to a steady-state solution as $k \to 0$. Therefore, $K_k \to K$. This is clearly a much easier control design, and usually does just about as well.
- To find the steady-state P and K, we let $P_k = P_{k+1} = P_{ss}$ in the above equation.

$$P_{ss} = (A - BK)^T P_{ss}(A - BK) + Q + K^T RK$$

and

$$K = \left(R + B^T P_{ss} B\right)^{-1} B^T P_{ss} A$$

which may be combined to get

$$P_{ss} = A^T P_{ss} A - A^T P_{ss} B \left(R + B^T P_{ss} B\right)^{-1} B^T P_{ss} A + Q$$

which is called a discrete-time algebraic Riccati equation, and may be solved in MATLAB using `dare.m`

EXAMPLE: For the previous example (with a finite end time), the solution reached for P_1 was

$$P_1 = \begin{bmatrix} 49.5336 & 28.5208 \\ 28.5208 & 20.8434 \end{bmatrix}.$$

In MATLAB, `dare(A,B,Q,R)` for the same system gives

$$P_{ss} = \begin{bmatrix} 49.5352 & 28.5215 \\ 28.5215 & 20.8438 \end{bmatrix}.$$

So, we see that the system settles very quickly to steady-state behavior.

- There are many ways to solve the D.A.R.E., but when Q has the form $C^T C$, and the system is SISO, there is a simple method which yields

the optimal closed-loop eigenvalues directly. (Note, when $Q = C^T C$ we are minimizing the output energy $|y[k]|^2$).

Chang–Letov method

- The optimal eigenvalues are the roots of the equation

$$1 + \frac{1}{\rho} G^T(z^{-1}) G(z) = 0$$

which are inside the unit circle, where

$$G(z) = C(zI - A)^{-1}B + D.$$

(Proved later for the continuous-time version).

EXAMPLE: Consider $G(z) = \dfrac{1}{z-1}$ so

$$1 + \frac{\rho^{-1}}{(z-1)(z^{-1}-1)} = 0$$

$$2 + \rho^{-1} - z - z^{-1} =$$

$$z = 1 + \frac{1}{2\rho} \pm \sqrt{\frac{1}{4\rho^2} + \frac{1}{\rho}}.$$

- The locus of optimal pole locations for all ρ form a reciprocal root locus.

Reciprocal root locus in MATLAB (SISO)

- We want to plot the root locus

$$1 + \frac{1}{\rho} G^T(z^{-1}) G(z) = 0,$$

where

$$G(z) = C(zI - A)^{-1}B + D.$$

- We know how to plot a root locus of the form

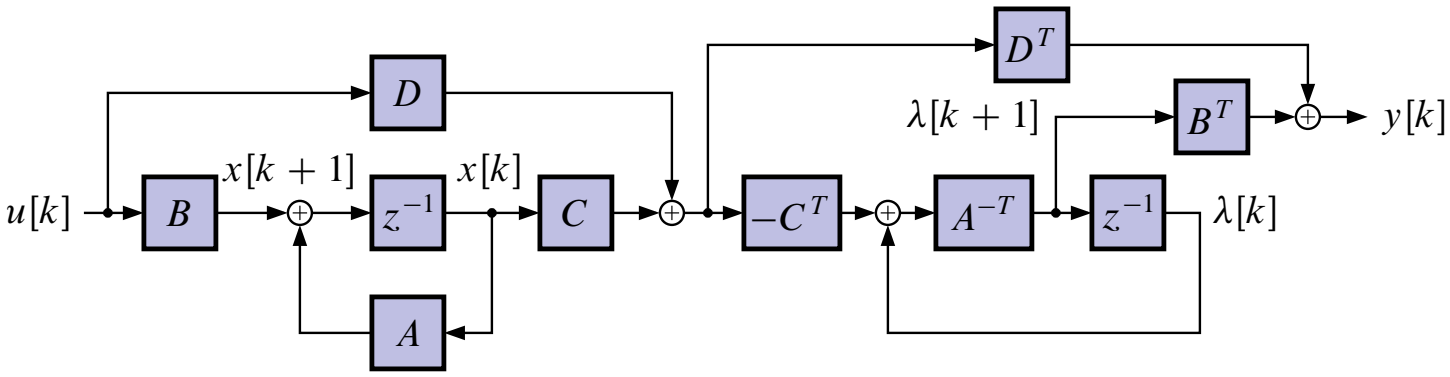
$$1 + KG'(z) = 0$$

so we need to find a way to convert $G^T(z^{-1})G(z)$ into $G'(z)$.

- We know that

$$\begin{aligned} G^T(z^{-1}) &= B^T\left(z^{-1}I - A^T\right)^{-1}C^T + D^T \\ &= B^T z\left(zI - A^{-T}\right)^{-1}(-A^{-T}C^T) + D^T. \end{aligned}$$

- Combining $G(z)$ and $G^T(z^{-1})$ in block-diagram form:



- The overall system has state

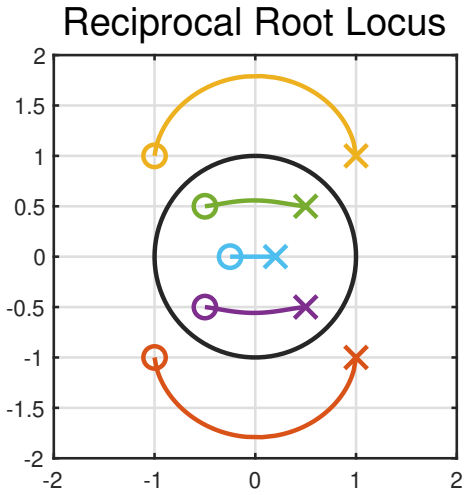
$$\left[\begin{array}{c} x[k+1] \\ \hline \lambda[k+1] \end{array}\right] = \left[\begin{array}{c|c} A & 0 \\ \hline -A^{-T}C^TC & A^{-T} \end{array}\right]\left[\begin{array}{c} x[k] \\ \hline \lambda[k] \end{array}\right] + \left[\begin{array}{c} B \\ \hline -A^{-T}C^TD \end{array}\right]u[k]$$

$$y[k] = \left[\begin{array}{c|c} -B^TA^{-T}C^TC + D^TC & B^TA^{-T} \end{array}\right]\left[\begin{array}{c} x[k] \\ \hline \lambda[k] \end{array}\right] + \left[D^TD - B^TA^{-T}C^TD\right]u[k].$$

```
function rrl(sys)
[A,B,C,D]=ssdata(sys);
bigA=[A zeros(size(A)); -inv(A)'*C'*C inv(A)'];
bigB=[B; -inv(A)'*C'*D];
bigC=[-B'*inv(A)'*C'*C+D'*C B'*inv(A)'];
```



```
bigD=-B'*inv(A)'*C'*D+D'*D;
rrlsys=ss(bigA,bigB,bigC,bigD,-1);
rlocus(rrlsys);
```



EXAMPLE: Let

$$G(z) = \frac{(z+0.25)(z^2+z+0.5)}{(z-0.2)(z^2-2z+2)}.$$

Note that $G(z)$ is unstable.

OBSERVATIONS: For the "expensive cost of control" case, stable poles remain where they are and unstable poles are mirrored into the unit disc. (They are not moved to be just barely stable, as we might expect!)

For the "cheap cost of control" case, poles migrate to the finite zeros of the transfer function, and to the origin (deadbeat control).

8.5: The continuous-time linear quadratic regulator problem (a–c)

- The continuous-time LQR problem is stated in a similar way, and there are corresponding results. We wish to minimize

$$J(x_o, u, t_o) = x^T(t_f) P_{t_f} x(t_f) + \int_{t_0}^{t_f} \left[x^T(t) Q x(t) + u^T(t) R u(t) \right] \mathrm{d}t.$$

- P_{t_f}, Q and R have same restrictions and interpretations as before.

RESULTS: The following are key results

- The *optimal* control is a linear (time varying) state feedback

$$u(t) = -R^{-1} B^T P(t) x(t).$$

- Symmetric psd matrix $P(t)$ satisfies (matrix) differential equation

$$\dot{P}(t) = P(t) B R^{-1} B^T P(t) - Q - P(t) A - A^T P(t),$$

with the boundary condition that $P(t_f) = P_{t_f}$. The differential equation runs *backwards* in time to find $P(t)$.

- If $t_f \to \infty$, $P(t) \to P_{ss}$ as $t \to 0$. Then,

$$0 = P_{ss} B R^{-1} B^T P_{ss} - Q - P_{ss} A - A^T P_{ss}.$$

This is the continuous-time algebraic Riccati equation, and may be solved in MATLAB using `care.m`; then,

$$u(t) = -R^{-1} B^T P_{ss} x(t),$$

which is a linear state feedback.

- There are many ways to solve the the C.A.R.E., but when Q has the form $C^T C$, and the system is SISO, a variant of the Chang–Letov method may be used:

- The optimal eigenvalues are the roots of the equation

$$1 + \frac{1}{\rho} G^T(-s) G(s) = 0$$

which are in the left-half plane, where

$$G(s) = C(sI - A)^{-1} B + D.$$

- The locus of all possible values of closed-loop optimal roots forms the symmetric root locus.

Solving the continuous-time LQR problem

1. Define the cost function.
2. Use Bellman's principle of optimality (dynamic programming).
3. Determine the Hamilton–Jacobi–Bellman equation.
4. Solve this equation (steps outlined later on).

Define the cost function

- We define the cost function we wish to minimize

$$J(x_o, u, t_o) = x^T(t_f) P_{t_f} x(t_f) + \int_{t_o}^{t_f} \left[x^T(t) Q x(t) + u^T(t) R u(t) \right] \mathrm{d}t$$

where $Q \geq 0$, $P_{t_f} \geq 0$ and $R > 0$.

- We define the optimal cost

$$V(x_o, t_o) = \min_{u(t)} J(x_o, u, t_o) \qquad \text{subject to } \dot{x}(t) = Ax(t) + Bu(t).$$

Invoke Bellman's principle of optimality

- We break the cost function into two pieces (where δt is small)

$$J(x_o, u, t_o) = x^T(t_f) P_{t_f} x(t_f) + \int_{t_o}^{t_o+\delta t} \left[x^T(t) Q x(t) + u^T(t) R u(t)\right] \mathrm{d}t + \int_{t_o+\delta t}^{t_f} \left[x^T(t) Q x(t) + u^T(t) R u(t)\right] \mathrm{d}t.$$

- From the Bellman equation we know that the optimal cost

$$V(x_o, t_o) = \min_{u(t)} \left\{ \int_{t_o}^{t_o+\delta t} \left[x^T(t) Q x(t) + u^T(t) R u(t)\right] \mathrm{d}t + V(x(t_o+\delta t), t_o+\delta t) \right\}.$$

- The minimum cost is the cost to go from $x(t_o)$ to $x(t_o+\delta t)$ plus the optimal cost to go from $x(t_o+\delta t)$ to $x(t_f)$. The latter part includes the terminal cost.

Determine the Hamilton–Jacobi–Bellman equation

- We evaluate $V(x(t_o+\delta t), t_o+\delta t)$ by computing its Taylor-series expansion around the point (x_o, t_o).

$$V(x(t_o+\delta t), t_o+\delta t) = V(x_o, t_o) + \left.\frac{\partial V(x,t)}{\partial t}\right|_{x_o,t_o} [(t_o+\delta t) - t_o] + \left.\frac{\partial V(x,t)}{\partial x}\right|_{x_o,t_o} [x(t_o+\delta t) - x(t_o)] + \text{h.o.t.}$$

- So, if δt is small

$$V(x_o, t_o) = \min_{u(t)} \left\{ \int_{t_o}^{t_o+\delta t} \left[x^T(t) Q x(t) + u^T(t) R u(t)\right] \mathrm{d}t + \underbrace{V(x_o, t_o) + \left.\frac{\partial V(x,t)}{\partial t}\right|_{x_o,t_o} \delta t}_{\text{Not functions of } u(t)} \right\}$$

$$+ \left.\frac{\partial V(x,t)}{\partial x}\right|_{x_o,t_o} [x(t_o+\delta t) - x(t_o)] \Bigg\}$$

$$= V(x_o,t_o) + \left.\frac{\partial V(x,t)}{\partial t}\right|_{x_o,t_o} \delta t$$

$$+ \min_{u(t)} \Bigg\{ \underbrace{\int_{t_o}^{t_o+\delta t} \left[x^T(t)Qx(t) + u^T(t)Ru(t)\right] \mathrm{d}t}_{\approx [x^T(t_o)Qx(t_o)+u^T(t_o)Ru(t_o)]\delta t}$$

$$+ \left.\frac{\partial V(x,t)}{\partial x}\right|_{x_o,t_o} \underbrace{[x(t_o+\delta t) - x(t_o)]}_{\approx [Ax(t_o)+Bu(t_o)]\delta t} \Bigg\}.$$

- Subtracting like terms from both sides and dividing by δt

$$0 = \left.\frac{\partial V(x,t)}{\partial t}\right|_{x_o,t_o} +$$

$$\min_{u(t)} \Bigg\{ \underbrace{[x^T(t_o)Qx(t_o) + u^T(t_o)Ru(t_o)] + \left.\frac{\partial V(x,t)}{\partial x}\right|_{x_o,t_o} [Ax(t_o) + Bu(t_o)]}_{\text{Hamiltonian}} \Bigg\}$$

- This is called the Hamilton–Jacobi–Bellman equation.
- To minimize the Hamiltonian (with respect to $u(t_o)$), take derivatives with respect to $u(t_o)$ and set to zero.

$$0 = \frac{\partial^T}{\partial u(t_o)} \left[[x^T(t_o)Qx(t_o) + u^T(t_o)Ru(t_o)] + \left.\frac{\partial V(x,t)}{\partial x}\right|_{x_o,t_o} [Ax(t_o) + Bu(t_o)] \right]$$

$$= 2Ru(t_o) + B^T \left.\frac{\partial V(x,t)^T}{\partial x}\right|_{x_o,t_o}.$$

- So,

$$u^*(t_o) = -\frac{1}{2} R^{-1} B^T \left.\frac{\partial V(x,t)^T}{\partial x}\right|_{x_o,t_o},$$

hence the need for R to be positive definite.

- We still need to determine $\left.\dfrac{\partial V(x,t)}{\partial x}\right|_{x_o,t_o}$.

  1. Show $V(z,t_o) = z^T P(t_o) z$ where $P(t_o)$ is symmetric, p.s.d.
  2. Use this result to compute the final desired term.

8.6: The continuous-time linear quadratic regulator problem (d)

Show that $V(z, t_o) = z^T P(t_o) z$

- The minimum cost-to-go starting in state z is a quadratic form in z. Can be shown in a number of steps. The main steps are:

  1. Show that the gradient operator on V (that is, ∇V) is linear.
  2. Integrate the (linear) gradient to get a quadratic form.

  We will develop a number of properties in order to prove these results.

PROPERTY I: For all scalars λ, $J(\lambda z, \lambda u, t_o) = \lambda^2 J(z, u, t_o)$ and therefore

$$V(\lambda z, t_o) = \lambda^2 V(z, t_o).$$

- Let $x(t)$ be the state that corresponds to an input $u(t)$ and an initial condition z. Then,

$$x(t) = e^{At} z + \int_0^t e^{A(t-\tau)} B u(\tau)\, \mathrm{d}\tau.$$

- Now, denote by $\tilde{x}(t)$ the state that corresponds to an input $\lambda u(t)$ and an initial condition λz. Then,

$$\tilde{x}(t) = \lambda \left[e^{At} z + \int_0^t e^{A(t-\tau)} B u(\tau)\, \mathrm{d}\tau \right] = \lambda x(t).$$

  Thus,

$$\begin{aligned} J(\lambda z, \lambda u, t_o) &= \lambda^2 x^T(t_f) P_{t_f} x(t_f) + \lambda^2 \int_{t_o}^{t_f} x^T(t) Q x(t) + u^T(t) R u(t)\, \mathrm{d}t \\ &= \lambda^2 J(z, u, t_o) \end{aligned}$$

  and

$$V(\lambda z, t_o) = \lambda^2 V(z, t_o).$$

PROPERTY II: Let u and $\tilde{u}$ be two input sequences, and let z and $\tilde{z}$ be two initial states. We will show that

$$J(z+\tilde{z}, u+\tilde{u}, t_o) + J(z-\tilde{z}, u-\tilde{u}, t_o) = 2J(z, u, t_o) + 2J(\tilde{z}, \tilde{u}, t_o)$$

by "plugging in" and collecting terms.

- Suppose

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad x_o = z;$$

$$\dot{\tilde{x}}(t) = A\tilde{x}(t) + B\tilde{u}(t), \qquad \tilde{x}_o = \tilde{z}.$$

- Adding (or subtracting) the above equations we obtain

$$\dot{x}(t) \pm \dot{\tilde{x}}(t) = A(x(t) \pm \tilde{x}(t)) + B(u(t) \pm \tilde{u}(t)), \qquad x_o \pm \tilde{x}_o = z \pm \tilde{z}.$$

Therefore, $x(t) \pm \tilde{x}(t)$ is the state that corresponds to an input $u(t) \pm \tilde{u}(t)$ and initial condition $z \pm \tilde{z}$ (respectively).

- Now, we plug in

$$\begin{aligned}
&J(z \pm \tilde{z}, u \pm \tilde{u}, t_o) \\
&= (x \pm \tilde{x})^T(t_f) P_{t_f} (x \pm \tilde{x})(t_f) \\
&\qquad + \int_{t_o}^{t_f} \left[(x \pm \tilde{x})^T(t) Q (x \pm \tilde{x})(t) + (u \pm \tilde{u})^T(t) R (u \pm \tilde{u})(t) \right] \mathrm{d}t \\
&= x^T(t_f) P_{t_f} x(t_f) \pm x^T(t_f) P_{t_f} \tilde{x}(t_f) \pm \tilde{x}^T(t_f) P_{t_f} x(t_f) + \tilde{x}^T(t_f) P_{t_f} \tilde{x}(t_f) \\
&\qquad + \int_{t_o}^{t_f} \Big[x^T(t) Q x(t) \pm \tilde{x}^T(t) Q x(t) \pm x^T(t) Q \tilde{x}(t) + \tilde{x}^T(t) Q \tilde{x}(t) \\
&\qquad\quad u^T(t) R u(t) \pm u^T(t) R \tilde{u}(t) \pm \tilde{u}^T(t) R u(t) + \tilde{u}^T(t) R \tilde{u}(t) \Big] \mathrm{d}t.
\end{aligned}$$

- Therefore,

$$J(z+\tilde{z}, u+\tilde{u}, t_o) + J(z-\tilde{z}, u-\tilde{u}, t_o) = 2J(z, u, t_o) + 2J(\tilde{z}, \tilde{u}, t_o).$$

PROPERTY III: Next, minimize the RHS with respect to $u(t)$ and $\tilde{u}(t)$. Conclude

$$V(z+\tilde{z},t_o)+V(z-\tilde{z},t_o)\geq 2V(z,t_o)+2V(\tilde{z},t_o).$$

- Minimizing

$$\begin{aligned}&\min_{u,\tilde{u}}\left\{J(z+\tilde{z},u+\tilde{u},t_o)+J(z-\tilde{z},u-\tilde{u},t_o)\right\}\\&=\min_{u}\left\{2J(z,u,t_o)\right\}+\min_{\tilde{u}}\left\{2J(\tilde{z},\tilde{u},t_o)\right\}.\end{aligned}$$

- Now,

$$\text{RHS}=2V(z,t_o)+2V(\tilde{z},t_o)$$

but

$$\text{LHS}\leq V(z+\tilde{z},t_o)+V(z-\tilde{z},t_o)$$

by the triangle inequality. Therefore,

$$V(z+\tilde{z},t_o)+V(z-\tilde{z},t_o)\geq 2V(z,t_o)+2V(\tilde{z},t_o).$$

PROPERTY IV: Apply the above inequality with $(z+\tilde{z})/2$ substituted for z and $(z-\tilde{z})/2$ substituted for $\tilde{z}$ to get:

$$V(z+\tilde{z},t_o)+V(z-\tilde{z},t_o)=2V(z,t_o)+2V(\tilde{z},t_o).$$

- Substitute as directed.

$$2V\left(\frac{z+\tilde{z}}{2},t_o\right)+2V\left(\frac{z-\tilde{z}}{2},t_o\right)\leq V(z,t_o)+V(\tilde{z},t_o).$$

- By scalar multiplication principle,

$$\frac{2}{4}V(z+\tilde{z},t_o)+\frac{2}{4}V(z-\tilde{z},t_o)\leq V(z,t_o)+V(\tilde{z},t_o).$$

- Multiply both sides by 2

$$V(z+\tilde{z},t_o)+V(z-\tilde{z},t_o)\leq 2V(z,t_o)+2V(\tilde{z},t_o),$$

and, combined with results of property III,

$$V(z+\tilde{z},t_o)+V(z-\tilde{z},t_o)=2V(z,t_o)+2V(\tilde{z},t_o).$$

PROPERTY V: Now we are getting somewhere. Recall that linearity requires superposition and scaling properties be met. First, we prove superposition of the gradient operator. Take partial derivatives of this equation with respect to z and $\tilde{z}$. Show that

$$\nabla V(z+\tilde{z})=\nabla V(z)+\nabla V(\tilde{z}).$$

- The gradient operator is defined as

$$\nabla f(x)=\frac{\partial f(x)^T}{\partial x}. \qquad \left(\text{Also, } \nabla f(ax)=\frac{\partial f(ax)^T}{\partial ax}.\right)$$

- Take partial derivatives of the equation with respect to z:

$$\frac{\partial V(z+\tilde{z},t_o)}{\partial z}+\frac{\partial V(z-\tilde{z},t_o)}{\partial z}=2\frac{\partial V(z,t_o)}{\partial z}+2\frac{\partial V(\tilde{z},t_o)}{\partial z}$$

$$\frac{\partial V(z+\tilde{z},t_o)}{\partial (z+\tilde{z})}\frac{\partial (z+\tilde{z})}{\partial z}+\frac{\partial V(z-\tilde{z},t_o)}{\partial (z-\tilde{z})}\frac{\partial (z-\tilde{z})}{\partial z}=2\nabla V(z,t_o)^T$$

$$\nabla V(z+\tilde{z},t_o)+\nabla V(z-\tilde{z},t_o)=2\nabla V(z,t_o).$$

- Take partial derivatives of the equation with respect to $\tilde{z}$:

$$\frac{\partial V(z+\tilde{z},t_o)}{\partial \tilde{z}}+\frac{\partial V(z-\tilde{z},t_o)}{\partial \tilde{z}}=2\frac{\partial V(z,t_o)}{\partial \tilde{z}}+2\frac{\partial V(\tilde{z},t_o)}{\partial \tilde{z}}$$

$$\frac{\partial V(z+\tilde{z},t_o)}{\partial (z+\tilde{z})}\frac{\partial (z+\tilde{z})}{\partial \tilde{z}}+\frac{\partial V(z-\tilde{z},t_o)}{\partial (z-\tilde{z})}\frac{\partial (z-\tilde{z})}{\partial \tilde{z}}=2\nabla V(\tilde{z},t_o)^T$$

$$\nabla V(z+\tilde{z},t_o)-\nabla V(z-\tilde{z},t_o)=2\nabla V(\tilde{z},t_o).$$

- Add the two results and divide by two to get

$$\nabla V(z+\tilde{z},t_o) = \nabla V(z,t_o) + \nabla V(\tilde{z},t_o).$$

PROPERTY VI: To show linearity of the gradient the last step we must perform is to show

$$\nabla V(\lambda z,t_o) = \lambda \nabla V(z,t_o).$$

- From the definition of the gradient,

$$\begin{aligned}\nabla V(\lambda z,t_o) &= \frac{\partial V(\lambda z,t_o)^T}{\partial(\lambda z)} \\ &= \frac{\partial \lambda^2 V(z,t_o)^T}{\lambda \partial z} \\ &= \lambda \nabla V(z,t_o).\end{aligned}$$

- So, the gradient is linear. This means that $\nabla V(z,t_o)$, which is a vector, is linear in z and hence has a matrix representation

$$\nabla V(z,t_o) = M(t_o)z$$

where $M(t_o) \in \mathbb{R}^{n\times n}$.

PROPERTY VII: We are nearly ready to integrate the gradient to show our desired result. First, we must show that

$$V(z,t_o) = V(0,t_o) + \int_0^1 \nabla V(\theta z,t_o)^T z \,\mathrm{d}\theta.$$

- Note that $V(z,t_o)$ is a scalar. Consider a scalar function $f(\theta)$. Then,

$$\int_0^1 \frac{\partial f(\theta)}{\partial \theta}\,\mathrm{d}\theta = f(1) - f(0).$$

- Let $f(\theta) = V(\theta z,t_o)$. Then,

$$V(z,t_o) - V(0,t_o) = \int_0^1 \frac{\partial V(\theta z, t_o)}{\partial \theta}\,\mathrm{d}\theta$$

$$V(z,t_o) = V(0,t_o) + \int_0^1 \frac{\partial V(\theta z, t_o)}{\partial(\theta z)}\frac{\partial \theta z}{\partial \theta}\,\mathrm{d}\theta$$

$$= V(0,t_o) + \int_0^1 \nabla V(\theta z, t_o)^T z\,\mathrm{d}\theta$$

PROPERTY VIII: Now, integrate away to show the desired result. Note that $V(0,t_o) = 0$.

$$V(z,t_o) = \int_0^1 (M(t_o)(\theta z))^T z\,\mathrm{d}\theta = \int_0^1 z^T M^T(t_o) z\theta\,\mathrm{d}\theta$$

$$= z^T M^T(t_o) z\,\frac{\theta^2}{2}\bigg|_0^1$$

$$= z^T M^T(t_o) z/2.$$

- Since $V(z,t_o)$ is a scalar, $V(z,t_o)^T = V(z,t_o) = z^T M(t_o) z/2$. Averaging our two (identical) results,

$$V(z,t_o) = z^T \left(\frac{M(t_o) + M^T(t_o)}{4}\right) z.$$

- Therefore, $P(t_o) = \dfrac{M(t_o) + M^T(t_o)}{4}$. Also, $P(t_o) \geq 0$ since $J(z,u,t_o) \geq 0$ for all u, z. Thus

$$V(z,t_o) = \min_u J(z,u,t_o) = z^T P(t_o) z \geq 0 \quad \forall\, z,$$

and we have (finally) proven the desired result.

The optimal $u^*(t)$ and differential Riccati equation

- Because $V(x,t) = x^T P(t) x$, $\left.\dfrac{\partial V(x,t)}{\partial x}\right|_{x_o,t_o} = 2x^T(t_o)P^T(t_o)$. We can now state

$$u^*(t) = -\underbrace{R^{-1}B^T P(t)}_{K(t)} x(t)$$

so we see that the *optimum* control, with no a priori constraints on the structure of the $u(t)$ signal, is a (time varying) linear state feedback.

- We need to determine $P(t)$. Note that in the Hamilton–Jacobi–Bellman equation we have yet to determine

$$\left.\frac{\partial V(x,t)}{\partial t}\right|_{x_o,t_o} = \left.\frac{\partial}{\partial t} x^T P(t) x\right|_{x_o,t_o} = x_o^T \dot{P}(t_o) x_o.$$

- Substitute all results, including optimum $u^*(t)$

$$0 = x^T(t_o)\dot{P}(t_o)x(t_o) + x^T(t_o)Qx(t_o) + x^T(t_o)P(t_o)BR^{-1}B^T P(t_o)x(t_o)$$
$$+2x^T(t_o)P(t_o)Ax(t_o) - 2x^T(t_o)P(t_o)BR^{-1}B^T P(t_o)x(t_o).$$

- This expression is valid for all t_o. Also note that we can write

$$2x^T(t_o)P(t_o)Ax(t_o) = x^T(t_o)P(t_o)Ax(t_o) + x^T(t_o)A^T P(t_o)x(t_o),$$

so

$$0 = x^T(t)\left[\dot{P}(t) + Q - P(t)BR^{-1}B^T P(t) + P(t)A + A^T P(t)\right]x(t)$$

which is true for any $x(t)$. Therefore,

$$\dot{P}(t) = P(t)BR^{-1}B^T P(t) - P(t)A - A^T P(t) - Q$$

which is called the differential (matrix) Riccati equation.

- This is a nonlinear differential equation with boundary condition $P(t_f) = P_{t_f}$, solved *backward* in time.

Steady-state solution

- As the differential equation for $P(t)$ is simulated backward in time from the terminal point, it tends toward steady-state values as $t \to 0$.

It is much simpler to approximate the optimal control gains as a constant set of gains calculated using P_{ss}.

$$0 = P_{ss}BR^{-1}B^T P_{ss} - P_{ss}A - A^T P_{ss} - Q.$$

This is called the *Algebraic Riccati Equation.* In MATLAB, `care.m`

8.7: Solving the differential Riccati equation via simulation

- The differential Riccati equation may be solved numerically by integrating the matrix differential equation

$$\dot{P}(t) = P(t)BR^{-1}B^T P(t) - P(t)A - A^T P(t) - Q$$

  backward in time.

- The problem we discover is that MATLAB's integration routines `ode45.m` will work only on vector differential equations, not matrix differential equations such as this.

- The Kronecker product $\otimes$ comes to the rescue once again, along with the matrix stacking operator. We can write the above matrix differential equation as a vector differential equation:

$$\dot{P}_{st} = \left(A^T \otimes I + I \otimes A^T\right) P_{st} + Q_{st} - \left(P^T \otimes P\right)\left(BR^{-1}B^T\right)_{st}.$$

  A "$-$" sign has been introduced in order for the forward-time `ode45.m` (for example) to work on the backward-time equation.

- In MATLAB

```
pdot=(kron(A',eye(size(A)) + kron(eye(size(A)),A')) ...
      *st(P) + st(Q) - kron(P',P)*st(B*inv(R)*B');

function col=st(m) % stack subfunction
col=reshape(m,prod(size(m)),1);

function mat=unst(v) % unstack subfunction
mat=reshape(v,sqrt(length(v)),sqrt(length(v)));
```

EXAMPLE: Consider the continuous-time system

$$\dot{x}(t) = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \qquad \text{and} \qquad y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(t).$$

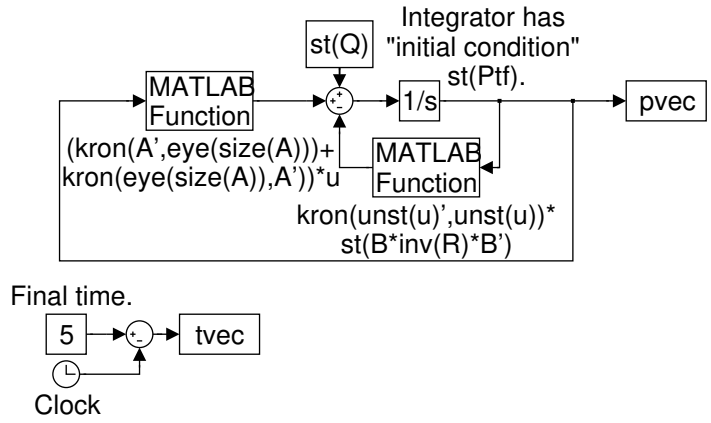
- Solve the differential matrix Riccati equation that results in the control signal that minimizes the cost function

$$J = x^T(5)\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} x(5) + \int_0^5 [y^T(t)y(t) + u^T(t)u(t)]\,\mathrm{d}t.$$

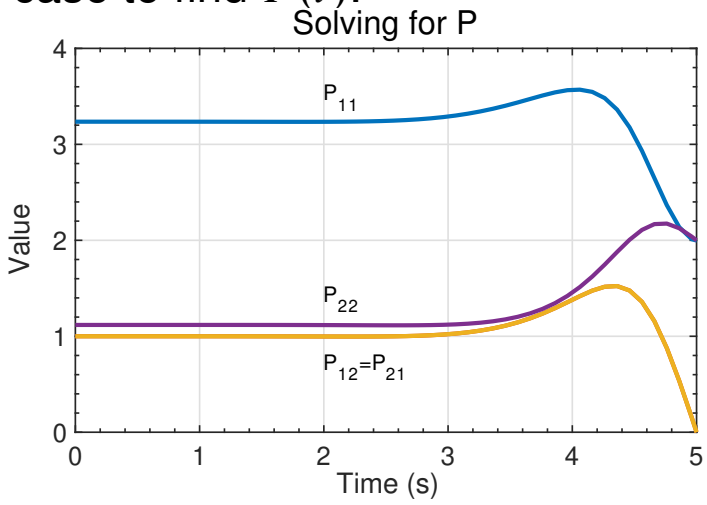
- First, note that the open-loop system is unstable, with poles at 0 and 1. It is controllable and observable.
- The cost function is written in terms of $y(t)$ but not $x(t)$. However, since there is no feedthrough term, we can also write it as

$$J = x^T(5)\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} x(5) + \int_0^5 \left[x^T(t)C^TCx(t) + u^T(t)u(t)\right]\,\mathrm{d}t.$$

  This is a common "trick".
- Therefore, the penalty matrices are $Q = C^TC$ and $R = \rho = 1$.
- We can simulate the finite horizon case to find $P(t)$.



To plot: plot(tvec.signals.values,pvec.signals.values)



- We can also solve the infinite-horizon case (analytically, for this example). Consider the A.R.E.

$$0 = A^TP + PA + C^TC - PBR^{-1}B^TP$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} -$$

$$\begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

$$= \begin{bmatrix} p_{11}+2p_{12} & p_{12}+2p_{22} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} p_{11}+2p_{12} & 0 \\ p_{12}+2p_{22} & 0 \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} p_{11}^2 & p_{11}p_{12} \\ p_{11}p_{12} & p_{12}^2 \end{bmatrix}.$$

- This matrix equality represents a set of three simultaneous equations (because P is symmetric). They are:

$$2p_{11} - p_{11}^2 + 4p_{12} = 0$$

$$p_{12} + 2p_{22} - p_{11}p_{12} = 0$$

$$1 - p_{12}^2 = 0.$$

- The final equation gives us $p_{12} = \pm 1$. If we select $p_{12} = -1$ then the first equation will have complex roots (bad). So, $p_{12} = 1$.
- Then, $p_{11} = 1 \pm \sqrt{5}$. If $p_{11} = 1 - \sqrt{5}$ then P cannot be positive definite. Therefore, $p_{11} = 1 + \sqrt{5} = 3.236$.
- Finally, we get $p_{22} = \sqrt{5}/2 = 1.118$.
- These are the same values as the steady-state solution found by integrating the differential Riccati equation.
- The static feedback control signal is

$$u(t) = -R^{-1}B^T P_{ss} x(t) = -\begin{bmatrix} 3.236 & 1 \end{bmatrix} x(t).$$

For this feedback, the closed-loop poles are at $-\frac{\sqrt{5}}{2} \pm \frac{\sqrt{3}}{2} j$ (stable).

8.8: Continuous-time systems and Chang–Letov (SISO only)

- For a SISO system, we can easily plot the locus of closed-loop poles.
- Tradeoff between control effort and output error is evident.
- Consider the infinite-horizon LQR problem with $Q = C^T C$, $C \in \mathbb{R}^{1\times n}$ and $R = \rho$, $\rho \in \mathbb{R}$.
- The cost function is then $J = \int_0^\infty y^2(t) + \rho u^2(t)\,\mathrm{d}t$.
- The algebraic Riccati equation becomes

$$A^T P + PA - PB\rho^{-1}B^T P + C^T C = 0$$

or

$$C^T C = P(sI - A) + (-sI - A^T)P + PB\rho^{-1}B^T P.$$

- Multiply on left by $B^T(-sI - A^T)^{-1}$ and on right by $(sI - A)^{-1}B\rho^{-1}$.

$$\frac{1}{\rho}\left\{B^T(-sI - A^T)^{-1}C^T\right\}\left\{C(sI - A)^{-1}B\right\} =$$

$$B^T(-sI - A^T)^{-1}\underbrace{PB\rho^{-1}}_{K^T} + \underbrace{\rho^{-1}B^T P}_{K}(sI - A)^{-1}B$$

$$+B^T(-sI - A^T)^{-1}PB\rho^{-2}B^T P(sI - A)^{-1}B.$$

- The left-hand side is $\frac{1}{\rho}G^T(-s)G(s)$. Add 1 to both sides and collect

$$(1 + K(sI - A)^{-1}B)(1 + K(-sI - A)^{-1}B) = 1 + \frac{1}{\rho}G^T(-s)G(s).$$

- Note that all terms are scalars.

FACT: Consider the determinant of a block matrix (we will not prove this fact here, but the result may be found in many linear algebra books):

$$\det\left[\begin{array}{c|c} \bar{A} & \bar{B} \\ \hline \bar{C} & \bar{D} \end{array}\right] = \det(\bar{A})\det(\bar{D} - \bar{C}\,\bar{A}^{-1}\bar{B}).$$

FACT: $1 + K(sI - A)^{-1}B = \dfrac{\det(sI - A + BK)}{\det(sI - A)} = \dfrac{\chi_{cl}(s)}{\chi_{ol}(s)}$.

PROOF: Consider the block matrix

$$M_1 = \left[\begin{array}{c|c} sI - A & B \\ \hline -K & 1 \end{array}\right]$$

$$\begin{aligned} \det(M_1) &= \det(sI - A)\det(1 + K(sI - A)^{-1}B) \\ &= \det(sI - A)(1 + K(sI - A)^{-1}B). \end{aligned}$$

- Now, consider the product of matrices (where $r \neq 0$)

$$\begin{aligned} M_2 &= \left[\begin{array}{c|c} sI - A & B \\ \hline -K & 1 \end{array}\right]\left[\begin{array}{c|c} I & p \\ \hline K & r \end{array}\right] \\ &= \left[\begin{array}{c|c} sI - A + BK & (sI - A)p + Br \\ \hline 0 & -Kp + r \end{array}\right] \end{aligned}$$

$$\begin{aligned} \det(M_2) &= \det(sI - A + BK)\det(-Kp + r) \\ &= \left[\det(sI - A)(1 + K(sI - A)^{-1}B)\right]\det(-Kp + r), \end{aligned}$$

or

$$1 + K(sI - A)^{-1}B = \frac{\det(sI - A + BK)}{\det(sI - A)} = \frac{\chi_{cl}(s)}{\chi_{ol}(s)}.$$

- So, from before, we have

$$\frac{\chi_{cl}(s)\chi_{cl}(-s)}{\chi_{ol}(s)\chi_{ol}(-s)} = 1 + \frac{1}{\rho}G^T(-s)G(s) \triangleq \Delta(s).$$

- Therefore $\Delta(s) = 0$ requires $\chi_{cl}(s) = 0$ or $\chi_{cl}(-s) = 0$. LQR requires that $\chi_{cl}(s)$ be Hurwitz (stable), so we have the conclusion:

  Closed-loop poles are the LHP zeros of $1 + \frac{1}{\rho} G^T(-s)G(s)$.

Symmetric root locus in MATLAB

- We want to plot the root locus

$$1 + \frac{1}{\rho} G^T(-s)G(s) = 0.$$

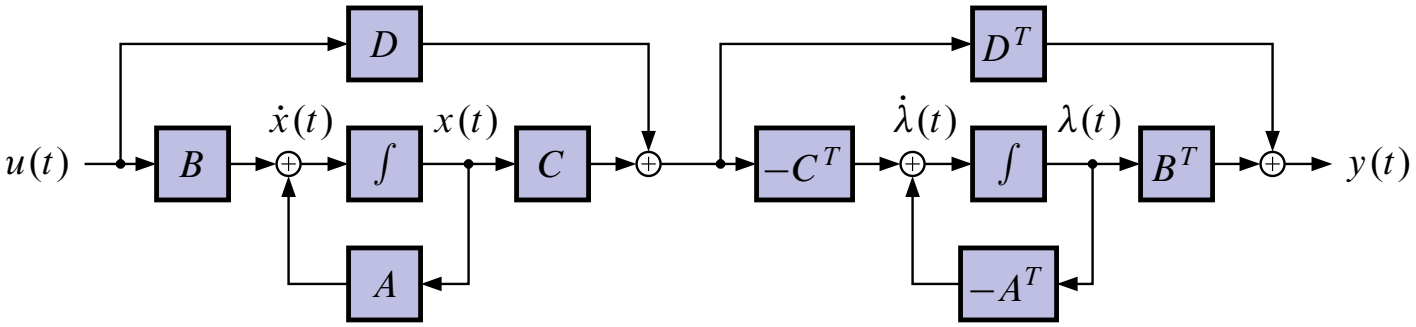
- We need to find a way to represent $G^T(-s)G(s)$ as a state-space system in MATLAB.

$$G(s) = C(sI - A)^{-1}B + D$$

and

$$\begin{aligned} G^T(-s) &= B^T\left(-sI - A^T\right)^{-1} C^T + D^T \\ &= B^T(sI - (-A^T))^{-1}(-C)^T + D^T. \end{aligned}$$

- This can be represented in block-diagram form as:

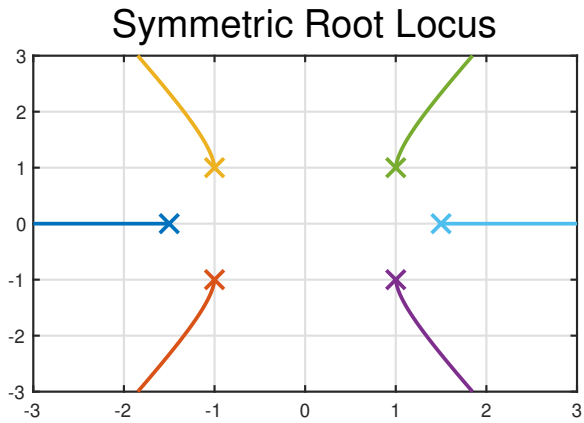


- The overall system has state

$$\left[\begin{array}{c} \dot{x}(t) \\ \hline \dot{\lambda}(t) \end{array}\right] = \left[\begin{array}{c|c} A & 0 \\ \hline -C^T C & -A^T \end{array}\right] \left[\begin{array}{c} x(t) \\ \hline \lambda(t) \end{array}\right] + \left[\begin{array}{c} B \\ \hline -C^T D \end{array}\right] u(t)$$

$$y(t) = \left[\begin{array}{c|c} D^T C & B^T \end{array}\right] \left[\begin{array}{c} x(t) \\ \hdashline \lambda(t) \end{array}\right] + D^T D u(t).$$

```
function srl(sys)
[A,B,C,D]=ssdata(sys);
bigA=[A zeros(size(A)); -C'*C -A'];
bigB=[B; -C'*D];
bigC=[D'*C B'];
bigD=D'*D;
srlsys=ss(bigA,bigB,bigC,bigD);
rlocus(srlsys);
```



EXAMPLE: Let

$$G(s) = \frac{1}{(s-1.5)(s^2+2s+2)}.$$

Note that $G(s)$ is unstable.

EXAMPLE: Multivariable control via LQR. Place poles of the MIMO MagLev using LQR for

$$Q = \text{diag}(1,3,1,3) \qquad R = \text{diag}(100,100).$$

The poles end up at -179.3358, -7.0858, -101.2163, -3.6269. Place poles at these locations using the Lyapunov method (from section 6) as well, and compare $u(t)$ and $x(t)$.