LINEAR QUADRATIC REGULATOR

3.1: Cost functions; deterministic LQR problem

Cost functions

- 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

- Optimizing a tradeoff like this can be turned into a mathematical optimization by defining a cost function that must be minimized.
- The cost function used to compare controllers should include a measure of size of output errors and size of control

$$J = \int_0^{t_f} z^T(t) W(t) z(t)\, \mathrm{d}t = \|z(t)\|^2_{W(t),[0,t_f]},$$

where $z(t)$ is assumed to include both output errors and control inputs and $W(t)$ is psd at all times.

- The cost function may be expanded to give more detailed description

$$J = \int_0^{t_f} \left[e^T(t) \;\vdots\; u^T(t) \right] \left[\begin{array}{c:c} W_1(t) & 0 \\ \hdashline 0 & W_2(t) \end{array} \right] \left[\begin{array}{c} e(t) \\ \hdashline u(t) \end{array} \right] \mathrm{d}t$$

$$= \int_0^{t_f} e^T(t) W_1(t) e(t) + u^T(t) W_2(t) u(t)\, \mathrm{d}t$$

where $W_1(t)$ and $W_2(t)$ are both psd.

- This cost function is not as general, but often sufficient.
- Note we can write $e(t) = r(t) - C_y x(t)$.
- Then,

$$J = \int_0^{t_f} (r(t) - C_y x(t))^T W_1(t)(r(t) - C_y x(t)) + u^T(t) W_2(t) u(t)\, \mathrm{d}t.$$

- Often, we can formulate a goal such that we want to drive $x(t) \to 0$ or some linear combination of states to zero. This is called a regulator.
- Then,

$$J = \int_0^{t_f} x^T(t) Q(t) x(t) + u^T(t) R(t) u(t)\, \mathrm{d}t,$$

where $Q(t) = C_y^T W_1(t) C_y$ and $R(t) = W_2(t)$, for example.

- Sometimes, final output error is more critical than intermediate state errors. *e.g.,* flight of a missile

$$J = \left[r(t_f) - C_y x(t_f)\right]^T V \left[r(t_f) - C_y x(t_f)\right] + \int_0^{t_f} u^T(t) R u(t)\, \mathrm{d}t.$$

- Value of cost function depends on reference input applied, disturbance applied, initial conditions, final conditions, constraints on state or control.
  - Collectively, these are known as test conditions. *e.g.,* nominal or worst-case.
  - For valid comparison between controllers, same test conditions must be used.

The deterministic LQR problem

- In our case, the deterministic linear quadratic regulator (LQR) problem assumes that we have full state information available, and

that we desire to minimize the cost function

$$J(x_k, u_k) = x_N^T H_d x_N + \sum_{k=0}^{N-1} [x_k^T Q_d x_k + u_k^T R_d u_k],$$

subject to $x_{k+1} = Ax_k + Bu_k$ (if the system dynamics are discrete), or

$$J(x(t), u(t)) = x^T(t_f) H x(t_f) + \int_0^{t_f} x^T(t) Q x(t) + u^T(t) R u(t)\, \mathrm{d}t$$

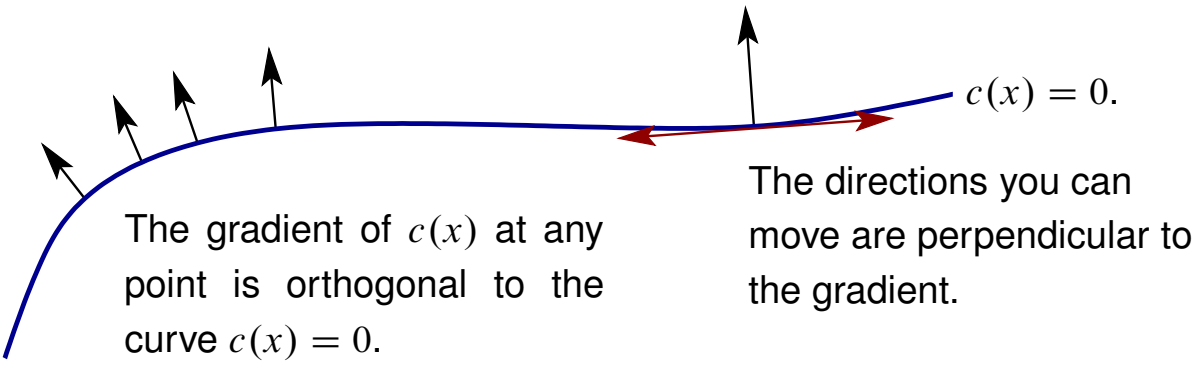
subject to $\dot{x}(t) = Ax(t) + Bu(t)$ (if system dynamics are continuous).

- $x_N^T H_d x_N$ or $x^T(t_f) H x(t_f)$ is the penalty for "missing" the desired final state.
- $x_k^T Q_d x_k$ or $x^T(t) Q x(t)$ is the penalty on excessive state size.
- $u_k^T R_d u_k$ or $u^T(t) R u(t)$ is the penalty on excessive control effort.

- Matrices H, H_d, Q, Q_d and R, R_d are symmetric matrices that put more (or less) cost on each term in the cost function.
  - We require $H, H_d \geq 0$, $Q, Q_d \geq 0$ and $R, R_d > 0$.
  - The weighting of these matrices is relative to each other; if we multiply everything by 2, the solution does not change (but the minimum cost is doubled).
- The solution is an input trajectory, either $u(t)$ or u_k.
- There are two approaches to optimization problems such as this:
  - Calculus of variations *[Burl];*
  - Dynamic programming *[Bay].*
- We will look at both, since both are common in the literature.
  - We'll use calculus of variations to derive the continuous-time result.
  - We will review the dynamic-programming results from ECE5520 for the discrete-time problem.

Lagrange multipliers

- The LQR optimization is subject to the constraint imposed by the system dynamics: e.g., $\dot{x}(t) = Ax(t) + Bu(t)$.
- Without the constraint, we might consider optimizing the cost function by using its gradient, ∇J.
  - The gradient at any location points in the direction of the steepest increase in the function.
  - Following the *negative* gradient will lead us down to a minimum of the function.
  - Whenever the gradient is zero, we are either at a minimum, a maximum, or a saddle point.
  - Second-derivative tests can tell us what is the case at any particular location.
- Now, suppose that we must minimize a function $J(x)$ subject to the constraint $c(x) = 0$. Plotting the constraint itself:



- If ∇J has any component aligned with the constraint, then the constrained cost can be reduced by moving along the constraint in the negative direction of that component.
- However, when ∇J is perpendicular to the constraint, that point is a local minimum, maximum, or saddle point.

- Since the gradient of the constraint is perpendicular to the constraint, we have that $\nabla J = -\lambda \nabla c$, and the Lagrange multiplier λ is the proportionality factor.
- So, we satisfy a constrained optimization when $\nabla(J + \lambda c) = 0$.
  - We do so by making an augmented cost $J_a(x, \lambda) = J(x) + \lambda c(x)$,
  - Taking partial derivatives of $J_a(x, \lambda)$, and
  - Setting these derivatives to zero.
- We will use this approach to solve the LQR problem.

3.2: Optimization via calculus of variations

- Differentiation is primary tool for optimization w.r.t. a (scalar) variable.
- Our objective requires differentiation of a real scalar cost function w.r.t. the control input (a function of time).
- Can be accomplished by using a generalization of the differential called the variation.
- Consider: The real scalar function of a scalar $J(x)$ has a local minimum at x^* iff

$$J(x^* + \delta x) \geq J(x^*)$$

for all δx sufficiently small. Equivalently,

$$\Delta J(x^*, \delta x) = J(x^* + \delta x) - J(x^*) \geq 0.$$

- $\Delta J(x^*, \delta x)$ is called the *increment of* J. Expand via Taylor series

$$\begin{aligned}\Delta J(x^*, \delta x) &= J(x^* + \delta x) - J(x^*) \\ &= \frac{\mathrm{d}J(x^*)}{\mathrm{d}x}\delta x + \frac{\mathrm{d}^2 J(x^*)}{\mathrm{d}x^2}\delta x^2 + \text{h.o.t.} \geq 0.\end{aligned}$$

- Above, δx is called the differential of x, and the term linear in δx (which is $[\mathrm{d}J(x^*)/\mathrm{d}x]\delta x$ in this case) is called the differential of J.
- When dealing with a functional (a real scalar function of functions), δx is called the *variation of* x and the term linear in δx is called the *variation of* J and is denoted $\delta J(x^*, \delta x)$. So,

$$\Delta J(x^*, \delta x) = \delta J(x^*, \delta x) + \text{h.o.t.} \geq 0.$$

- The variation of J is a generalization of the differential and can be applied to the optimization of a functional.

NECESSARY CONDITION OF OPTIMALITY: The variation of J is zero at x^* for all δx.

EXAMPLE: Consider

$$J(x) = x^2 + 6x + 8.$$

1. Find the increment ΔJ.
2. Simplify, and extract the variation δJ.
3. Set $\delta J = 0$ and solve.

- The increment of J is

$$\begin{aligned}\Delta J(x, \delta x) &= (x + \delta x)^2 + 6(x + \delta x) + 8 - (x^2 + 6x + 8) \\ &= x^2 + 2x\delta x + \delta x^2 + 6x + 6\delta x + 8 - x^2 - 6x - 8 \\ &= (2x + 6)\delta x + \delta x^2.\end{aligned}$$

- For optimality, the variation of J must be zero

$$\delta J(x, \delta x) = (2x + 6)\delta x = 0$$

  for all δx ➠ Therefore, $x = -3$.

- Using standard calculus, we would have $\dfrac{\mathrm{d}J}{\mathrm{d}x} = 2x + 6 = 0$ and would get the answer more quickly. However, the calculus of variations can be directly extended to the optimization of a functional.

Lagrange multipliers (again)

- The optimal control problem is a constrained minimization problem.
- Want minimum cost subject to dynamics of plant $\dot{x} = Ax + Bu$.
- The calculus of variations applies to unconstrained minimization problems.

- Lagrange multipliers convert a constrained minimization problem to a higher-order unconstrained minimization problem. Can then use calculus of variations.
- Consider minimizing $J(x)$, $x \in \mathbb{R}^n$, subject to the constraint $c(x) = 0$.
  - The differential in J must be parallel to the differential in c for an optimal solution.
- That is, generalizing our prior solution (and dividing both sides by δx), we must have
$$\frac{\delta J(x^*)}{\delta x} + \lambda \frac{\delta c(x^*)}{\delta x} = 0$$
for some scalar λ. Can also be written as the solution to an unconstrained augmented cost function
$$J_a(x, \lambda) = J(x) + \lambda c(x).$$
- Note that we are just adding zero to our cost function in a clever way that enforces the constraint. When $c(x)$ is a vector, we use
$$J_a(x, \lambda) = J(x) + \lambda^T c(x).$$

EXAMPLE: We wish to minimize $J(x) = x_1^2 + x_2^2$ subject to the constraint that $c(x) = 2x_1 + x_2 + 4 = 0$.

- The augmented cost function is
$$J_a(x, \lambda) = x_1^2 + x_2^2 + \lambda(2x_1 + x_2 + 4).$$
- The increment of the augmented cost function is

$$\begin{aligned}\Delta J_a(x,\lambda) &= J_a(x+\delta x,\lambda+\delta\lambda) - J_a(x,\lambda)\\ &= (x_1+\delta x_1)^2 + (x_2+\delta x_2)^2 + (\lambda+\delta\lambda)\Big[2(x_1+\delta x_1)\\ &\qquad +(x_2+\delta x_2)+4\Big] - x_1^2 - x_2^2 - \lambda[2x_1+x_2+4]\\ &= (2x_1+2\lambda)\delta x_1 + (2x_2+\lambda)\delta x_2 + (2x_1+x_2+4)\delta\lambda\\ &\qquad +\delta x_1^2 + \delta x_2^2 + 2\delta\lambda\delta x_1 + \delta\lambda\delta x_2.\end{aligned}$$

- Then,

$$\delta J_a(x,\lambda,\delta x,\delta\lambda) = (2x_1+2\lambda)\delta x_1 + (2x_2+\lambda)\delta x_2 + (2x_1+x_2+4)\delta\lambda = 0.$$

- Therefore,

$$\begin{aligned}\frac{\delta J_a}{\delta x_1} &= 2x_1+2\lambda = 0\\ \frac{\delta J_a}{\delta x_2} &= 2x_2+\lambda = 0\\ \frac{\delta J_a}{\delta\lambda} &= 2x_1+x_2+4 = 0.\end{aligned}$$

- Notice that the unconstrained minimization problem simply has $2x_1 = 2x_2 = 0$. Adding the constraint is seen by the added "λ" terms plus the $\delta J_a/\delta\lambda$ term.
- Solving these three equations and three unknowns gives

$$\begin{bmatrix} x_1 & x_2 & \lambda \end{bmatrix} = \begin{bmatrix} -1.6 & -0.8 & 1.6 \end{bmatrix}.$$

3.3: The LQR problem solved via calculus of variations

- For the continuous-time LQR problem, we wish to minimize

$$J(x(t),u(t)) = \frac{1}{2}x^T(t_f)Hx(t_f) + \frac{1}{2}\int_0^{t_f} x^T(t)Qx(t) + u^T(t)Ru(t)\,\mathrm{d}t$$

  subject to the constraint that $\dot{x}(t) = Ax(t) + B_u u(t)$.

- We add the constraint to the integral term to get

$$J_a(x(t),u(t),\lambda(t)) = J(x(t),u(t)) + \int_0^{t_f} \lambda^T(t)[Ax(t) + B_u u(t) - \dot{x}(t)]\,\mathrm{d}t.$$

$$= \frac{1}{2}x^T(t_f)Hx(t_f) + \int_0^{t_f} \left(\frac{1}{2}x^T(t)Qx(t) + \frac{1}{2}u^T(t)Ru(t)\right.$$

$$\left. +\lambda^T(t)[Ax(t) + B_u u(t) - \dot{x}(t)]\right)\mathrm{d}t.$$

- The Lagrange multiplier $\lambda(t)$ is often referred to as the costate for reasons that will become clear later.

- We find the optimal control by first forming the increment of J_a.

$$\Delta J_a(x,u,\lambda,\delta x,\delta u,\delta\lambda)$$

$$= J_a(x+\delta x, u+\delta u, \lambda+\delta\lambda) - J_a(x,u,\lambda)$$

$$= \frac{1}{2}\left[x(t_f)+\delta x(t_f)\right]^T H\left[x(t_f)+\delta x(t_f)\right]$$

$$+\int_0^{t_f} \frac{1}{2}(x+\delta x)^T Q(x+\delta x) + \frac{1}{2}(u+\delta u)^T R(u+\delta u)$$

$$+(\lambda+\delta\lambda)^T\left[A(x+\delta x) + B_u(u+\delta u) - (\dot{x}+\delta\dot{x})\right]\mathrm{d}t$$

$$-\frac{1}{2}x^T(t_f)Hx(t_f) - \int_0^{t_f} \frac{1}{2}x^TQx + \frac{1}{2}u^TRu + \lambda^T(Ax + B_u u - \dot{x})\,\mathrm{d}t,$$

where time dependence has been omitted to simplify the expression.

- Expand, noting that $x^T Q\delta x = (x^T Q\delta x)^T = \delta x^T Q^T x = \delta x^T Q x$,

$$\begin{aligned}\Delta J_a = &\frac{1}{2}\delta x^T(t_f)H\delta x(t_f)\\ &+\int_0^{t_f}\frac{1}{2}\delta x^T Q\delta x+\frac{1}{2}\delta u^T R\delta u+\delta\lambda^T(A\delta x+B_u\delta u-\delta\dot{x})\,\mathrm{d}t\\ &+x^T(t_f)H\delta x(t_f)+\int_0^{t_f}x^T Q\delta x+u^T R\delta u+\delta\lambda^T(Ax+B_u u-\dot{x})\\ &+\lambda^T(A\delta x+B_u\delta u-\delta\dot{x})\,\mathrm{d}t.\end{aligned}$$

- The variation of J_a must equal zero

$$\begin{aligned}\delta J_a(x,u,\lambda,\delta x,\delta u,\delta\lambda) = &x^T(t_f)H\delta x(t_f)\\ &+\int_0^{t_f}(x^T Q+\lambda^T A)\delta x+(u^T R+\lambda^T B_u)\delta u\\ &+\delta\lambda^T(Ax+B_u u-\dot{x})-\lambda^T\delta\dot{x}\,\mathrm{d}t=0.\end{aligned}$$

- The last term, $\lambda^T\delta\dot{x}$, deserves special attention. It is a function of δx so is not an independent variable. We can eliminate it from the equation via integration-by-parts

$$\int_0^{t_f}\lambda^T(t)\delta\dot{x}(t)\,\mathrm{d}t=\lambda^T(t)\delta x(t)\Big|_0^{t_f}-\int_0^{t_f}\dot{\lambda}^T(t)\delta x(t)\,\mathrm{d}t.$$

- Note that the initial state is fixed, so its partial is zero so

$$\int_0^{t_f}\lambda^T(t)\delta\dot{x}(t)\,\mathrm{d}t=\lambda^T(t_f)\delta x(t_f)-\int_0^{t_f}\dot{\lambda}^T(t)\delta x(t)\,\mathrm{d}t.$$

- Substituting,

$$\begin{aligned}\delta J_a(x,u,\lambda,\delta x,\delta u,\delta\lambda) = &\left[x^T(t_f)H-\lambda^T(t_f)\right]\delta x(t_f)\\ &+\int_0^{t_f}(x^T Q+\lambda^T A+\dot{\lambda}^T)\delta x+(u^T R+\lambda^T B_u)\delta u\\ &+\delta\lambda^T(Ax+B_u u-\dot{x})\,\mathrm{d}t=0.\end{aligned}$$

- Since the variations δx, δu, $\delta\lambda$ are arbitrary, this expression is zero iff

$$\lambda^T(t_f) = x^T(t_f)H, \qquad 0 = u^T(t)R + \lambda^T(t)B_u,$$
$$\dot{\lambda}^T(t) = -x^T(t)Q - \lambda^T(t)A, \qquad \dot{x}(t) = Ax(t) + B_u u(t).$$

- From the third result, the optimal control is $u(t) = -R^{-1}B_u^T\lambda(t)$, where the inverse exists since $R > 0$. Still need to solve for $\lambda(t)$ ...

- Combining two remaining equations (substitute $u(t)$ as above)

$$\begin{bmatrix} \dot{x}(t) \\ \hdashline \dot{\lambda}(t) \end{bmatrix} = \underbrace{\left[\begin{array}{c:c} A & -B_u R^{-1} B_u^T \\ \hdashline -Q & -A^T \end{array}\right]}_{\text{Hamiltonian matrix}} \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix} = \mathscr{Z}\begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix}$$

- We solve this system of equations noting that $x(0) = x_0$ and $\lambda(t_f) = Hx(t_f)$.

$$\begin{bmatrix} x(t_f) \\ \hdashline \lambda(t_f) \end{bmatrix} = e^{\mathscr{Z}(t_f - t)} \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix} = \left[\begin{array}{c:c} \Phi_{11}(t_f - t) & \Phi_{12}(t_f - t) \\ \hdashline \Phi_{21}(t_f - t) & \Phi_{22}(t_f - t) \end{array}\right] \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix}$$

- Substitute $\lambda(t_f)$

$$\begin{bmatrix} x(t_f) \\ \hdashline Hx(t_f) \end{bmatrix} = \left[\begin{array}{c:c} \Phi_{11}(t_f - t) & \Phi_{12}(t_f - t) \\ \hdashline \Phi_{21}(t_f - t) & \Phi_{22}(t_f - t) \end{array}\right] \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix}$$

- Eliminating $x(t_f)$ by substituting first equation into second

$$H\left[\Phi_{11}(t_f - t)x(t) + \Phi_{12}(t_f - t)\lambda(t)\right] = \Phi_{21}(t_f - t)x(t) + \Phi_{22}(t_f - t)\lambda(t).$$

- Solve for $\lambda(t)$

$$\begin{aligned}\lambda(t) &= \left[\Phi_{22}(t_f - t) - H\Phi_{12}(t_f - t)\right]^{-1}\left[H\Phi_{11}(t_f - t) - \Phi_{21}(t_f - t)\right]x(t) \\ &= P(t)x(t),\end{aligned}$$

so

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

Solving for $P(t)$ via the Hamiltonian System

- From above, $\lambda(t) = P(t)x(t)$. Differentiating (via product rule) we get
$$\dot{\lambda}(t) = \dot{P}(t)x(t) + P(t)\dot{x}(t).$$
- Substitute for $\dot{\lambda}(t)$ and $\dot{x}(t)$ to get
$$-Qx(t) - A^T\lambda(t) = \dot{P}(t)x(t) + P(t)[Ax(t) - B_u R^{-1}B_u^T\lambda(t)].$$
- Substitute $\lambda(t) = P(t)x(t)$ to get
$$[\dot{P}(t) + P(t)A + A^T P(t) + Q - P(t)B_u R^{-1}B_u^T P(t)]x(t) = 0.$$
- Since this is valid for arbitrary $x(t)$ we get
$$\dot{P}(t) = -P(t)A - A^T P(t) - Q + P(t)B_u R^{-1}B_u^T P(t).$$
- This is called the differential (matrix) Riccati equation.
- It is a nonlinear differential equation with boundary condition $P(t_f) = H$, solved *backwards* 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}.
$$0 = P_{\text{ss}}BR^{-1}B^T P_{\text{ss}} - P_{\text{ss}}A - A^T P_{\text{ss}} - Q.$$
- This is called the (continuous-time) algebraic Riccati equation (ARE). In MATLAB, `care.m`

3.4: Solving the Riccati equation

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.

- An easy way to do this is to use Simulink with matrix signals.

EXAMPLE: Consider the continuous-time system

$$\dot{x}(t) = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(t).$$

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

$$J = \frac{1}{2}x^T(5)\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} x(5) + \frac{1}{2}\int_0^5 [y^T(t)y(t) + u^T(t)u(t)]\,\mathrm{d}t.$$

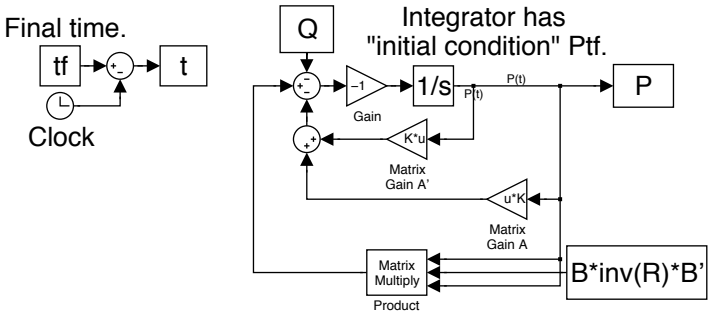
- First, note that the open-loop system is unstable, with poles at 0 and 1. It is controllable and observable.

- 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

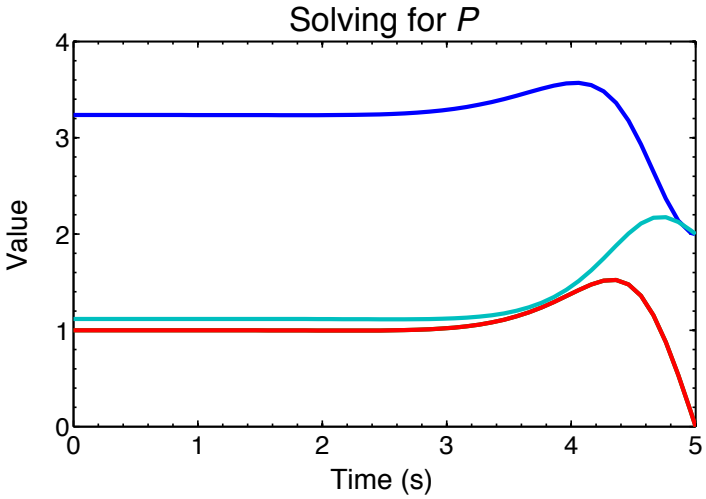
$$J = \frac{1}{2}x^T(5)\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} x(5) + \frac{1}{2}\int_0^5 \left[x^T(t)C^T Cx(t) + u^T(t)u(t)\right]\,\mathrm{d}t.$$

  This is a common “trick”.

- Therefore, the penalty matrices are $Q = C^T C$ and $R = \rho = 1$.
- We can simulate the finite-horizon case to find $P(t)$.



To plot:
```
P=P.signals.values;
t=squeeze(t.signals.values);
P2=reshape(P,[min(size(P))^2 1 length(P)]);
plot(t,squeeze(P2)')
```



Solving the algebraic Riccati equation manually

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

$$0 = A^T P + PA + C^T C - PBR^{-1}B^T P$$

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

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

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

$$2p_{11} - p_{11}^2 + 4p_{12} = 0 \qquad 1 - p_{12}^2 = 0.$$

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

- 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

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

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

Solving the algebraic Riccati equation generally

- Can sometimes solve ARE manually, as above.
- Or, can solve ARE by substituting $P(t_f) = H$ and solving differential equation backwards in time until steady-state achieved.
  - Usually a bad idea due to computation involved and propagation of numeric errors.
- Instead, recall

$$\begin{bmatrix} \dot{x}(t) \\ \hdashline \dot{\lambda}(t) \end{bmatrix} = \left[\begin{array}{c:c} A & -B_u R^{-1} B_u^T \\ \hdashline -Q & -A^T \end{array}\right] \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix} = \mathscr{Z} \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix}$$

- Diagonalize the Hamiltonian matrix. This can be done if all eigenvalues distinct. If not, perturb Q or R.

$$\begin{bmatrix} \dot{z}_1(t) \\ \hdashline \dot{z}_2(t) \end{bmatrix} = \left[\begin{array}{c:c} -\Lambda & 0 \\ \hdashline 0 & \Lambda \end{array}\right] \begin{bmatrix} z_1(t) \\ \hdashline z_2(t) \end{bmatrix}$$

- The matrix Λ is diagonal, and contains RHP poles of Hamiltonian matrix.
- We achieved the transformation via the transformation matrix Ψ

$$\begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix} = \left[\begin{array}{c:c} \Psi_{11} & \Psi_{12} \\ \hdashline \Psi_{21} & \Psi_{22} \end{array}\right] \begin{bmatrix} z_1(t) \\ \hdashline z_2(t) \end{bmatrix}$$

- The reverse relationship is then

$$\begin{bmatrix} z_1(t) \\ \hdashline z_2(t) \end{bmatrix} = \left[\begin{array}{c:c} (\Psi^{-1})_{11} & (\Psi^{-1})_{12} \\ \hdashline (\Psi^{-1})_{21} & (\Psi^{-1})_{22} \end{array}\right] \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix}$$

- Solving for $z_2(t)$ we get

$$\begin{aligned} z_2(t) = e^{\Lambda t} z_2(0) &= (\Psi^{-1})_{21} x(t) + (\Psi^{-1})_{22} \lambda(t) \\ &= \left[(\Psi^{-1})_{21} + (\Psi^{-1})_{22} P(t)\right] x(t). \end{aligned}$$

- As $t \to \infty$, $x(t) \to 0$ for the cost function to remain finite. Therefore

$$\lim_{t\to\infty} z_2(t) = \lim_{t\to\infty} e^{\Lambda t} z_2(0) = \lim_{t\to\infty} \left[(\Psi^{-1})_{21} + (\Psi^{-1})_{22} P(t)\right] x(t) = 0$$

which forces $z_2(0) = 0$ and therefore $z_2(t) = 0$. Then

$$x(t) = \Psi_{11} z_1(t);$$

$$\lambda(t) = \Psi_{21} z_1(t).$$

- Since $\lambda(t) = Px(t) = \Psi_{21}(\Psi_{11})^{-1}x(t)$ then

$$P = \Psi_{21}(\Psi_{11})^{-1}$$

and the steady-state optimal feedback gain matrix is

$$K = R^{-1}B_u^T P = R^{-1}B_u^T \Psi_{21}(\Psi_{11})^{-1}.$$

- Summary:

1. Find the eigenvalues and eigenvectors of the Hamiltonian matrix.
2. Select the eigenvectors associated with stable eigenvalues and write as

$$\left[\begin{array}{c} \Psi_{11} \\ \hdashline \Psi_{21} \end{array}\right].$$

3. Compute the steady-state $P = \Psi_{21}(\Psi_{11})^{-1}$.
4. Compute the steady-state control gain as $K = R^{-1}B_u^T \Psi_{21}(\Psi_{11})^{-1}$.

- "Schur decomposition" algorithm also exists—numerically more stable.

3.5: Symmetric root locus

- We have found that the closed-loop dynamics are governed by

$$\left[\begin{array}{c:c} \dot{x}(t) \\ \hdashline \dot{\lambda}(t) \end{array}\right] = \left[\begin{array}{c:c} A & -B_u R^{-1} B_u^T \\ \hdashline -Q & -A^T \end{array}\right] \left[\begin{array}{c:c} x(t) \\ \hdashline \lambda(t) \end{array}\right] = \mathscr{Z} \left[\begin{array}{c:c} x(t) \\ \hdashline \lambda(t) \end{array}\right]$$

- We have seen how to eliminate $\lambda(t)$ using the solution to a Riccati equation (more later), but the Hamiltonian is still useful since it can tell us where the closed-loop poles are.
  - Closed-loop poles are given by $\det(sI - \mathscr{Z}) = 0$.
  - Can evaluate this using the block-matrix determinant identity:

$$\det \left[\begin{array}{c:c} A & B \\ \hdashline C & D \end{array}\right] = \det(A) \det(D - CA^{-1}B).$$

- So, substituting terms from the Hamiltonian,

$$\begin{aligned} &\det(sI - \mathscr{Z}) \\ &= |sI - A| \cdot \det[(sI + A^T) - Q(sI - A)^{-1} B_u R^{-1} B_u^T] \\ &= |sI - A| \cdot |sI + A^T| \cdot \det[I - (sI + A^T)^{-1} Q (sI - A)^{-1} B_u R^{-1} B_u^T] = 0. \end{aligned}$$

- Note: $\det(I + ABC) = \det(I + CAB)$ so,

$$\begin{aligned} &\det(sI - \mathscr{Z}) \\ &= |sI - A| \cdot |sI + A^T| \cdot \det[I + R^{-1} B_u^T (-sI - A^T)^{-1} Q (sI - A)^{-1} B_u]. \end{aligned}$$

- Let $Q = C_y^T Q_y C_y$. Also

$$G_{yu}(s) = C_y (sI - A)^{-1} B_u$$

$$G_{yu}^T(-s) = B_u^T (-sI - A^T)^{-1} C_y^T$$

$$D(s) = \det(sI - A)$$

$$D(-s) = \det(sI + A)^T (-1)^n$$

- Therefore

$$\begin{aligned}\det(sI - \mathscr{Z}) &= (-1)^n D(s)D(-s)\det[I + R^{-1}G_{yu}^T(-s)Q_y G_{yu}(s)] \\ &= (-1)^n D(s)D(-s)[I + R^{-1}Q_y G_{yu}(s)G_{yu}(-s)] \quad \text{if SISO.}\end{aligned}$$

- This is called the "symmetric root locus" for reasons that will become clear.
  - In SISO case, there will be easy drawing rules.
  - In MIMO case, don't have easy drawing rules, but still have closed-loop pole symmetry such that if p is a pole of the system, then $-p$ is also a pole of the system.
- The stable poles are roots of the regulator. The "unstable" poles when solving the differential equation in the forward direction become stable poles when solving in the backward direction, so are poles of $\lambda(t)$.

Symmetric root locus in MATLAB

- We want to plot the root locus

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

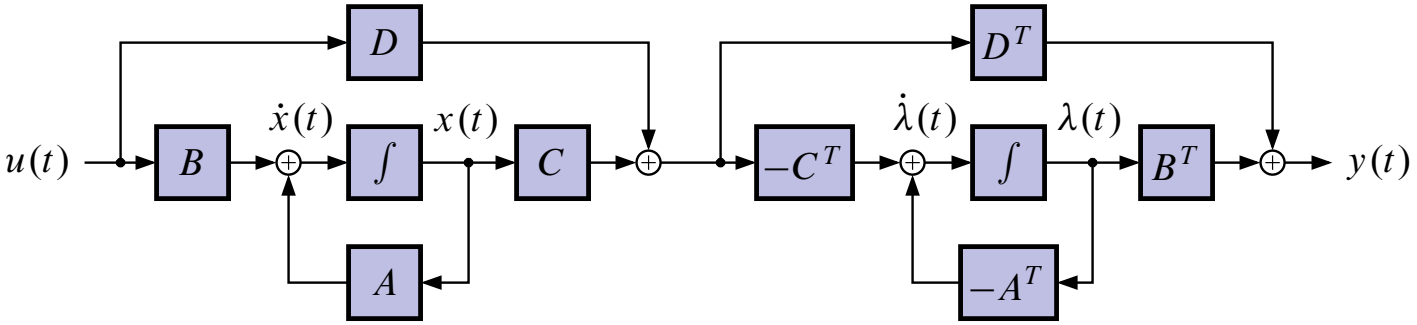
- We need to find a way to represent $G^T(-s)G(s)$ as a state-space system in MATLAB.

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

and

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

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



- The overall system has state

$$\begin{bmatrix} \dot{x}(t) \\ \hdashline \dot{\lambda}(t) \end{bmatrix} = \left[\begin{array}{c:c} A & 0 \\ \hdashline -C^TC & -A^T \end{array}\right] \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix} + \begin{bmatrix} B \\ \hdashline -C^TD \end{bmatrix} u(t)$$

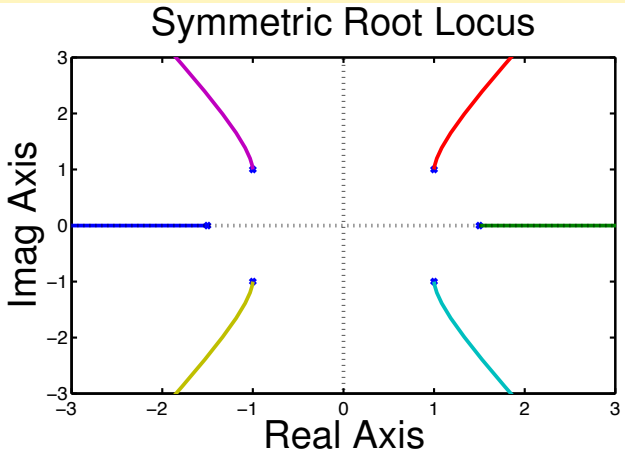
$$y(t) = \left[\begin{array}{c:c} D^TC & B^T \end{array}\right] \begin{bmatrix} x(t) \\ \hdashline \lambda(t) \end{bmatrix} + D^TDu(t).$$

```
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.



3.6: Stochastic LQR

- An intermediate step to developing controllers for systems with incomplete, noisy measurements.
- We consider full state available, noisy system.

$$\dot{x}(t) = Ax(t) + B_u u(t) + B_w w(t)$$

and $w(t) \sim \mathcal{N}(0, S_w)$ and white; $x(0)$ stochastic too: $\mathbb{E}[x(0)] = m_o$ and $\mathbb{E}[x(0)x(0)^T] = \Sigma_x(0)$.

Cost functions involving random signals

- Need to use expectation to come up with real measure of performance

$$\begin{aligned} J &= \mathbb{E}\left[\int_0^{t_f} z^T(t)W(t)z(t)\,\mathrm{d}t\right] \\ &= \int_0^{t_f} \mathbb{E}\left[z^T(t)W(t)z(t)\right]\,\mathrm{d}t. \end{aligned}$$

- For a finite final time, constant W and stationary random inputs, this is proportional to

$$J = \mathbb{E}\left[z^T(t)Wz(t)\right].$$

- We can *compute* the cost by noticing that J is a scalar and that $J = \mathrm{trace}\{J\}$.

$$J = \mathrm{trace}\left\{\mathbb{E}\left[z^T(t)Wz(t)\right]\right\}.$$

- The trace operator is invariant under cyclic permutation

$$J = \mathrm{trace}\left\{W\mathbb{E}\left[z(t)z^T(t)\right]\right\} = \mathrm{trace}\left\{W\Sigma_z\right\}.$$

- We can compute

$$\Sigma_z = \int_0^\infty \int_0^\infty g_{cl}(\tau_1) R_w(\tau_1 - \tau_2) g_{cl}^T(\tau_2)\, \mathrm{d}\tau_1\, \mathrm{d}\tau_2$$

where g_{cl} is the impulse response of the closed-loop system from input w to output z.

- *If* w is white with spectral density S_w, this simplifies

$$\Sigma_z = \int_0^\infty g_{cl}(\tau) S_w g_{cl}^T(\tau)\, \mathrm{d}\tau.$$

- We can also compute Σ_z from Σ_x if $z(t) = C_z x(t)$

$$\Sigma_z = C_z \mathbb{E}\left[x(t)x^T(t)\right] C_z^T = C_z \Sigma_x C_z^T.$$

Then $J = \mathrm{trace}\{W C_z \Sigma_x C_z^T\}$, where

$$A_{cl}\Sigma_x + \Sigma_x A_{cl}^T + B_{cl} S_w B_{cl}^T = 0$$

may be solved for Σ_x using `lyap`.

Finite horizon, white process noise

- Moving from cost functions in general to specific LQR case: cost is now of the form

$$J_s = \mathbb{E}\left[x^T(t_f) H x(t_f) + \int_0^{t_f} x^T(t) Q x(t) + u^T(t) R u(t)\, \mathrm{d}t\right].$$

- The controller *must* be some kind of feedback system (state or output) since $x(0)$ is not known *a priori*. Linearity in the feedback is assumed (and is optimal if the disturbance inputs are Gaussian).
- The system state contains all the information about past inputs and past states that contribute to the future behavior of the system.

- Also, since $w(t)$ is white, noise inputs are uncorrelated with past inputs and cannot be predicted from past inputs or states.
- Therefore, the current state contains all the available information on the future behavior of the plant.
- Therefore, control at any time instant same as deterministic LQR with unknown initial state.

Finite horizon, colored process noise

- If we know that the noise is colored, we should augment the shaping filter dynamics.
- Design stochastic regulator for augmented system. Same design procedure as deterministic since input is white Gaussian.
- Solution includes feedback on disturbance dynamics "state." Non physical—we cannot measure $x_h(t)$.
- Therefore will need to estimate $x_h(t)$. . . more on this later with LQG.

Infinite horizon, white process noise

- Extend analysis of stochastic LQR to case of infinite time horizon $t_f \to \infty$.
- Already did this for deterministic LQR. Had no particular problems because
  - System asymptotically stable,
  - Only disturbance is initial condition $x(0)$ which goes to zero as $t_f \to \infty$. Therefore J_{LQR} finite.

- Cannot make similar claims for the stochastic case. Have continuing driving noise so even optimized cost $\to \infty$.
- Solution: Use a time-averaged cost

$$J = \lim_{t_f \to \infty} \mathbb{E}\left[\frac{1}{t_f}\int_0^{t_f} x^T(t)Qx(t) + u^T(t)Ru(t)\,\mathrm{d}t.\right]$$

- Then, same optimality conditions as for finite horizon case since both are fixed end-time problems.
- Consider: A time-invariant system.
- If control properly posed, optimum closed-loop system stable. $x(t)$ and $u(t)$ are stationary random processes, and we can write

$$J = \lim_{t \to \infty} \mathbb{E}\left[x(t)^T Qx(t) + u(t)^T Ru(t)\right].$$

- Provides steady-state mean-square response.
- Can use this form to analyze controller performance. Consider control $u = -Kx$. Then

$$\dot{x}(t) = (A - B_u K)x(t) + B_w w(t)$$

$$y(t) = C_y x(t).$$

- Then,

$$\begin{aligned} J &= \lim_{t \to \infty} \mathbb{E}\left[x^T(t)Qx(t) + x^T(t)K^T RKx(t)\right] \\ &= \lim_{t \to \infty} \mathbb{E}\left[x^T(t)\left\{Q + K^T RK\right\}x(t)\right] \\ &= \lim_{t \to \infty} \operatorname{trace}\left\{(Q + K^T RK)\mathbb{E}\left[x(t)x(t)^T\right]\right\} \\ &= \operatorname{trace}\left\{(Q + K^T RK)\Sigma_{x,\mathrm{ss}}\right\}, \end{aligned}$$

where $\Sigma_{x,\mathrm{ss}}$ solves

$$(A - B_u K)\Sigma_{x,\mathrm{ss}} + \Sigma_{x,\mathrm{ss}}(A - BK)^T + B_w S_w B_w^T = 0.$$

- We will later use this cost when comparing the performance of LQR versus LQG.

$\mathscr{H}_2$ optimal control

- A final observation is that we can pose the stochastic LQR problem as an $\mathscr{H}_2$ optimization.
- The $\mathscr{H}$ refers to the Hardy space of all stable, LTI systems.
- The subscript "2" denotes the applicable system norm.
- The $\mathscr{H}_2$ problem is to find the LTI controller for the plant

$$\dot{x}(t) = Ax(t) + \left[\begin{array}{c:c} B_u & I \end{array}\right] \left[\begin{array}{c} u(t) \\ \hdashline w(t) \end{array}\right];$$

$$\left[\begin{array}{c} m(t) \\ \hdashline y_1(t) \\ u_1(t) \end{array}\right] = \left[\begin{array}{c} I \\ \hdashline Q^{1/2} \\ 0 \end{array}\right] x(t) + \left[\begin{array}{c:c} 0 & 0 \\ \hdashline 0 & 0 \\ R^{1/2} & 0 \end{array}\right] \left[\begin{array}{c} u(t) \\ \hdashline w(t) \end{array}\right],$$

that stabilizes the closed-loop system and minimizes the system 2-norm

$$J_2 = \left[\int_0^\infty \text{trace}\{g_{cl}^T(t) g_{cl}(t)\, \mathrm{d}t\right]^{1/2} = \|G_{cl}\|_2,$$

where $g_{cl}(t)$ is the impulse response from $w(t)$ to the reference output (combination of $u_1(t)$ and $y_1(t)$).

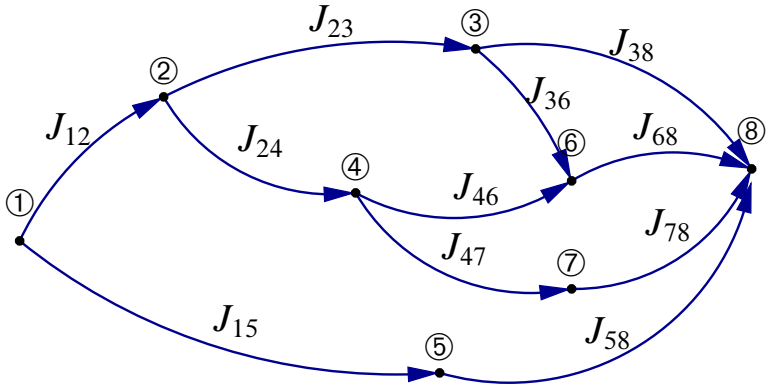
- Equivalent to steady-state stochastic regulator with $S_w = I$. Then, steady-state mean square value of reference output is

$$\mathbb{E}\left\{\left[\begin{array}{c:c} y_1^T(\infty) & u_1^T(\infty) \end{array}\right] \left[\begin{array}{c} y_1(\infty) \\ \hdashline u_1(\infty) \end{array}\right]\right\} = \mathbb{E}[x^T(\infty) Q x(\infty) + u^T(\infty) R u(\infty)],$$

which is also the square of the closed-loop 2-norm $\|G_{cl}\|_2^2$.

3.7: The LQR problem solved via dynamic programming

- We now consider the discrete-time LQR problem, and a different method of solution known as dynamic programming.
  - Review from final chapter of ECE5520 materials.
- Dynamic programming optimizes multi-step optimizations a single step at a time, efficiently eliminating infeasible paths.
- 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_{t_f} *and* x_i *is on the optimal path,* then $J_{of}^* = J_{oi} + J_{if}^*$.

- This is called Bellman's principle of optimality:

  "An optimal path has the property that whatever the initial conditions and control variables (choices) over some initial period, the control (or decision variables) chosen over the remaining period must be optimal for the remaining problem, with the state resulting from the early decisions taken to be the initial condition."

Vector derivatives

- In the derivation, we will need to take derivatives of vector/ matrix quantities.
- This small dictionary should help: $x, y \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$.
  1. $\frac{\partial}{\partial x}(x^T A y) = Ay$,
  2. $\frac{\partial}{\partial x}(y^T A x) = A^T y$,
  3. $\frac{\partial}{\partial x}(x^T A x) = (A + A^T)x$.

The discrete-time LQR problem

- We want to choose u_k such that we minimize

$$J_{i,N} = x_N^T H_d x_N + \sum_{k=i}^{N-1} \left[x_k^T Q_d x_k + u_k^T R_d u_k \right],$$

- To find the optimum u_k, we start at the last step and work backwards.

$$J_{N-1,N} = x_N^T H_d x_N + x_{N-1}^T Q_d x_{N-1} + u_{N-1}^T R_d u_{N-1}.$$

- We express x_N as a function of x_{N-1} and u_{N-1} via the system dynamics

$$\begin{aligned} J_{N-1,N} &= (Ax_{N-1} + Bu_{N-1})^T H_d (Ax_{N-1} + Bu_{N-1}) \\ &\quad + x_{N-1}^T Q_d x_{N-1} + u_{N-1}^T R_d u_{N-1} \\ &= x_{N-1}^T A^T H_d A x_{N-1} + u_{N-1}^T B^T H_d B u_{N-1} \\ &\quad + x_{N-1}^T A^T H_d B u_{N-1} + u_{N-1}^T B^T H_d A x_{N-1} \\ &\quad + x_{N-1}^T Q_d x_{N-1} + u_{N-1}^T R_d u_{N-1}. \end{aligned}$$

- We minimize over all possible inputs u_{N-1} by differentiation

$$\frac{\partial J_{N-1,N}}{\partial u_{N-1}} = 2B^T H_d B u_{N-1} + 2B^T H_d A x_{N-1} + 2R_d u_{N-1} = 0$$

$$0 = 2\left(R_d + B^T H_d B\right) u_{N-1} + 2B^T H_d A x_{N-1}.$$

- Therefore,

$$u^*_{N-1} = -\left(R_d + B^T H_d B\right)^{-1} B^T H_d A x_{N-1}.$$

- 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

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

such that

$$u^*_{N-1} = -K_{N-1} x_{N-1}.$$

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

$$J^*_{N-1,N} = \left[\left(Ax_{N-1} - BK_{N-1}x_{N-1}\right)^T H_d \left(Ax_{N-1} - BK_{N-1}x_{N-1}\right) + x^T_{N-1} Q_d x_{N-1} + x^T_{N-1} K^T_{N-1} R_d K_{N-1} x_{N-1}\right]$$

$$= x^T_{N-1}\left[(A - BK_{N-1})^T H_d (A - BK_{N-1}) + Q_d + K^T_{N-1} R_d K_{N-1}\right] x_{N-1}.$$

- Simplify notation once again by defining $P_N = H_d$ and

$$P_{N-1} = (A - BK_{N-1})^T P_N (A - BK_{N-1}) + Q_d + K^T_{N-1} R_d K_{N-1},$$

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

$$J_{N,N} = J^*_{N,N} = x_N^T P_N x_N \triangleq x_N^T H_d x_N.$$

- 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

$$J_{N-2,N-1} = (Ax_{N-2} + Bu_{N-2})^T P_{N-1} (Ax_{N-2} + Bu_{N-2})$$
$$+x_{N-2}^T Q_d x_{N-2} + u_{N-2}^T R_d u_{N-2}.$$

- We can find the best result just as before

$$u^*_{N-2} = -K_{N-2} x_{N-2}$$

  where

$$K_{N-2} = \left(R_d + B^T P_{N-1} B\right)^{-1} B^T P_{N-1} A.$$

- In general,

$$u^*[k] = -K_k x[k]$$

  where

$$K_k = \left(R_d + B^T P_{k+1} B\right)^{-1} B^T P_{k+1} A$$

  and

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

- 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

$$J = x_{10}^T \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} x_{10} + \sum_{k=1}^{9} \left(x_k^T \begin{bmatrix} 2 & 0 \\ 0 & 0.1 \end{bmatrix} x_k + 2u_k^2 \right)$$

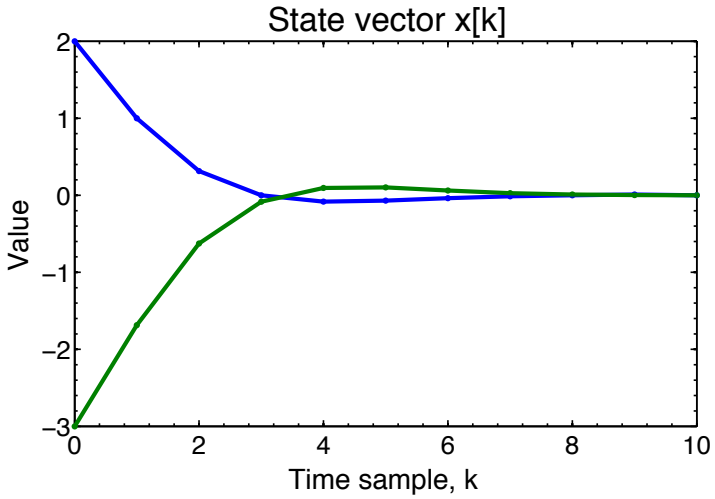
is minimized.

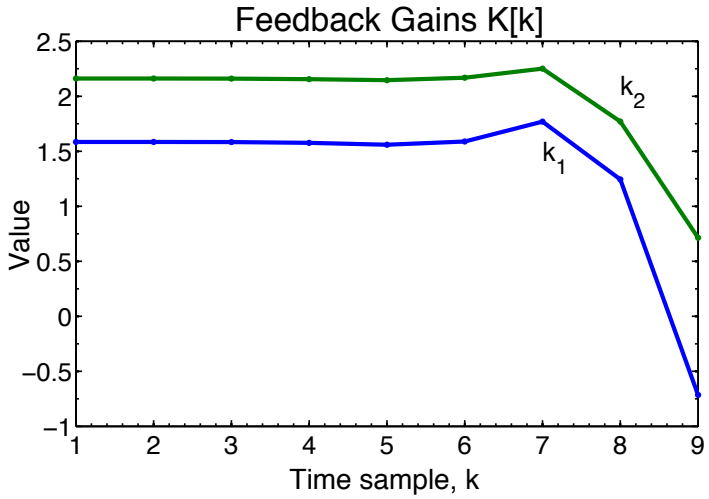
- From the problem, we gather that

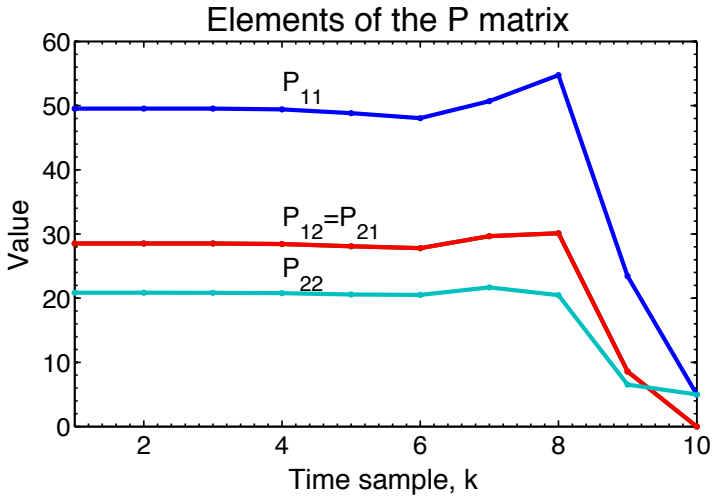
$$P_{10} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, \qquad Q_d = \begin{bmatrix} 2 & 0 \\ 0 & 0.1 \end{bmatrix}, \qquad R_d = [2].$$

- 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
```







3.8: 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 $P_k = P_{k+1} = P_{\text{ss}}$ in the above equation.

$$P_{\text{ss}} = (A - BK)^T P_{\text{ss}}(A - BK) + Q_d + K^T R_d K$$

and

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

which may be combined to get

$$P_{\text{ss}} = A^T P_{\text{ss}} A - A^T P_{\text{ss}} B \left(R_d + B^T P_{\text{ss}} B\right)^{-1} B^T P_{\text{ss}} A + Q_d$$

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

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

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

- There are many ways to solve the the DARE, but when Q_d 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 $Q_d = C^T C$ 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 earlier 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

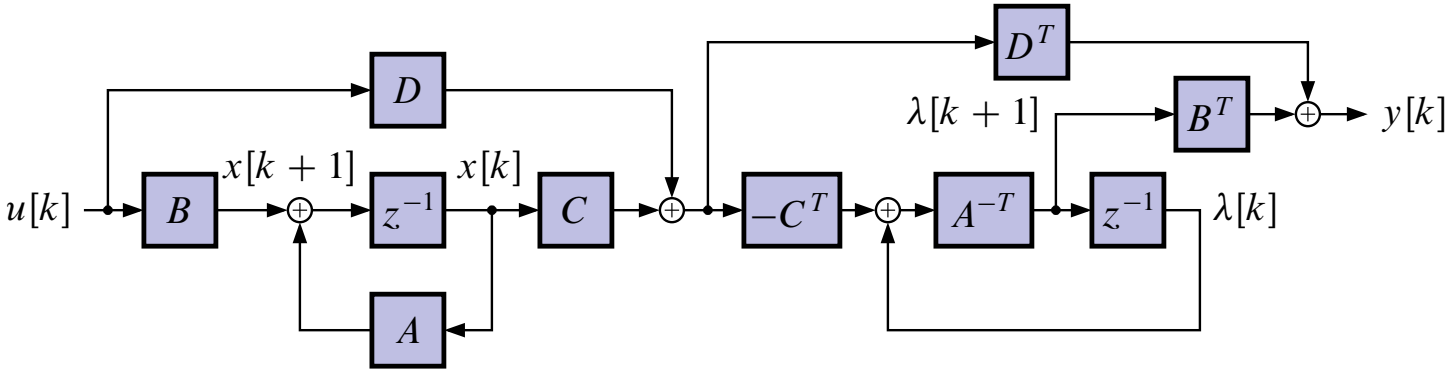
$$1 + KG'(z) = 0$$

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

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

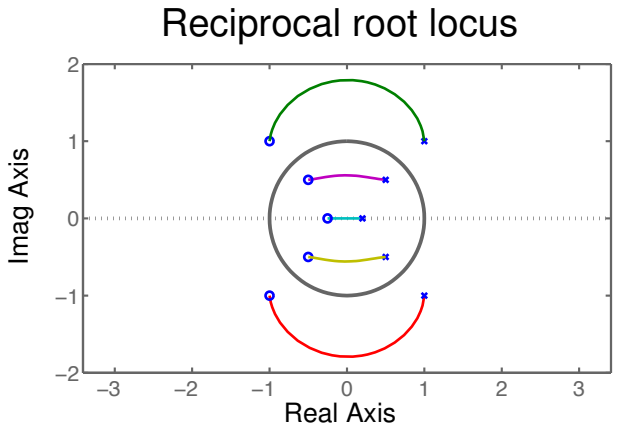
$$y[k] = \left[\begin{array}{c:c} -B^TA^{-T}C^TC + D^TC & B^TA^{-T} \end{array}\right]\left[\begin{array}{c} x[k] \\ \hdashline \lambda[k] \end{array}\right] + \left[D^TD - B^TA^{-T}C^TD\right]u[k].$$

```
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).