سایت اختصاصی
مهندسی کنترل

16.323 Lecture 1

Nonlinear Optimization

- Unconstrained nonlinear optimization
- Line search methods

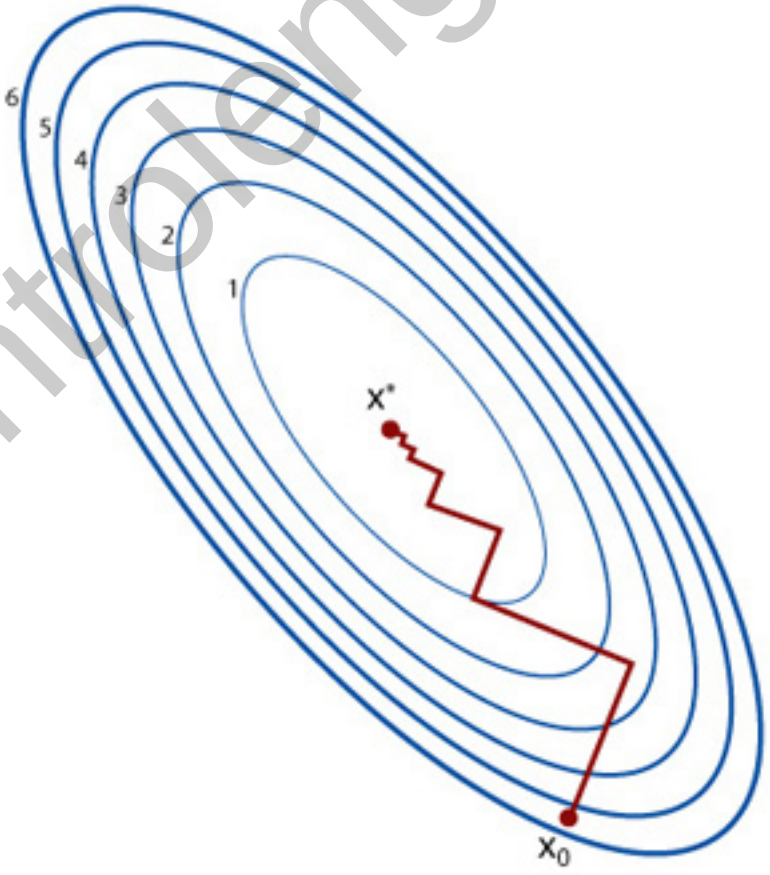


Figure by MIT OpenCourseWare.

Basics – Unconstrained

- Typical objective is to minimize a nonlinear function $F(\mathbf{x})$ of the parameters $\mathbf{x}$.
  - Assume that $F(\mathbf{x})$ is scalar $\Rightarrow \mathbf{x}^\star = \arg\min_{\mathbf{x}} F(\mathbf{x})$
- Define two types of minima:
  - **Strong**: objective function increases locally in all directions

  A point $\mathbf{x}^\star$ is a strong minimum of a function $F(\mathbf{x})$ if a scalar $\delta > 0$ exists such that $F(\mathbf{x}^\star) < F(\mathbf{x}^\star + \Delta\mathbf{x})$ for all $\Delta\mathbf{x}$ such that $0 < \|\Delta\mathbf{x}\| \leq \delta$
  - **Weak**: objective function remains same in some directions, and increases locally in other directions

  Point $\mathbf{x}^\star$ is a weak minimum of a function $F(\mathbf{x})$ if is not a strong minimum and a scalar $\delta > 0$ exists such that $F(\mathbf{x}^\star) \leq F(\mathbf{x}^\star + \Delta\mathbf{x})$ for all $\Delta\mathbf{x}$ such that $0 < \|\Delta\mathbf{x}\| \leq \delta$

- Note that a minimum is a **unique global minimum** if the definitions hold for $\delta = \infty$. Otherwise these are **local** minima.

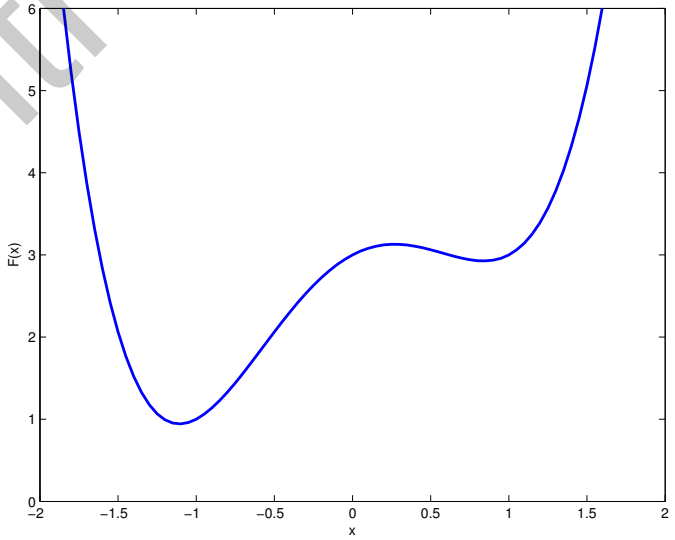


Figure 1.1: $F(x) = x^4 - 2x^2 + x + 3$ with local and global minima

First Order Conditions

- If $F(\mathbf{x})$ has continuous second derivatives, can approximate function in the neighborhood of an arbitrary point using Taylor series:

$$F(\mathbf{x}+\Delta\mathbf{x}) \approx F(\mathbf{x}) + \Delta\mathbf{x}^T\mathbf{g}(\mathbf{x}) + \frac{1}{2}\Delta\mathbf{x}^T G(\mathbf{x})\Delta\mathbf{x} + \dots$$

where $\mathbf{g} \sim$ gradient of F and $G \sim$ second derivative of F

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{g} = \left(\frac{\partial F}{\partial \mathbf{x}}\right)^T = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{bmatrix}, G = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}$$

- **First-order condition** from first two terms (assume $\|\Delta\mathbf{x}\| \ll 1$)
  - Given **ambiguity of sign** of the term $\Delta\mathbf{x}^T\mathbf{g}(\mathbf{x})$, can only avoid cost decrease $F(\mathbf{x}+\Delta\mathbf{x}) < F(\mathbf{x})$ if $\mathbf{g}(\mathbf{x}^\star) = 0$
    $\Rightarrow$ Obtain further information from higher derivatives
  - $\mathbf{g}(\mathbf{x}^\star) = 0$ is a necessary and sufficient condition for a point to be a **stationary point** – a necessary, but not sufficient condition to be a minima.
  - Stationary point could also be a maximum or a saddle point.

- Additional conditions can be derived from the Taylor expansion if we set $\mathbf{g}(\mathbf{x}^\star) = 0$, in which case:

$$F(\mathbf{x}^\star + \Delta\mathbf{x}) \approx F(\mathbf{x}^\star) + \frac{1}{2}\Delta\mathbf{x}^T G(\mathbf{x}^\star)\Delta\mathbf{x} + \dots$$

  - For a strong minimum, need $\Delta\mathbf{x}^T G(\mathbf{x}^\star)\Delta\mathbf{x} > 0$ for all $\Delta\mathbf{x}$, which is sufficient to ensure that $F(\mathbf{x}^\star + \Delta\mathbf{x}) > F(\mathbf{x}^\star)$.
  - To be true for arbitrary $\Delta\mathbf{x} \neq 0$, **sufficient** condition is that $G(\mathbf{x}^\star) > 0$ (PD). [1]

- Second order **necessary** condition for a strong minimum is that $G(\mathbf{x}^\star) \geq 0$ (PSD), because in this case the higher order terms in the expansion can play an important role, i.e.

$$\Delta\mathbf{x}^T G(\mathbf{x}^\star)\Delta\mathbf{x} = 0$$

  but the third term in the Taylor series expansion is positive.

- **Summary**: require $\mathbf{g}(\mathbf{x}^\star) = 0$ and $G(\mathbf{x}^\star) > 0$ (sufficient) or $G(\mathbf{x}^\star) \geq 0$ (necessary)

[1] Positive Definite Matrix

Solution Methods

- Typically solve minimization problem using an iterative algorithm.
  - Given: An initial estimate of the optimizing value of $\mathbf{x} \Rightarrow \hat{\mathbf{x}}_k$ and a search direction $\mathbf{p}_k$
  - Find: $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k$, for some scalar $\alpha_k \neq 0$

- Sounds good, but there are some questions:
  - How find $\mathbf{p}_k$?
  - How find α_k ? $\Rightarrow$ "**line search**"
  - How find initial condition $\mathbf{x}_0$, and how sensitive is the answer to the choice?

- **Search direction:**
  - Taylor series expansion of $F(\mathbf{x})$ about current estimate $\hat{\mathbf{x}}_k$

$$\begin{aligned} F_{k+1} \equiv F(\hat{\mathbf{x}}_k + \alpha \mathbf{p}_k) &\approx F(\hat{\mathbf{x}}_k) + \frac{\partial F}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k) \\ &= F_k + \mathbf{g}_k^T(\alpha_k \mathbf{p}_k) \end{aligned}$$

    - ◇ Assume that $\alpha_k > 0$, and to ensure function decreases (i.e. $F_{k+1} < F_k$), set

$$\mathbf{g}_k^T \mathbf{p}_k < 0$$

    - ◇ $\mathbf{p}_k$'s that satisfy this property provide a **descent direction**
  - **Steepest descent** given by $\mathbf{p}_k = -\mathbf{g}_k$

- **Summary:** gradient search methods (first-order methods) using estimate updates of the form:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k - \alpha_k \mathbf{g}_k$$

Line Search

- Line Search - given a search direction, must decide how far to "step"
  - Expression $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ gives a new solution for all possible values of α - what is the right value to pick?
  - Note that $\mathbf{p}_k$ defines a slice through solution space – is a very specific combination of how the elements of $\mathbf{x}$ will change together.
- Would like to pick α_k to minimize $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$
  - Can do this line search in gory detail, but that would be very time consuming
    - ◇ Often want this process to be fast, accurate, and easy
    - ◇ Especially if you are not that confident in the choice of $\mathbf{p}_k$

- Consider simple problem: $F(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2$ with

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{p}_0 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \Rightarrow \mathbf{x}_1 = \mathbf{x}_0 + \alpha \mathbf{p}_0 = \begin{bmatrix} 1 \\ 1 + 2\alpha \end{bmatrix}$$

which gives that $F = 1 + (1 + 2\alpha) + (1 + 2\alpha)^2$ so that

$$\frac{\partial F}{\partial \alpha} = 2 + 2(1 + 2\alpha)(2) = 0$$

with solution $\alpha^\star = -3/4$ and $\mathbf{x}_1 = [1 \ \ -1/2]^T$

  - This is hard to generalize this to N-space – need a better approach

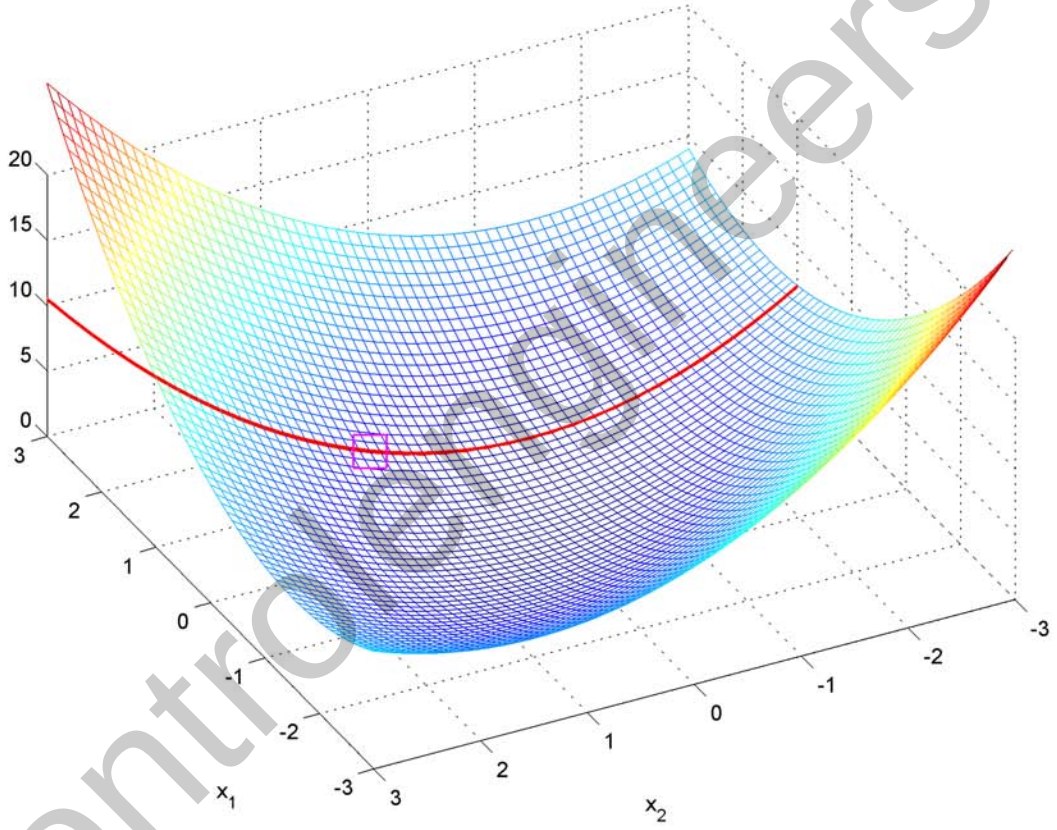


Figure 1.2: $F(x) = x_1^2 + x_1x_2 + x_2^2$ doing a line search in arbitrary direction

Line Search – II

- First step: search along the line until you think you have bracketed a "local minimum"

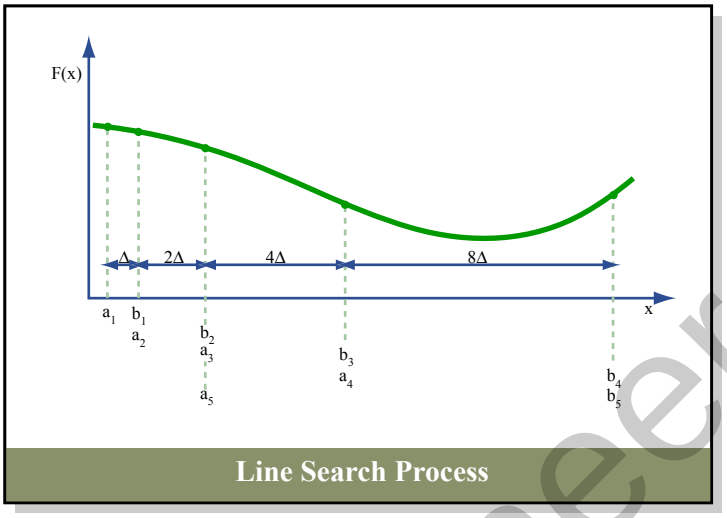


Figure by MIT OpenCourseWare.

Figure 1.3: Line search process

- Once you think you have a bracket of the local min – what is the smallest number of function evaluations that can be made to reduce the size of the bracket?
  - Many ways to do this:
    - ◇ Golden Section Search
    - ◇ Bisection
    - ◇ Polynomial approximations
  - First 2 have linear convergence, last one has "superlinear"

- Polynomial approximation approach
  - Approximate function as quadratic/cubic in the interval and use the minimum of that polynomial as the estimate of the local min.
  - Use with care since it can go very wrong – but it is a good termination approach.

- Cubic fits are a favorite:

$$\begin{aligned}\hat{F}(x) &= px^3 + qx^2 + rx + s \\ \hat{g}(x) &= 3px^2 + 2qx + r \ (\ = 0 \text{ at min})\end{aligned}$$

  Then $x^\star$ is the point (pick one) $x^\star = (-q \pm (q^2 - 3pr)^{1/2})/(3p)$ for which $\hat{G}(x^\star) = 6px^\star + 2q > 0$

- Great, but how do we find $x^\star$ in terms of what we know ($F(x)$ and $g(x)$ at the end of the bracket $[a, b]$)?

$$x^\star = a + (b - a)\left[1 - \frac{g_b + v - w}{g_b - g_a + 2v}\right]$$

  where

$$v = \sqrt{w^2 - g_a g_b} \quad \text{and} \quad w = \frac{3}{b-a}(F_a - F_b) + g_a + g_b$$

Figure 1.4: Cubic line search [Scales, pg. 40]

- **Observations**:
  - Tends to work well "near" a function local minimum (good convergence behavior)
  - But can be very poor "far away" $\Rightarrow$ use a hybrid approach of bisection followed by cubic.
- **Rule of thumb**: do not bother making the linear search too accurate, especially at the beginning
  - A waste of time and effort
  - Check the min tolerance – and reduce it as it you think you are approaching the overall solution.

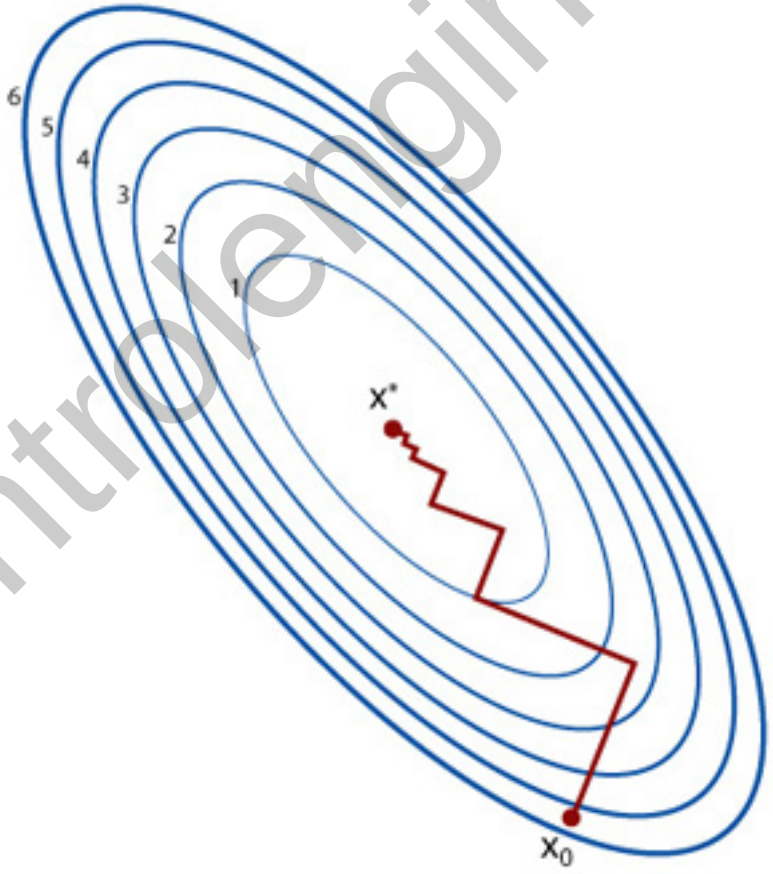


Figure by MIT OpenCourseWare.

Figure 1.5: zig-zag typical of steepest decent line searches

Second Order Methods

- Second order methods typically provide faster termination
  - Assume F is quadratic, and expand gradient $\mathbf{g}_{k+1}$ at $\hat{\mathbf{x}}_{k+1}$

$$\begin{aligned}\mathbf{g}_{k+1} \equiv \mathbf{g}(\hat{\mathbf{x}}_k + \mathbf{p}_k) &= \mathbf{g}_k + G_k(\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k) \\ &= \mathbf{g}_k + G_k\mathbf{p}_k\end{aligned}$$

    where there are no other terms because of the assumption that F is quadratic and

$$\mathbf{x}_k = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{g}_k = \left(\frac{\partial F}{\partial \mathbf{x}}\right)^T = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{bmatrix}_{\hat{\mathbf{x}}_k}$$

$$G_k = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}_{\hat{\mathbf{x}}_k}$$

  - So for $\hat{\mathbf{x}}_{k+1}$ to be at the minimum, need $\mathbf{g}_{k+1} = 0$, so that

$$\mathbf{p}_k = -G_k^{-1}\mathbf{g}_k$$

- Problem is that $F(\mathbf{x})$ typically not quadratic, so the solution $\hat{\mathbf{x}}_{k+1}$ is not at the minimum $\Rightarrow$ need to iterate

- Note that for a complicated $F(\mathbf{x})$, we may not have explicit gradients (should always compute them if you can)
  - But can always approximate them using finite difference techniques – but pretty expensive to find G that way
  - Use Quasi-Newton approximation methods instead, such as **BFGS** (Broyden-Fletcher-Goldfarb-Shanno)

FMINUNC Example

- Function minimization without constraints
  - Does quasi-Newton and gradient search
  - No gradients need to be formed
  - Mixture of cubic and quadratic line searches
- Performance shown on a complex function by Rosenbrock

$$F(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

  - Start at $x = [-1.9\ 2]$. Known global min it is at $x = [1\ 1]$

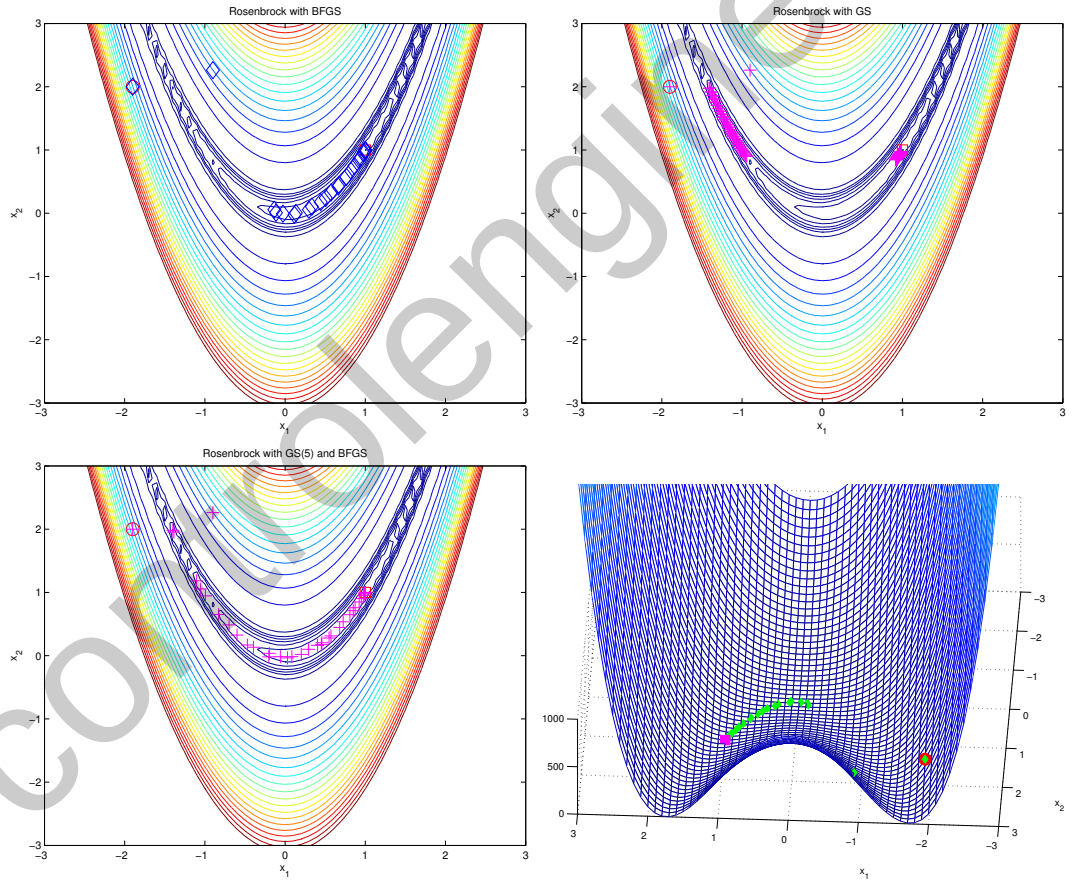
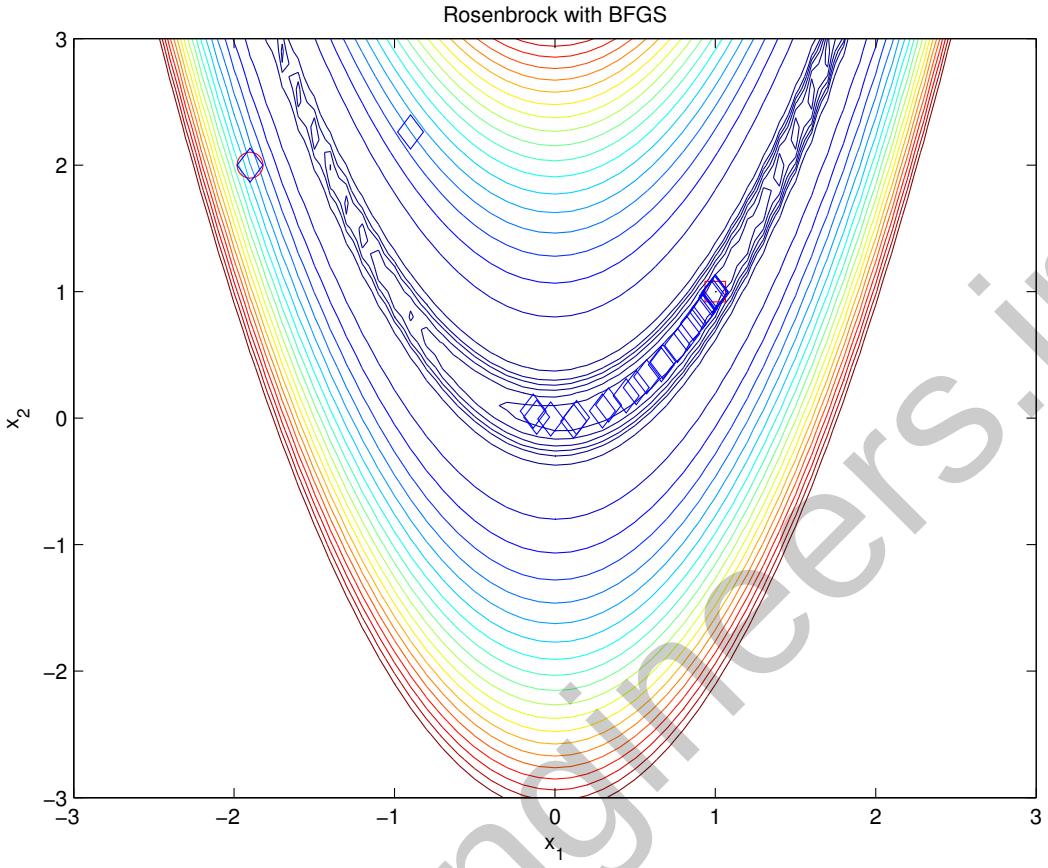
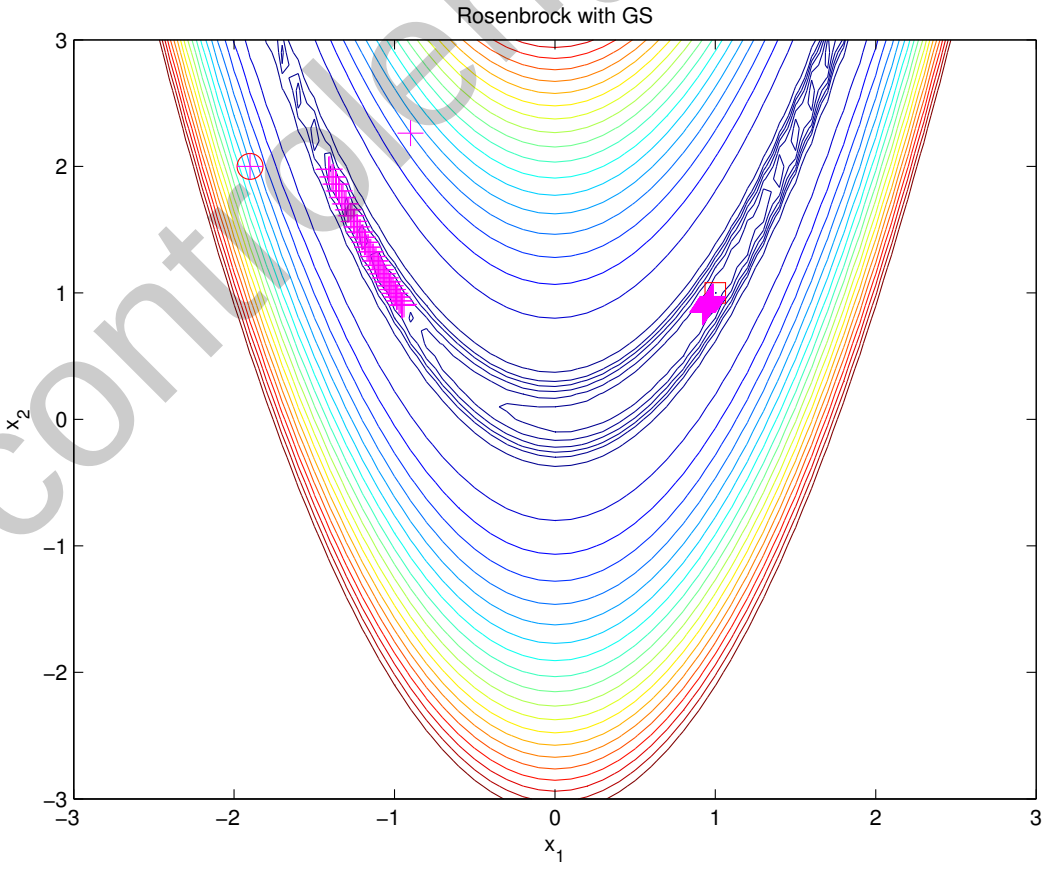
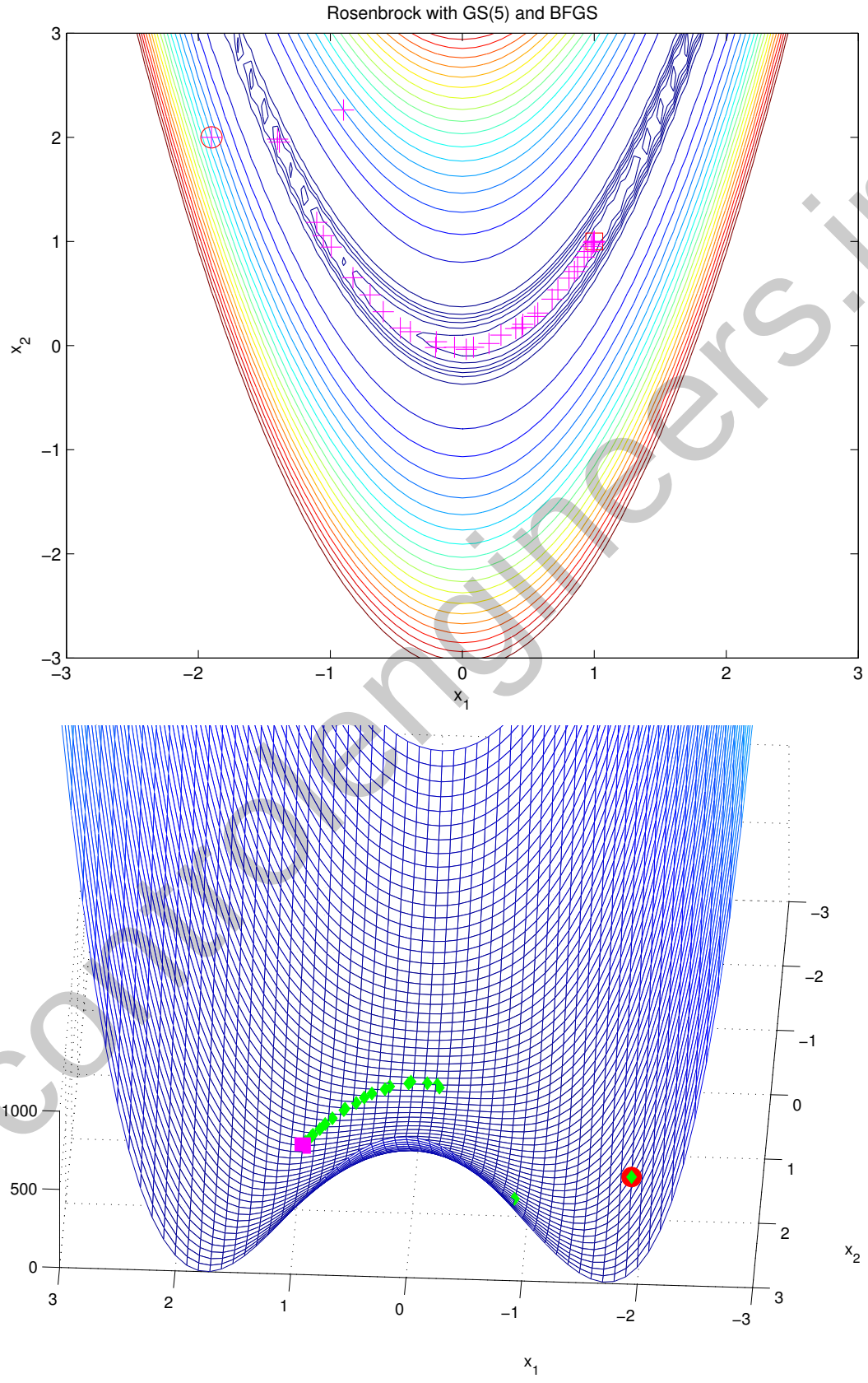


Figure 1.6: How well do the algorithms work?

- Quasi-Newton (BFGS) does well - gets to optimal solution in 26 iterations (35 ftn calls), but gradient search (steepest descent) fails (very close though), even after 2000 function calls (550 iterations).







- **Observations:**

1. Typically not a good idea to start the optimization with QN, and I often find that it is better to do GS for 100 iterations, and then switch over to QN for the termination phase.

2. $\hat{\mathbf{x}}_0$ tends to be very important – standard process is to try many different cases to see if you can find consistency in the answers.

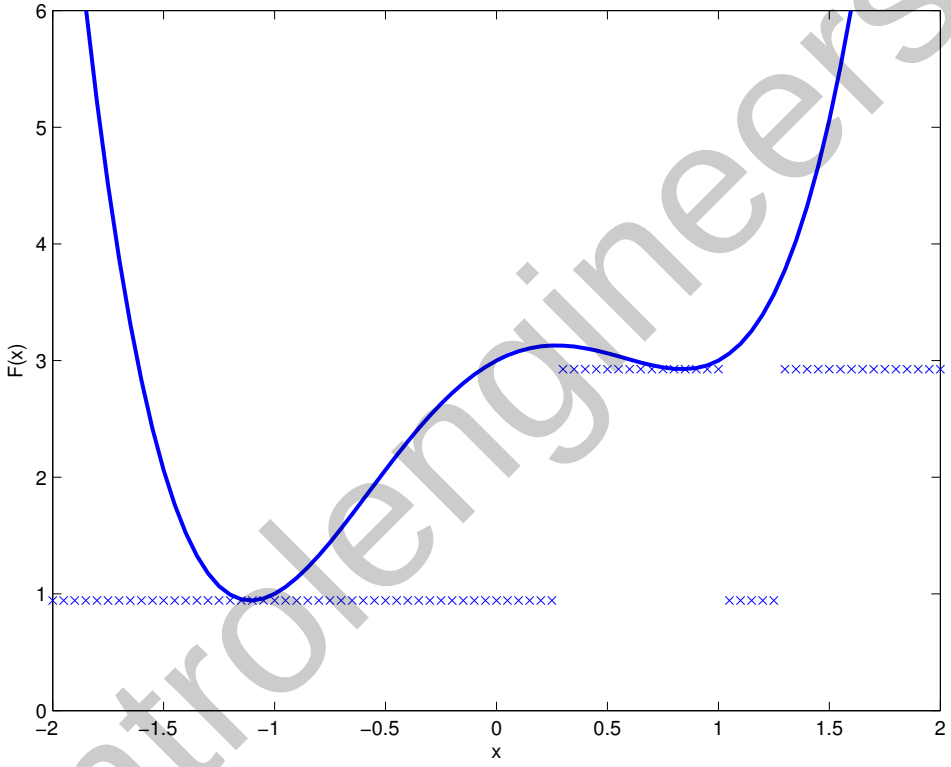


Figure 1.7: Shows how the point of convergence changes as a function of the initial condition.

3. Typically the convergence is to a local minimum and can be slow

4. Are there any guarantees on getting a good final answer in a reasonable amount of time? Typically yes, but not always.

Unconstrained Optimization Code

```
function [F,G]=rosen(x)
%global xpath

%F=100*(x(1)^2-x(2))^2+(1-x(1))^2;

if size(x,1)==2, x=x'; end

F=100*(x(:,2)-x(:,1).^2).^2+(1-x(:,1)).^2;
G=[100*(4*x(1)^3-4*x(1)*x(2))+2*x(1)-2; 100*(2*x(2)-2*x(1)^2)];

return

%
% Main calling part below - uses function above
%

global xpath

clear FF
x1=[-3:.1:3]'; x2=x1; N=length(x1);
for ii=1:N,
    for jj=1:N,
        FF(ii,jj)=rosen([x1(ii) x2(jj)]');
    end,
end

% quasi-newton
%
xpath=[];t0=clock;
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','bfgs','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity',...
    'MaxFunEvals',150,'OutputFcn', @outftn);

x0=[-1.9 2]';

xout1=fminunc('rosen',x0,opt) % quasi-newton
xbfgs=xpath;

% gradient search
%
xpath=[];
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','steepdesc','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity','MaxFunEvals',2000,'MaxIter',1000,'OutputFcn', @outftn);
xout=fminunc('rosen',x0,opt)
xgs=xpath;


% hybrid GS and BFGS
%
xpath=[];
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','steepdesc','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity','MaxFunEvals',5,'OutputFcn', @outftn);
xout=fminunc('rosen',x0,opt)
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','bfgs','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity','MaxFunEvals',150,'OutputFcn', @outftn);
xout=fminunc('rosen',xout,opt)

xhyb=xpath;

figure(1);clf
contour(x1,x2,FF',[0:2:10 15:50:1000])
hold on
plot(x0(1),x0(2),'ro','Markersize',12)
```

```
68  plot(1,1,'rs','Markersize',12)
69  plot(xbfgs(:,1),xbfgs(:,2),'bd','Markersize',12)
70  title('Rosenbrock with BFGS')
71  hold off
72  xlabel('x_1')
73  ylabel('x_2')
74  print -depsc rosen1a.eps;jpdf('rosen1a')
75
76  figure(2);clf
77  contour(x1,x2,FF',[0:2:10 15:50:1000])
78  hold on
79  xlabel('x_1')
80  ylabel('x_2')
81  plot(x0(1),x0(2),'ro','Markersize',12)
82  plot(1,1,'rs','Markersize',12)
83  plot(xgs(:,1),xgs(:,2),'m+','Markersize',12)
84  title('Rosenbrock with GS')
85  hold off
86  print -depsc rosen1b.eps;jpdf('rosen1b')
87
88  figure(3);clf
89  contour(x1,x2,FF',[0:2:10 15:50:1000])
90  hold on
91  xlabel('x_1')
92  ylabel('x_2')
93  plot(x0(1),x0(2),'ro','Markersize',12)
94  plot(1,1,'rs','Markersize',12)
95  plot(xhyb(:,1),xhyb(:,2),'m+','Markersize',12)
96  title('Rosenbrock with GS(5) and BFGS')
97  hold off
98  print -depsc rosen1c.eps;jpdf('rosen1c')
99
100 figure(4);clf
101 mesh(x1,x2,FF')
102 hold on
103 plot3(x0(1),x0(2),rosen(x0')+5,'ro','Markersize',12,'MarkerFaceColor','r')
104 plot3(1,1,rosen([1 1]),'ms','Markersize',12,'MarkerFaceColor','m')
105 plot3(xbfgs(:,1),xbfgs(:,2),rosen(xbfgs)+5,'gd','MarkerFaceColor','g')
106 %plot3(xgs(:,1),xgs(:,2),rosen(xgs)+5,'m+')
107 hold off
108 axis([-3 3 -3 3 0 1000])
109 hh=get(gcf,'children');
110 xlabel('x_1')
111 ylabel('x_2')
112 set(hh,'View',[-177 89.861],'CameraPosition',[-0.585976 11.1811 5116.63]);%
113 print -depsc rosen2.eps;jpdf('rosen2')
114
```

```
1  function stop = outftn(x, optimValues, state)
2
3  global xpath
4  stop=0;
5  xpath=[xpath;x'];
6
7  return
```

16.323 Lecture 2

Nonlinear Optimization

- Constrained nonlinear optimization
- Lagrange multipliers
- Penalty/barrier functions also often used, but will not be discussed here.

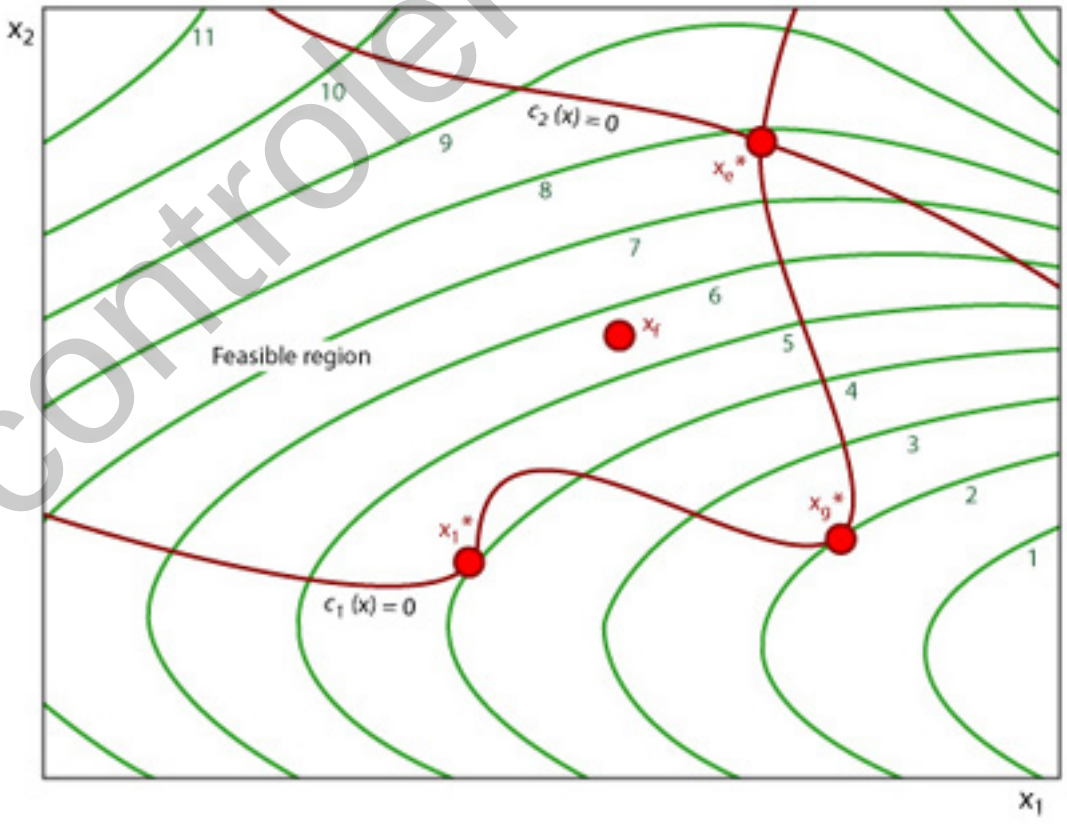


Figure by MIT OpenCourseWare.

Constrained Optimization

- Consider a problem with the next level of complexity: optimization with equality constraints

$$\min_{\mathbf{y}} F(\mathbf{y})$$
$$\text{such that } \mathbf{f}(\mathbf{y}) = 0$$

  a vector of n constraints

- To simplify the notation, assume that the p-state vector $\mathbf{y}$ can be separated into a decision m-vector $\mathbf{u}$ and a state n-vector $\mathbf{x}$ related to the decision variables through the constraints. Problem now becomes:

$$\min_{\mathbf{u}} F(\mathbf{x}, \mathbf{u})$$
$$\text{such that } \mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$$

  – Assume that $p > n$ otherwise the problem is completely specified by the constraints (or over specified).

- One solution approach is **direct substitution**, which involves
  – Solving for $\mathbf{x}$ in terms of $\mathbf{u}$ using $\mathbf{f}$
  – Substituting this expression into F and solving for $\mathbf{u}$ using an unconstrained optimization.
  – Works best if $\mathbf{f}$ is linear (assumption is that not both of $\mathbf{f}$ and F are linear.)

- Example: minimize $F = x_1^2 + x_2^2$ subject to the constraint that $x_1 + x_2 + 2 = 0$
  - Clearly the unconstrained minimum is at $x_1 = x_2 = 0$
  - Substitution in this case gives equivalent problems:

$$\min_{x_2} \tilde{F}_2 = (-2 - x_2)^2 + x_2^2$$

or

$$\min_{x_1} \tilde{F}_1 = x_1^2 + (-2 - x_1)^2$$

for which the solution $(\partial \tilde{F}_2 / \partial x_2 = 0)$ is $x_1 = x_2 = -1$

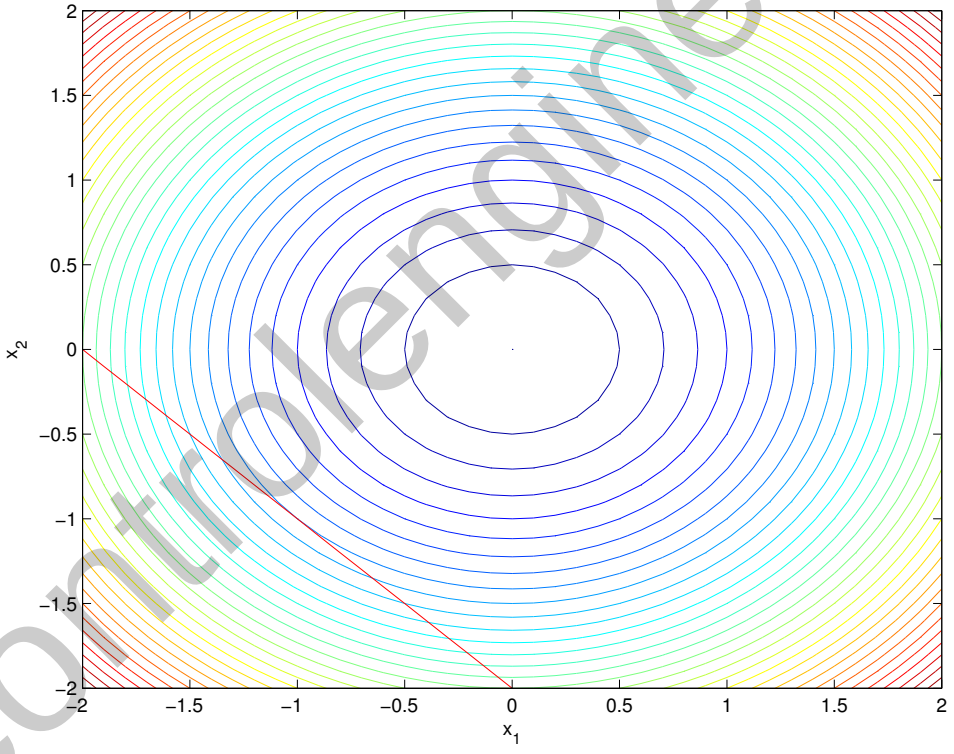


Figure 2.8: Simple function minimization with constraint.

- **Bottom line**: substitution works well for linear constraints, but process hard to generalize for larger systems/nonlinear constraints.

Lagrange Multipliers

- Need a more general strategy - using Lagrange multipliers.
- Since $\mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$, we can adjoin it to the cost with constants

$$\boldsymbol{\lambda}^T = \begin{bmatrix} \lambda_1 & \dots & \lambda_n \end{bmatrix}$$

without changing the function value along the constraint to create **Lagrangian** function

$$L(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = F(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u})$$

- Given values of $\mathbf{x}$ and $\mathbf{u}$ for which $\mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$, consider differential changes to the Lagrangian from differential changes to $\mathbf{x}$ and $\mathbf{u}$:

$$dL = \frac{\partial L}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial L}{\partial \mathbf{u}} d\mathbf{u}$$

where $\frac{\partial L}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial L}{\partial u_1} & \cdots & \frac{\partial L}{\partial u_m} \end{bmatrix}$ (row vector)

- Since $\mathbf{u}$ are the decision variables it is convenient to choose $\boldsymbol{\lambda}$ so that

$$\frac{\partial L}{\partial \mathbf{x}} \triangleq \frac{\partial F}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \equiv 0 \tag{2.1}$$

$$\Rightarrow \boldsymbol{\lambda}^T = -\frac{\partial F}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^{-1} \tag{2.2}$$

- To proceed, must determine what changes are possible to the **cost** keeping the equality constraint satisfied.
  - Changes to $\mathbf{x}$ and $\mathbf{u}$ are such that $\mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$, then

$$d\mathbf{f} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} d\mathbf{u} \equiv 0 \tag{2.3}$$

$$\Rightarrow d\mathbf{x} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} d\mathbf{u} \tag{2.4}$$

- Then the allowable cost variations are

$$dF = \frac{\partial F}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial F}{\partial \mathbf{u}} d\mathbf{u} \tag{2.5}$$

$$= \left(-\frac{\partial F}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} + \frac{\partial F}{\partial \mathbf{u}} \right) d\mathbf{u}$$

$$= \left(\frac{\partial F}{\partial \mathbf{u}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right) d\mathbf{u} \tag{2.6}$$

$$\equiv \frac{\partial L}{\partial \mathbf{u}} d\mathbf{u} \tag{2.7}$$

- So the gradient of the cost F with respect to $\mathbf{u}$ while keeping the constraint $\mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$ is just

$$\frac{\partial L}{\partial \mathbf{u}}$$

and we need this gradient to be zero to have a stationary point so that $dF = 0 \; \forall \; d\mathbf{u} \neq 0$.

- Thus the necessary conditions for a stationary value of F are

$$\frac{\partial L}{\partial \mathbf{x}} = 0 \tag{2.8}$$

$$\frac{\partial L}{\partial \mathbf{u}} = 0 \tag{2.9}$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = 0 \tag{2.10}$$

which are $2n + m$ equations in $2n + m$ unknowns.

- Note that Eqs. 2.8–2.10 can be written compactly as

$$\frac{\partial L}{\partial \mathbf{y}} = 0 \tag{2.11}$$

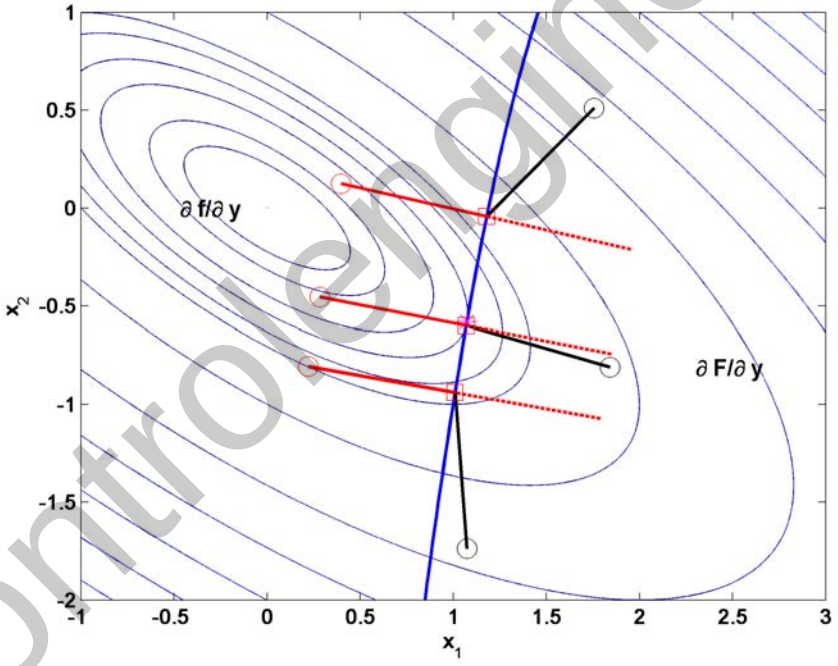
$$\frac{\partial L}{\partial \boldsymbol{\lambda}} = 0 \tag{2.12}$$

  - The solutions of which give the stationary points.

Intuition

- Can develop the intuition that the constrained solution will be a **point of tangency** of the constant cost curves and the constraint function
  - No further improvements possible while satisfying the constraints.

- Equivalent to saying that the gradient of the cost ftn (normal to the constant cost curve) $\partial F/\partial \mathbf{y}$ [black lines] must lie in the space spanned by the constraint gradients $\partial \mathbf{f}/\partial \mathbf{y}$ [red lines]
  - Means cost cannot be improved without violating the constraints.
  - In 2D case, this corresponds to $\partial F/\partial \mathbf{y}$ being collinear to $\partial f/\partial \mathbf{y}$



- **Note:** If this were not true, then it would be possible to take $d\mathbf{y}$ in the negative of the direction of the component of the cost gradient orthogonal to the constraint gradient, thereby reducing the cost and still satisfying the constraint.
  - Can see that at the points on the constraint above and blow the optimal value of x_2

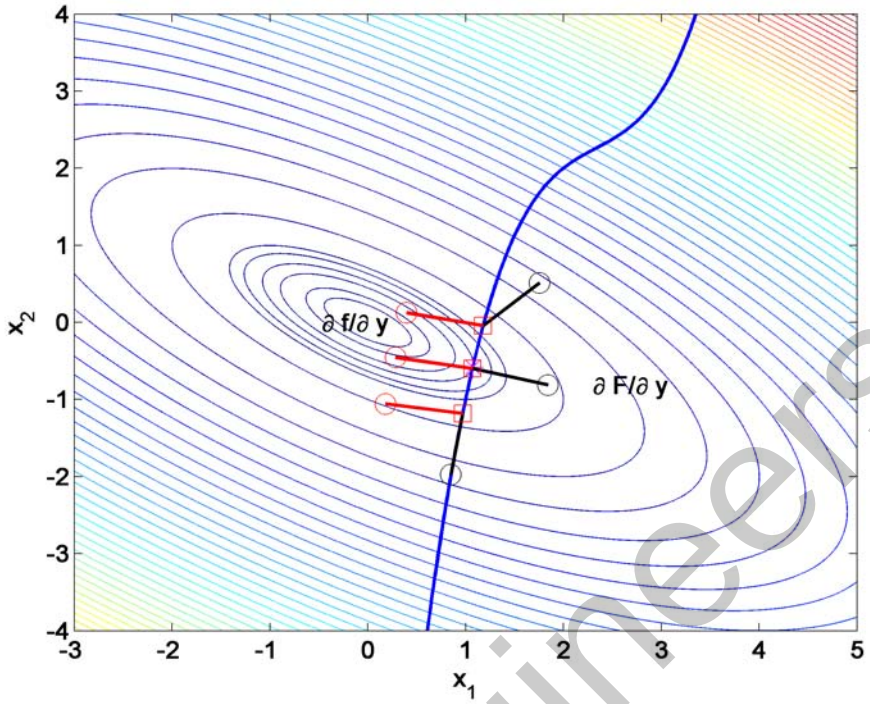


Figure 2.9: Minimization with equality constraints: shows that function and cost gradients are nearly collinear near optimal point and clearly not far away.

$$f(x_1, x_2) = x_2 - ((x_1)^3 - (x_1)^2 + (x_1) + 2) = 0 \text{ and } F = \tfrac{1}{2}\mathbf{x}^T \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$$

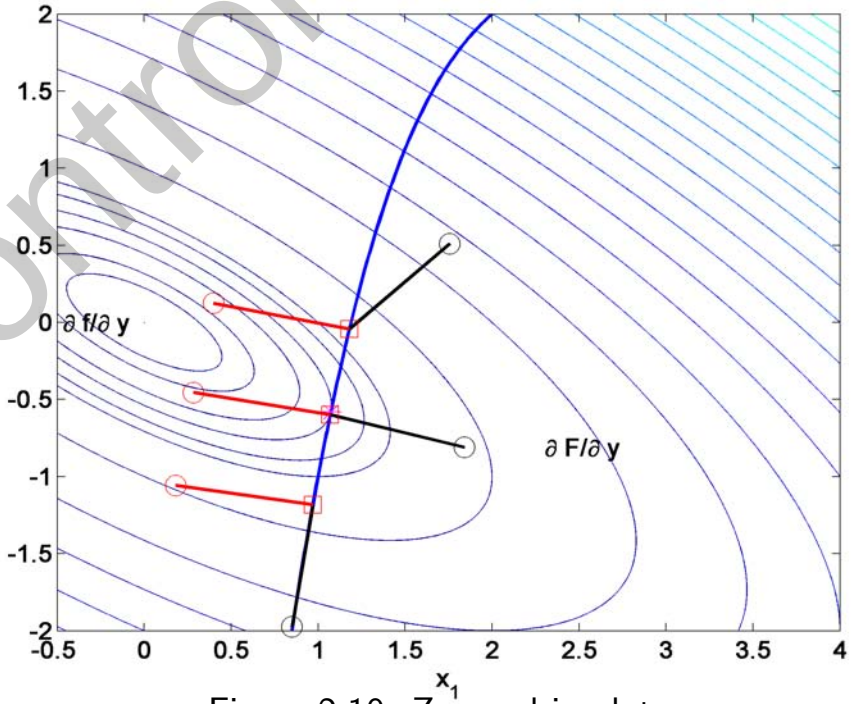
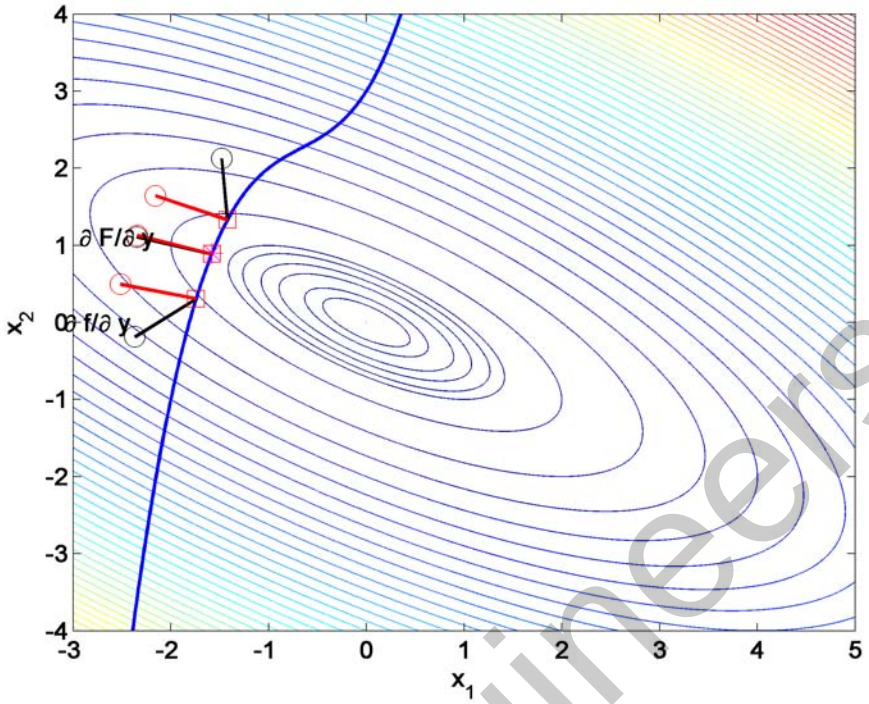


Figure 2.10: Zoomed in plot.



$f(x_1, x_2) = x_2 - ((x_1 - 2)^3 - (x_1 - 2)^2 + (x_1 - 2) + 2) = 0$ and $F = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$

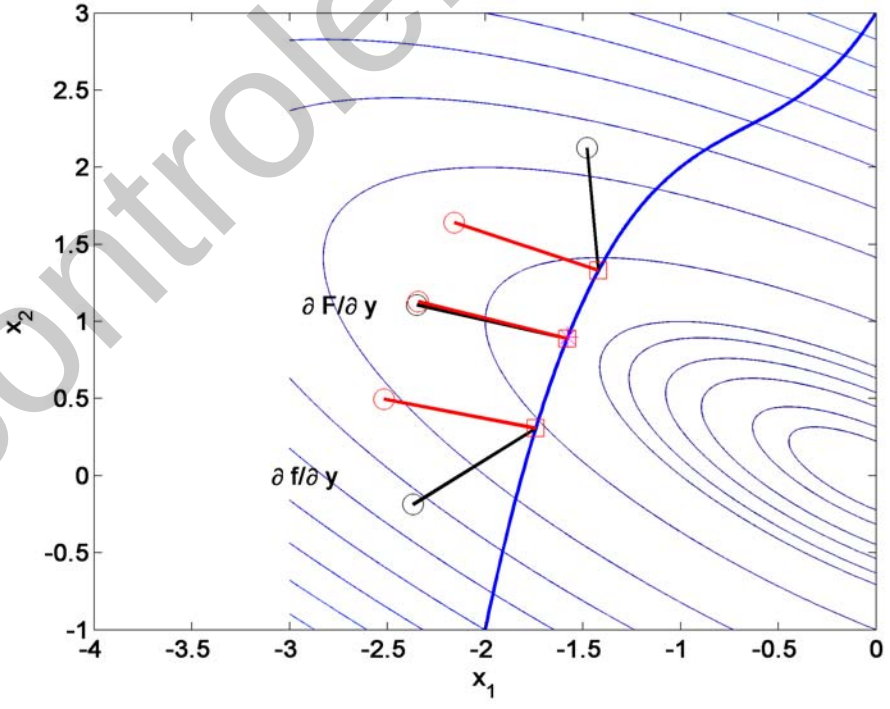


Figure 2.11: Change constraint - note that the cost and constraint gradients are collinear, but now aligned

- Generalize this intuition of being "collinear" to larger state dimensions to notion that the cost gradient **must lie in the space spanned** by the constraint gradients.
  - Equivalent to saying that it is possible to express the cost gradient as a linear combination of the constraint gradients
  - Again, if this was not the case, then improvements can be made to the cost without violating the constraints.

- So that at a constrained minimum, there must exist constants such that the cost gradient satisfies:

$$\frac{\partial F}{\partial \mathbf{y}} = -\lambda_1 \frac{\partial f_1}{\partial \mathbf{y}} - \lambda_2 \frac{\partial f_2}{\partial \mathbf{y}} - \cdots - \lambda_n \frac{\partial f_n}{\partial \mathbf{y}} \tag{2.13}$$

$$= -\boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \tag{2.14}$$

or equivalently that

$$\frac{\partial F}{\partial \mathbf{y}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{y}} = 0$$

which is, of course, the same as Eq. 2.11.

Constrained Example

- Minimize $F(x_1, x_2) = x_1^2 + x_2^2$ subject to $f(x_1, x_2) = x_1 + x_2 + 2 = 0$
  - Form the Lagrangian

$$L \triangleq F(x_1, x_2) + \lambda f(x_1, x_2) = x_1^2 + x_2^2 + \lambda(x_1 + x_2 + 2)$$

  - Where λ is the Lagrange multiplier

- The solution approach without constraints is to find the stationary point of $F(x_1, x_2)$ $(\partial F/\partial x_1 = \partial F/\partial x_2 = 0)$
  - With constraints we find the stationary points of L

$$\mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \frac{\partial L}{\partial \mathbf{y}} = 0, \quad \frac{\partial L}{\partial \lambda} = 0$$

which gives

$$\begin{aligned} \frac{\partial L}{\partial x_1} &= 2x_1 + \lambda = 0 \\ \frac{\partial L}{\partial x_2} &= 2x_2 + \lambda = 0 \\ \frac{\partial L}{\partial \lambda} &= x_1 + x_2 + 2 = 0 \end{aligned}$$

- This gives 3 equations in 3 unknowns, solve to find $x_1^\star = x_2^\star = -1$
- The key point here is that due to the constraint, the selection of x_1 and x_2 during the minimization are not independent
  - The Lagrange multiplier captures this dependency.
- Difficulty can be solving the resulting equations for the optimal points (can be ugly nonlinear equations)

Inequality Constraints

- Now consider the problem

$$\min_{\mathbf{y}} F(\mathbf{y}) \tag{2.15}$$

$$\text{such that } \mathbf{f}(\mathbf{y}) \leq 0 \tag{2.16}$$

  - Assume that there are n constraints, but do not need to constrain n with respect to the state dimension p since not all inequality constraints will limit a degree of freedom of the solution.

- Have similar picture as before, but now not all constraints are active
  - Black line at top is inactive since $x_1 + x_2 - 1 < 0$ at the optimal value $\mathbf{x} = [1 \ \ -0.60] \Rightarrow$ it does not limit a degree of freedom in the problem.
  - Blue constraint is active, cost lower to the left, but $f_1 > 0$ there

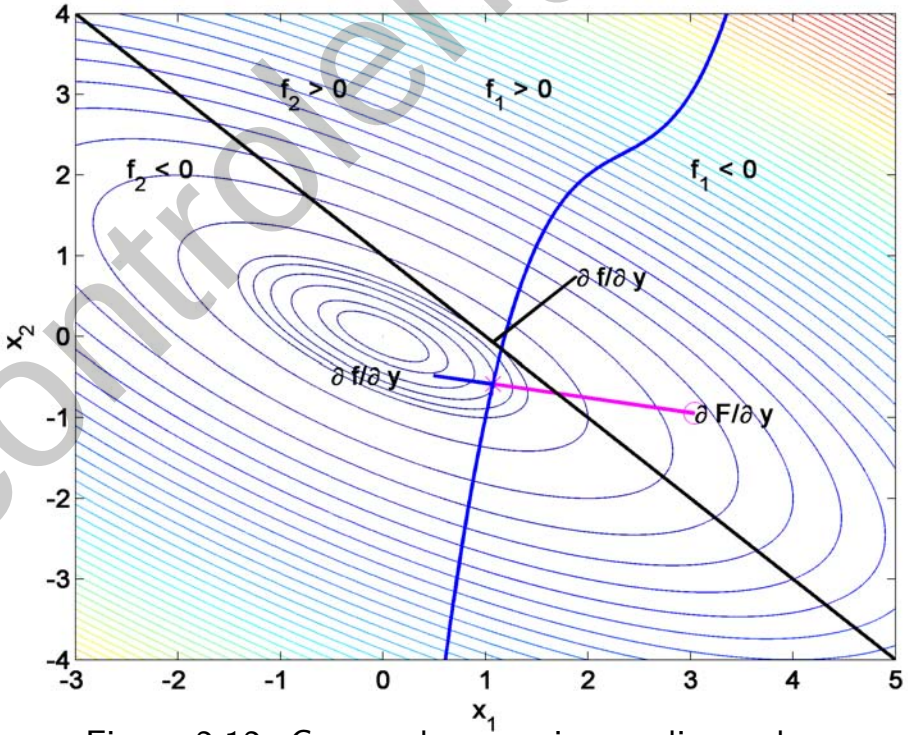
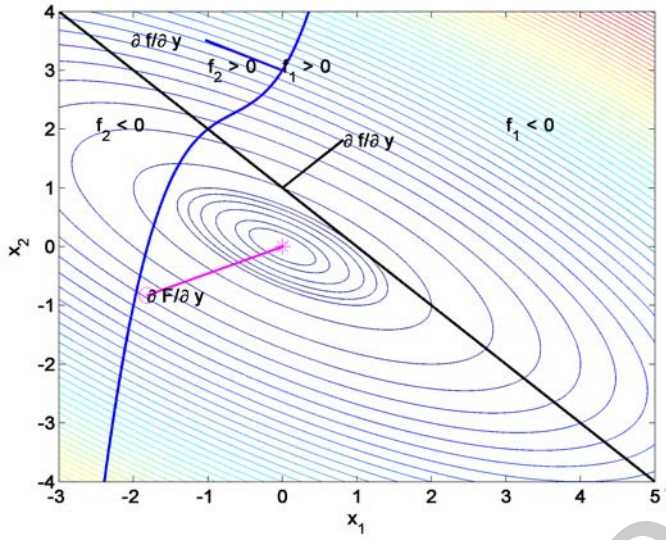
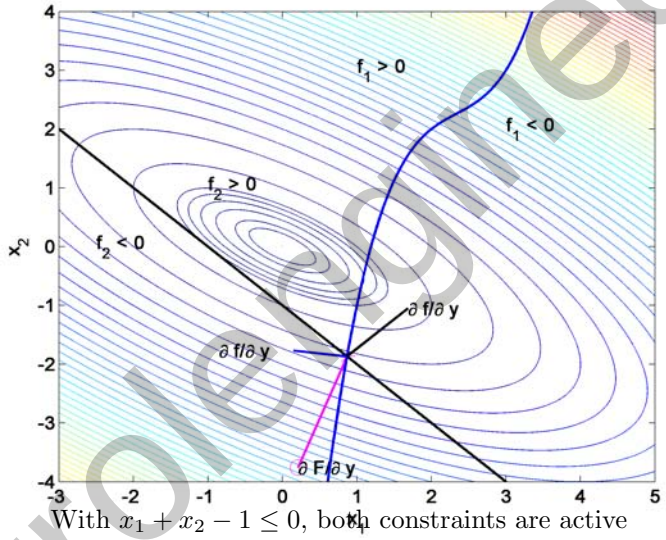


Figure 2.12: Cost and constraint gradients shown





With $x_1 + x_2 - 1 \leq 0$, both constraints are active

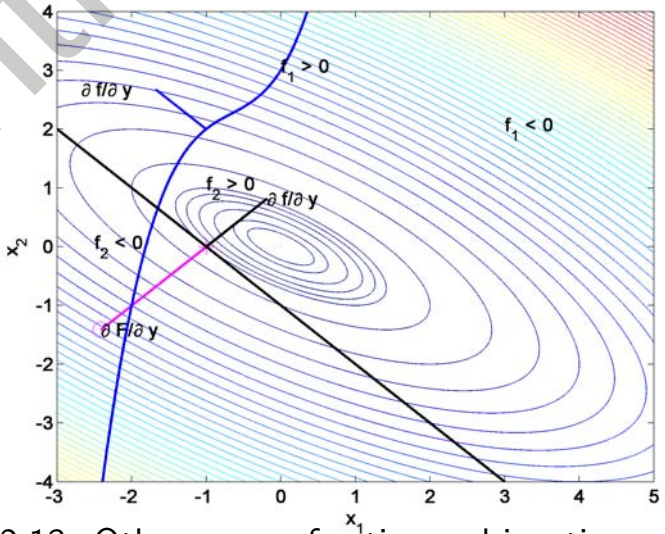


Figure 2.13: Other cases of active and inactive constraints

- Intuition in this case is that at the minimum, the cost gradient must lie in the space spanned by the **active** constraints - so split as:

$$\frac{\partial F}{\partial \mathbf{y}} = -\sum_{\substack{i \\ \text{active}}} \lambda_i \frac{\partial f_i}{\partial \mathbf{y}} - \sum_{\substack{j \\ \text{inactive}}} \lambda_j \frac{\partial f_j}{\partial \mathbf{y}} \tag{2.17}$$

  - And if the constraint is inactive, then can set $\lambda_j = 0$

- With equality constraints, needed the cost and function gradients to be collinear, but they could be in any orientation.

- For inequality constraints, need an additional constraint that is related to the allowable changes in the state.
  - Must restrict condition 2.17 so that the cost gradient points in the direction of the "allowable side" of the constraint ($f < 0$).
    $\Rightarrow$ Cost cannot be reduced without violating constraint.
    $\Rightarrow$ Cost and function gradients must point in opposite directions.
  - Given 2.17, require that $\lambda_i \geq 0$ for active constraints

- **Summary:** Active constraints, $\lambda_i \geq 0$, and Inactive ones $\lambda_j = 0$

- Given this, we can define the same Lagrangian as before $L = F + \boldsymbol{\lambda}^T \mathbf{f}$, and the necessary conditions for optimality are

$$\frac{\partial L}{\partial \mathbf{y}} = 0 \tag{2.18}$$

$$\lambda_i \frac{\partial L}{\partial \lambda_i} = 0 \ \forall i \tag{2.19}$$

  where the second property applies to all constraints
  - Active ones have $\lambda_i \geq 0$ and satisfy $\frac{\partial L}{\partial \lambda_i} = f_i = 0$
  - Inactive ones have $\lambda_i = 0$ and satisfy $\frac{\partial L}{\partial \lambda_i} = f_i < 0$.

- Equations 2.18 and 2.19 are the "essence" of the Kuhn-Tucker theorem in nonlinear programming - more precise statements available with more careful specification of the constraints properties.
  - Must also be careful in specifying the second order conditions for a stationary point to be a minimum - see Bryson and Ho, sections 1.3 and 1.7.

- Note that there is an implicit assumption here of **regularity** – that the active constraint gradients are linearly independent – for the $\lambda' s$ to be well defined.
  - Avoids redundancy

Cost Sensitivity

- Often find that the constraints in the problem are picked somewhat arbitrarily - some flexibility in the limits.
  - Thus it would be good to establish the extent to which those choices impact the solution.

- Note that at the solution point,

$$\frac{\partial L}{\partial \mathbf{y}} = 0 \Rightarrow \frac{\partial F}{\partial \mathbf{y}} = -\boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$$

  If the state changes by $\Delta\mathbf{y}$, would expect change in the

$$\text{Cost} \qquad \Delta F = \frac{\partial F}{\partial \mathbf{y}} \Delta \mathbf{y}$$

$$\text{Constraint} \qquad \Delta \mathbf{f} = \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \Delta \mathbf{y}$$

  So then we have that

$$\Delta F = -\boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \Delta \mathbf{y} = -\boldsymbol{\lambda}^T \Delta \mathbf{f}$$

$$\Rightarrow \qquad \frac{dF}{d\mathbf{f}} = -\boldsymbol{\lambda}^T$$

  - **Sensitivity of the cost to changes in the constraint function is given by the Lagrange Multipliers.**

- For active constraints $\boldsymbol{\lambda} \geq 0$, so expect that $dF/d\mathbf{f} \leq 0$
  - Makes sense because if it is active, then allowing $\mathbf{f}$ to increase will move the constraint boundary in the direction of reducing F
  - Correctly predicts that inactive constraints will not have an impact.

Alternative Derivation of Cost Sensitivity

- Revise the constraints so that they are of the form $\mathbf{f} \leq \mathbf{c}$, where $\mathbf{c} \geq 0$ is a constant that is nominally 0.
  - The constraints can be rewritten as $\bar{\mathbf{f}} = \mathbf{f} - \mathbf{c} \leq 0$, which means

$$\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{y}} \equiv \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$$

and assuming the $\bar{\mathbf{f}}$ constraint remains active as we change $\mathbf{c}$

$$\frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{c}} \equiv \frac{\partial \mathbf{f}}{\partial \mathbf{c}} - I = 0$$

- Note that at the solution point,

$$\frac{\partial L}{\partial \mathbf{y}} = 0 \Rightarrow \frac{\partial F}{\partial \mathbf{y}} = -\boldsymbol{\lambda}^T \frac{\partial \bar{\mathbf{f}}}{\partial \mathbf{y}} = -\boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$$

- To study cost sensitivity, must compute $\frac{\partial F}{\partial \mathbf{c}}$. To proceed, note that

$$\begin{aligned} \frac{\partial F}{\partial \mathbf{c}} &= \frac{\partial F}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{c}} \\ &= -\boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{c}} \\ &= -\boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{c}} \\ &= -\boldsymbol{\lambda}^T \end{aligned}$$

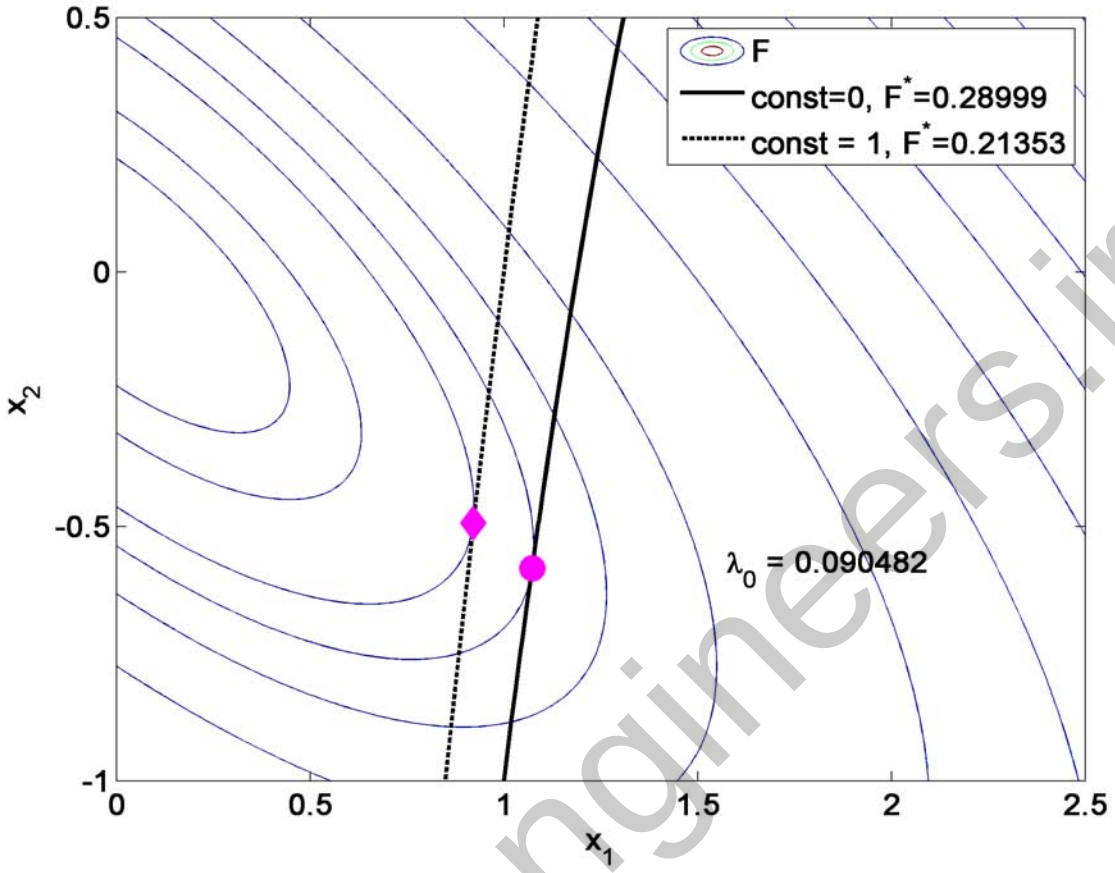


Figure 2.14: Shows that changes to the constraint impact cost in a way that can be predicted from the Lagrange Multiplier.

Simple Constrained Example

- Consider case $F = x_1^2 + x_1x_2 + x_2^2$ and $x_2 \geq 1$, $x_1 + x_2 \leq 3$
- Form Lagrangian

$$L = x_1^2 + x_1x_2 + x_2^2 + \lambda_1(1 - x_2) + \lambda_2(x_1 + x_2 - 3)$$

- Form necessary conditions:

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= 2x_1 + x_2 + \lambda_2 = 0 \\
\frac{\partial L}{\partial x_2} &= x_1 + 2x_2 - \lambda_1 + \lambda_2 = 0 \\
\lambda_1 \frac{\partial L}{\partial \lambda_1} &= \lambda_1(1 - x_2) = 0 \\
\lambda_2 \frac{\partial L}{\partial \lambda_2} &= \lambda_2(x_1 + x_2 - 3) = 0
\end{aligned}$$

- Now consider the various options:
  - Assume $\lambda_1 = \lambda_2 = 0$ both inactive

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= 2x_1 + x_2 = 0 \\
\frac{\partial L}{\partial x_2} &= x_1 + 2x_2 = 0
\end{aligned}$$

    gives solution $x_1 = x_2 = 0$ as expected, but does not satisfy all the constraints
  - Assume $\lambda_1 = 0$ (inactive), $\lambda_2 \geq 0$ (active)

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= 2x_1 + x_2 + \lambda_2 = 0 \\
\frac{\partial L}{\partial x_2} &= x_1 + 2x_2 + \lambda_2 = 0 \\
\lambda_2 \frac{\partial L}{\partial \lambda_2} &= \lambda_2(x_1 + x_2 - 3) = 0
\end{aligned}$$

    which gives solution $x_1 = x_2 = 3/2$, which satisfies the constraints, but $F = 6.75$ and $\lambda_2 = -9/2$

– Assume $\lambda_1 \geq 0$ (active), $\lambda_2 = 0$ (inactive)

$$\begin{aligned} \frac{\partial L}{\partial x_1} &= 2x_1 + x_2 = 0 \\ \frac{\partial L}{\partial x_2} &= x_1 + 2x_2 - \lambda_1 = 0 \\ \lambda_1 \frac{\partial L}{\partial \lambda_1} &= \lambda_1(1 - x_2) = 0 \end{aligned}$$

gives solution $x_1 = -1/2$, $x_2 = 1$, $\lambda_1 = 3/2$ which satisfies the constraints, and $F = 0.75$

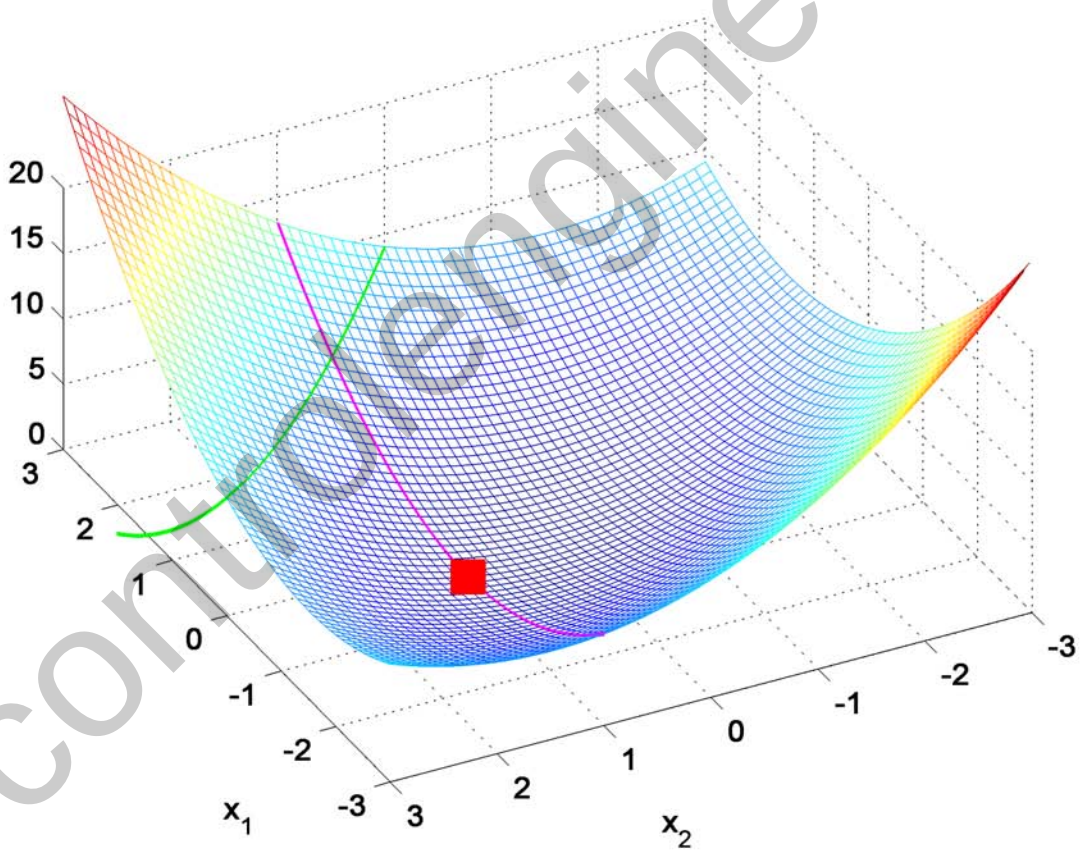


Figure 2.15: Simple example

Code to generate Figure 2.12

```
%
% 16.323 Spr 2008
% Plot of cost ftns and constraints

clear all;close all;
set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontWeight','demi')
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontWeight','demi')

global g G f

F=[];g=[0;0];G=[1 1;1 2];

testcase=0
if testcase
    f=inline('(1*(x1+1).^3-1*(x1+1).^2+1*(x1+1)+2)');
    dfdx=inline('(3*1*(x1+1).^2-2*1*(x1+1)+1)');
else
    f=inline('(1*(x1-2).^3-1*(x1-2).^2+1*(x1-2)+2)');
    dfdx=inline('(3*1*(x1-2).^2-2*1*(x1-2)+1)');
end

x1=-3:.01:5;x2=-4:.01:4;
for ii=1:length(x1);
    for jj=1:length(x2);
        X=[x1(ii) x2(jj)]';
        F(ii,jj)=g'*X+X'*G*X/2;
    end;
end;
figure(1);clf
contour(x1,x2,F',[min(min(F)) .05 .1 .2 .29 .4 .5 1:1:max(max(F))]);
xlabel('x_1' )
ylabel('x_2' )
hold on;
plot(x1,f(x1),'LineWidth',2);

% X=FMINCON(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON,OPTIONS)
xx=fmincon('meshf',[0;0],[],[],[],[],[],[],'meshc');
hold on
plot(xx(1),xx(2),'m*','MarkerSize',12)
axis([-3 5 -4 4]);

Jx=[];
[kk,II1]=min(abs(x1-xx(1)))
[kk,II2]=min(abs(x1-1.1*xx(1)))
[kk,II3]=min(abs(x1-0.9*xx(1)))
ll=[II1 II2 II3];
gam=.8; % line scaling
for ii=1:length(ll)
    X=[x1(ll(ii));f(x1(ll(ii)))]
    Jx(ii,:)=(g+G*X)';
    X2=X+Jx(ii,:)'*gam/norm(Jx(ii,:));

    Nx1=X(1);
    df=[-dfdx(Nx1);1];     % x_2=f(x_1) ==> x_2 - f(x_1) < =0

    X3=[Nx1;f(Nx1)];
    X4=X3+df*gam/norm(df);

    plot(X2(1),X2(2),'ko','MarkerSize',12)
    plot(X(1),X(2),'ks','MarkerSize',12)
    plot([X(1);X2(1)],[X(2);X2(2)],'k-','LineWidth',2)
    plot(X4(1),X4(2),'ro','MarkerSize',12)
    plot(X3(1),X3(2),'rs','MarkerSize',12)
    plot([X4(1);X3(1)],[X4(2);X3(2)],'r-','LineWidth',2)
    if ii==1;
        text([1.25*X2(1)],[X2(2)],'\partial F/\partial y' )
```

```
        text([X4(1)-.75],[0*X4(2)],'\partial f/\partial y' )
    end
end
hold off

%%%%%%%%%%%%%%%%%%%%%%%

f2=inline('-1*x1-1');global f2
df2dx=inline('-1*ones(size(x))');

figure(3);gam=2;
contour(x1,x2,F',[min(min(F)) .05 .1 .2 .3 .4 .5 1:1:max(max(F))]);
xlabel('x_1' );ylabel('x_2'  )

xx=fmincon('meshf',[0;0],[],[],[],[],[],[],'meshc2');
hold on
Jx=(g+G*xx)';
X2=xx+Jx'*gam/norm(Jx);
plot(xx(1),xx(2),'m*','MarkerSize',12)
plot(X2(1),X2(2),'mo','MarkerSize',12);
plot([xx(1);X2(1)],[xx(2);X2(2)],'m-','LineWidth',2)
text([X2(1)],[X2(2)],'\partial F/\partial y')
hold off

hold on;
plot(x1,f(x1),'LineWidth',2);
text(-1,1,'f_2 > 0')
text(-2.5,0,'f_2 < 0')
plot(x1,f2(x1),'k-','LineWidth',2);
text(3,2,'f_1 < 0')
if testcase
    text(0,3,'f_1 > 0')
else
    text(1,3,'f_1 > 0')
end

dd=[xx(1) 0 xx(1)]';
X=[dd f(dd)];
df=[-dfdx(dd) 1*ones(size(dd))];
X2=X+gam*df/norm(df);
for ii=3
    plot([X(ii,1);X2(ii,1)],[X(ii,2);X2(ii,2)],'LineWidth',2)
    text([X2(ii,1)-1],[X2(ii,2)],'\partial f/\partial y')
end
X=[dd f2(dd)];
df2=[-df2dx(dd) 1*ones(size(dd))];
X2=X+gam*df2/norm(df2);
%for ii=1:length(X)
for ii=1
    plot([X(ii,1);X2(ii,1)],[X(ii,2);X2(ii,2)],'k','LineWidth',2)
    text([X2(ii,1)],[X2(ii,2)],'\partial f/\partial y')
end
hold off

%%%%%%%%%%%%%%%%%%%%%%%

f2=inline('-1*x1+1');global f2
df2dx=inline('-1*ones(size(x))');

figure(4);clf;gam=2;
contour(x1,x2,F',[min(min(F)) .05 .1 .2 .3 .4 .5 1:1:max(max(F))]);
xlabel('x_1');ylabel('x_2')

xx=fmincon('meshf',[1;-1],[],[],[],[],[],[],'meshc2');
hold on
Jx=(g+G*xx)';
X2=xx+Jx'*gam/norm(Jx);
plot(xx(1),xx(2),'m*','MarkerSize',12)
plot(X2(1),X2(2),'mo','MarkerSize',12);
plot([xx(1);X2(1)],[xx(2);X2(2)],'m-','LineWidth',2)
text([X2(1)],[X2(2)],'\partial F/\partial y')
hold off
```

```
139
140 hold on;
141 plot(x1,f(x1),'LineWidth',2);
142 text(-1,3,'f_2 > 0')
143 text(-2.5,2,'f_2 < 0')
144 plot(x1,f2(x1),'k-','LineWidth',2);
145 text(3,2,'f_1 < 0')
146 if testcase
147     text(0,3,'f_1 > 0')
148 else
149     text(1,3,'f_1 > 0')
150 end
151
152 dd=[xx(1) 0 xx(1)]';
153 X=[dd f(dd)];
154 df=[-dfdx(dd) 1*ones(size(dd))];
155 X2=X+gam*df/norm(df);
156 for ii=3
157     plot([X(ii,1);X2(ii,1)],[X(ii,2);X2(ii,2)],'LineWidth',2)
158     text([X2(ii,1)-1],[X2(ii,2)],'\partial f/\partial y')
159 end
160 X=[dd f2(dd)];
161 df2=[-df2dx(dd) 1*ones(size(dd))];
162 X2=X+gam*df2/norm(df2);
163 %for ii=1:length(X)
164 for ii=1
165     plot([X(ii,1);X2(ii,1)],[X(ii,2);X2(ii,2)],'k','LineWidth',2)
166     text([X2(ii,1)],[X2(ii,2)],'\partial f/\partial y')
167 end
168 hold off
169
170 %%%%%%%%%%%%%%%%%%%%%%%
171
172 if testcase
173     figure(1)
174     print  -r300 -dpng mesh1b.png;%jpdf('mesh1b');
175     axis([-4 0 -1 3]);
176     print  -r300 -dpng mesh1c.png;%jpdf('mesh1c');
177     figure(3)
178     print  -r300 -dpng mesh2.png;%jpdf('mesh2');
179     figure(4)
180     print  -r300 -dpng mesh2a.png;%jpdf('mesh2a');
181 else
182     figure(1)
183     print  -r300 -dpng mesh1.png;%jpdf('mesh1');
184     axis([-.5 4 -2 2]);
185     print  -r300 -dpng mesh1a.png;%jpdf('mesh1a');
186     figure(3)
187     print  -r300 -dpng mesh4.png;%jpdf('mesh4');
188     figure(4)
189     print  -r300 -dpng mesh4a.png;%jpdf('mesh4a');
190 end
191
192 %
193 % sensitivity study
194 % line given by x_2=f(x_1), and the constraint is that x_2-f(x_1) <= 0
195 % changes are made to the constraint so that x_2-f(x_1) <= alp > 0
196 figure(5);clf
197 contour(x1,x2,F',[min(min(F)) .05 .1  .213 .29 .4 .6:.5:max(max(F))]);
198 xlabel('x_1')
199 ylabel('x_2')
200 hold on;
201 f=inline('(1*(x1-2).^3-1*(x1-2).^2+1*(x1-2)+2)');
202 dfdx=inline('(3*1*(x1-2).^2-2*1*(x1-2)+1)');
203 plot(x1,f(x1),'k-','LineWidth',2);
204 alp=1;
205 plot(x1,f(x1)+alp,'k--','LineWidth',2);
206
207 global alp
208 [xx1,temp,temp,temp,lam1]=fmincon('meshf',[0;0],[],[],[],[],[],[],'meshc3');
209 alp=0;
210 [xx0,temp,temp,temp,lam0]=fmincon('meshf',[0;0],[],[],[],[],[],[],'meshc3');
```

```
211
212 [meshf(xx0) lam0.ineqnonlin;meshf(xx1) lam1.ineqnonlin]
213
214 legend('F',['const=0, F^*=',num2str(meshf(xx0))],['const = 1, F^*=' ,num2str(meshf(xx1))])
215
216 hold on
217 plot(xx0(1),xx0(2),'mo','MarkerSize',12,'MarkerFaceColor','m')
218 plot(xx1(1),xx1(2),'md','MarkerSize',12,'MarkerFaceColor','m')
219
220 text(xx0(1)+.5,xx0(2),['\lambda_0 = ',num2str(lam0.ineqnonlin)])
221
222 axis([0 2.5 -1 .5])
223 print  -r300 -dpng mesh5;%jpdf('mesh5');
```

```
1 function F=meshf(X);
2
3 global g G
4
5 F=g'*X+X'*G*X/2;
6
7 end
```

```
1 function [c,ceq]=meshc(X);
2
3 global f
4
5 c=[];
6 %ceq=f(X(1))-X(2);
7 ceq=X(2)-f(X(1));
8
9 return
```

```
1 function [c,ceq]=meshc(X);
2
3 global f f2
4
5 %c=[f(X(1))-X(2);f2(X(1))-X(2)];
6 c=[X(2)-f(X(1));X(2)-f2(X(1))];
7 ceq=[];
8
9 return
```

Code for Simple Constrained Example

```
figure(1),clf
xx=[-3:.1:3]'; for ii=1:length(xx);for jj=1:length(xx); %
FF(ii,jj)= xx(ii)^2+xx(ii)*xx(jj)+xx(jj)^2;end;end;%
hh=mesh(xx,xx,FF);%
hold on;%

plot3(xx,ones(size(xx)),xx.^2+1+xx,'m-','LineWidth',2);%
plot3(xx,3-xx,xx.^2+(3-xx).^2+xx.*(3-xx),'g-','LineWidth',2);%

xlabel('x_1'); ylabel('x_2'); %
hold off; axis([-3 3 -3 3 0 20])%
hh=get(gcf,'children');%
set(hh,'View',[-109 74],'CameraPosition',[-26.5555 13.5307 151.881]);%

xx=fmincon('simplecaseF',[0;0],[],[],[],[],[],[],'simplecaseC');
hold on
plot3(xx(1),xx(2),xx(1).^2+xx(2).^2+xx(1).*xx(2),'rs','MarkerSize',20,'MarkerFace','r')
xx(1).^2+xx(2).^2+xx(1).*xx(2)

print -r300 -dpng simplecase.png;
```

```
function F=simplecaseF(X);

F=X(1)^2+X(1)*X(2)+X(2)^2;

return
```

```
function [c,ceq]=simplecaseC(X);

c=[1-X(2);X(1)+X(2)-3];
ceq=0;

return
```

16.323 Lecture 3

Dynamic Programming

- Principle of Optimality
- Dynamic Programming
- Discrete LQR

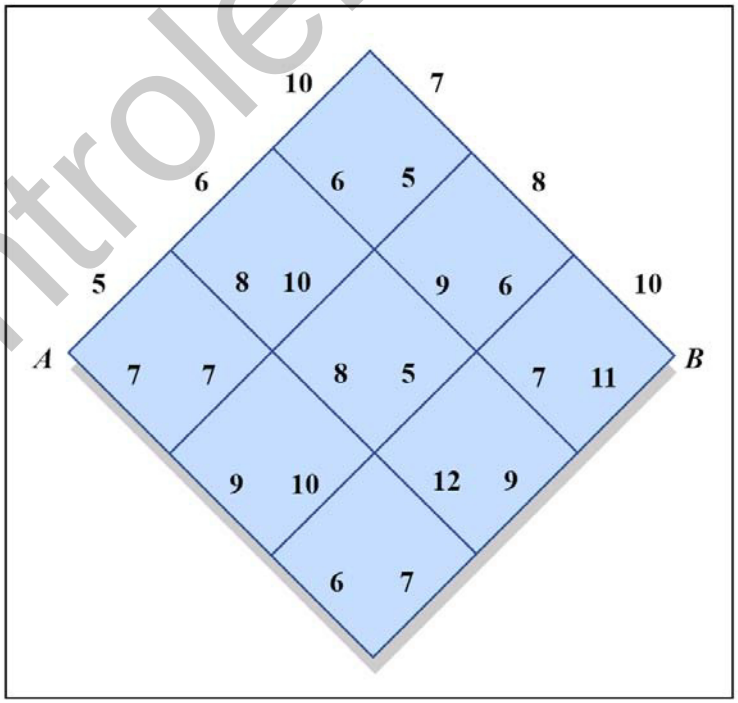
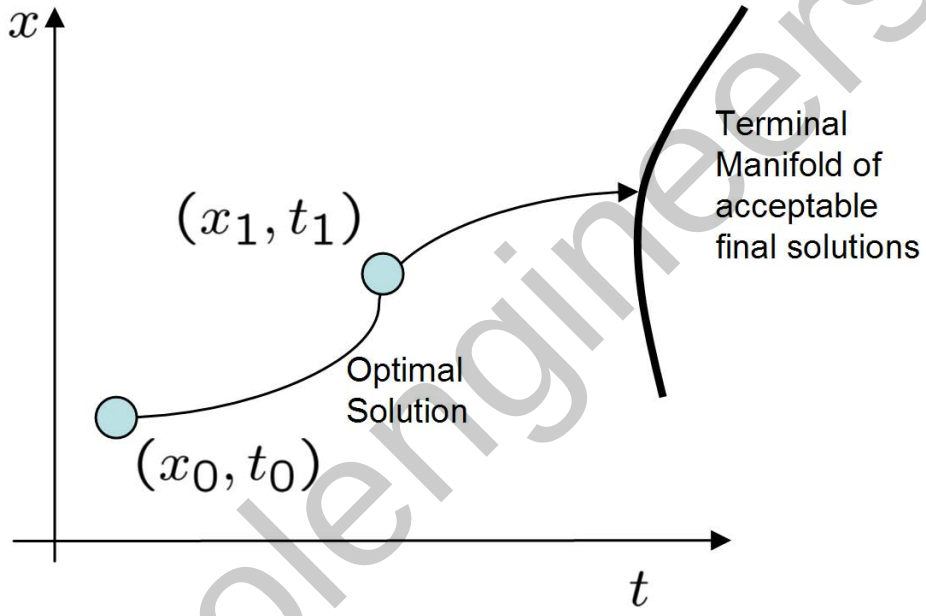


Figure by MIT OpenCourseWare.

Dynamic Programming

- DP is a central idea of control theory that is based on the

  Principle of Optimality: Suppose the optimal solution for a problem passes through some intermediate point (x_1, t_1), then the optimal solution to the same problem starting at (x_1, t_1) must be the continuation of the same path.



- Proof? What would the implications be if it was false?
- This principle leads to:
  - Numerical solution procedure called **Dynamic Programming** for solving multi-stage decision making problems.
  - Theoretical results on the structure of the resulting control law.

- Texts:
  - *Dynamic Programming* (Paperback) by Richard Bellman (Dover)
  - *Dynamic Programming and Optimal Control* (Vol 1 and 2) by D. P. Bertsekas

Classical Examples

- Shortest Path Problems (Bryson figure 4.2.1) – classic robot navigation and/or aircraft flight path problems

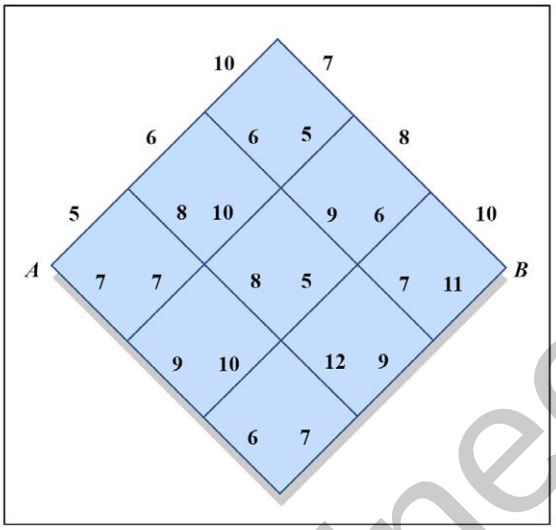


Figure by MIT OpenCourseWare.

- Goal is to travel from A to B in the shortest time possible
  - Travel times for each leg are shown in the figure
  - There are 20 options to get from A to B – could evaluate each and compute travel time, but that would be pretty tedious
- **Alternative approach:** Start at B and work backwards, invoking the principle of optimality along the way.
  - First step backward can be either up (10) or down (11)

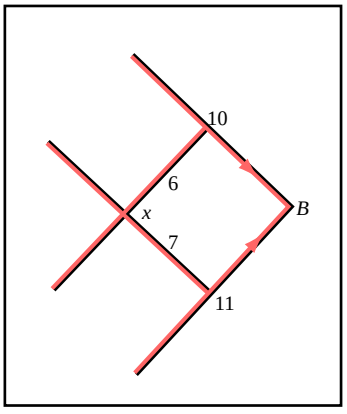


Figure by MIT OpenCourseWare.

- Consider the travel time from point x
  - Can go up and then down $6 + 10 = 16$

  - Or can go down and then up $7 + 11 = 18$
  - Clearly best option from x is go up, then down, with a time of 16
  - From principle of optimality, this is best way to get to B for any path that passes through x.

- Repeat process for all other points, until finally get to initial point $\Rightarrow$ shortest path traced by moving in the directions of the arrows.

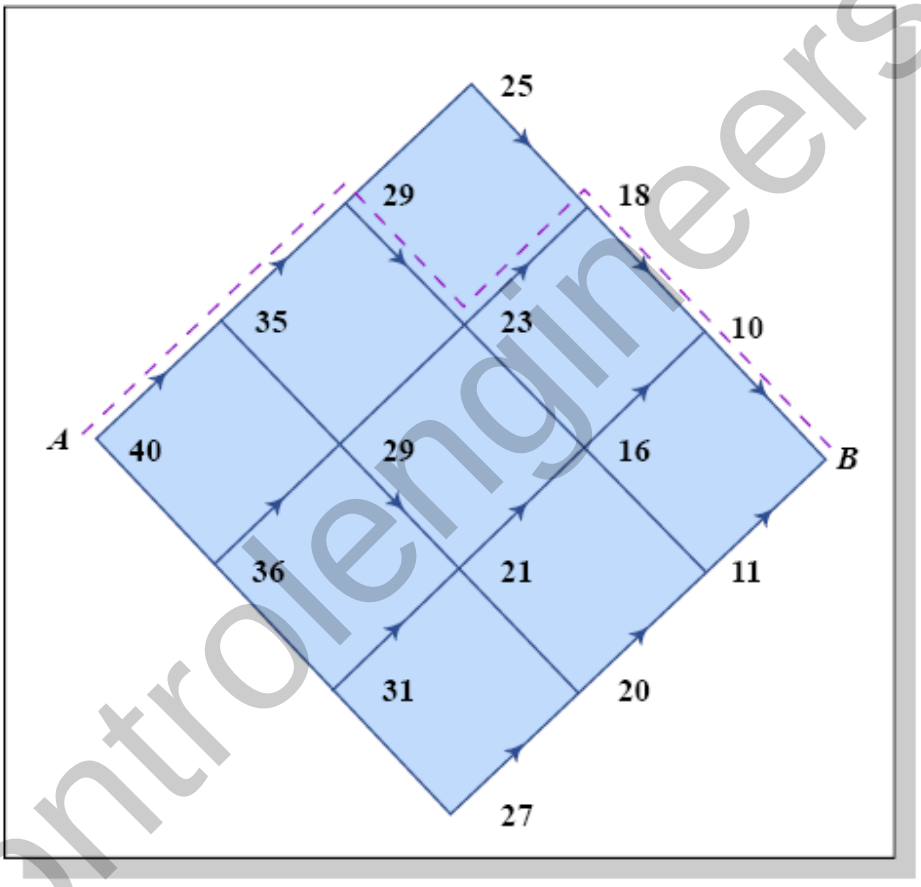


Figure by MIT OpenCourseWare.

- **Key advantage** is that only had to find 15 numbers to solve this problem this way rather than evaluate the travel time for 20 paths
  - Modest difference here, but scales up for larger problems.
  - If $n =$ number of segments on side (3 here) then:
    - ◇ Number of routes scales as $\sim (2n)!/(n!)^2$
    - ◇ Number DP computations scales as $\sim (n+1)^2 - 1$

Example 2

- Routing Problem [Kirk, page 56] through a street maze

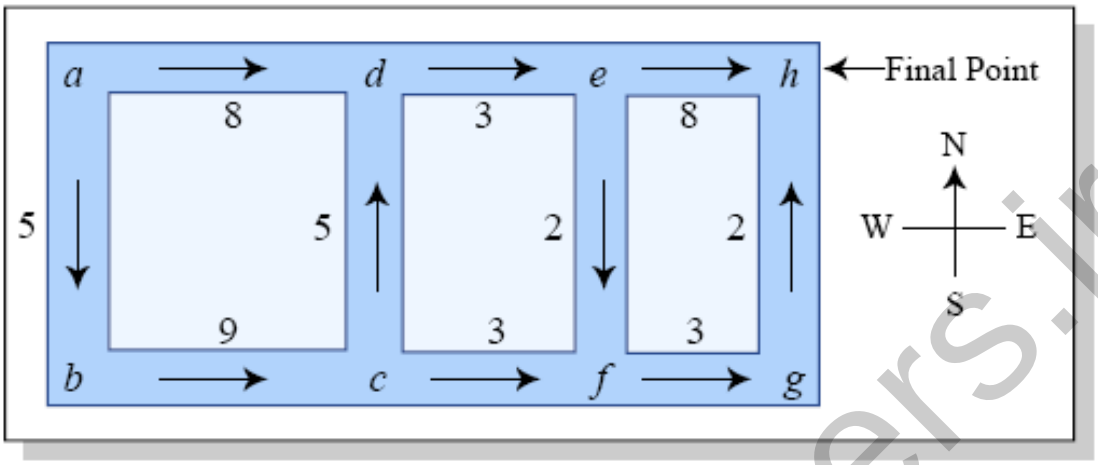


Figure by MIT OpenCourseWare.

  - Similar problem (minimize cost to travel from c to h) with a slightly more complex layout

- Once again, start at end (h) and work backwards
  - Can get to h from e, g directly, but there are 2 paths to h from e.
  - Basics: $J^\star_{gh} = 2$, and

$$J^\star_{fh} = J_{fg} + J^\star_{gh} = 5$$

  - Optimal cost from e to h given by

$$\begin{aligned} J^\star_{eh} &= \min\{J_{efgh}, J_{eh}\} = \min\{[J_{ef} + J^\star_{fh}], J_{eh}\} \\ &= \min\{2+5, 8\} = 7 \quad e \to f \to g \to h \end{aligned}$$

- Also $J^\star_{dh} = J^\star_{de} + J^\star_{eh} = 10$
  - Principle of optimality tells that, since we already know the best way to h from e, do not need to reconsider the various options from e again when starting at d – just use the best.

- Optimal cost from c to h given by

$$\begin{aligned} J^\star_{ch} &= \min\{J_{cdh}, J_{cfh}\} = \min\{[J_{cd} + J^\star_{dh}], [J_{cf} + J^\star_{fh}]\} \\ &= \min\{[5+10], [3+5]\} = 8 \quad c \to f \to g \to h \end{aligned}$$

- Examples show the basis of dynamic programming and use of principle of optimality.
  - In general, if there are numerous options at location α that next lead to locations $x_1, \ldots, x_n$, choose the action that leads to

$$J^\star_{\alpha h} = \min_{x_i} \left\{ [J_{\alpha x_1} + J^\star_{x_1 h}], [J_{\alpha x_2} + J^\star_{x_2 h}], \ldots, [J_{\alpha x_n} + J^\star_{x_n h}] \right\}$$

- Can apply the same process to more general control problems. Typically have to assume something about the system state (and possible control inputs), e.g., bounded, but also discretized.

  Roadmap:
  - Grid the time/state and quantized control inputs.
  - Time/state grid, evaluate necessary control
  - Discrete time problem $\Rightarrow$ discrete LQR
  - Continuous time problem $\Rightarrow$ calculus of variations $\Rightarrow$ cts LQR

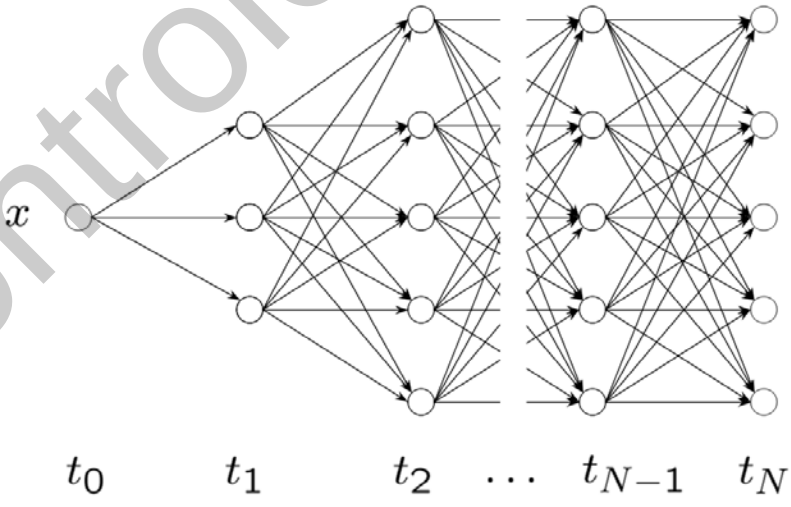


Figure 3.1: Classic picture of discrete time/quantized space grid with the linkages possible through the control commands. Again, it is hard to evaluate all options moving forward through the grid, but we can work backwards and use the principle of optimality to reduce this load.

Classic Control Problem

- Consider the problem of minimizing:

$$\min J = h(x(t_f)) + \int_{t_0}^{t_f} g(x(t), u(t), t)\, dt$$

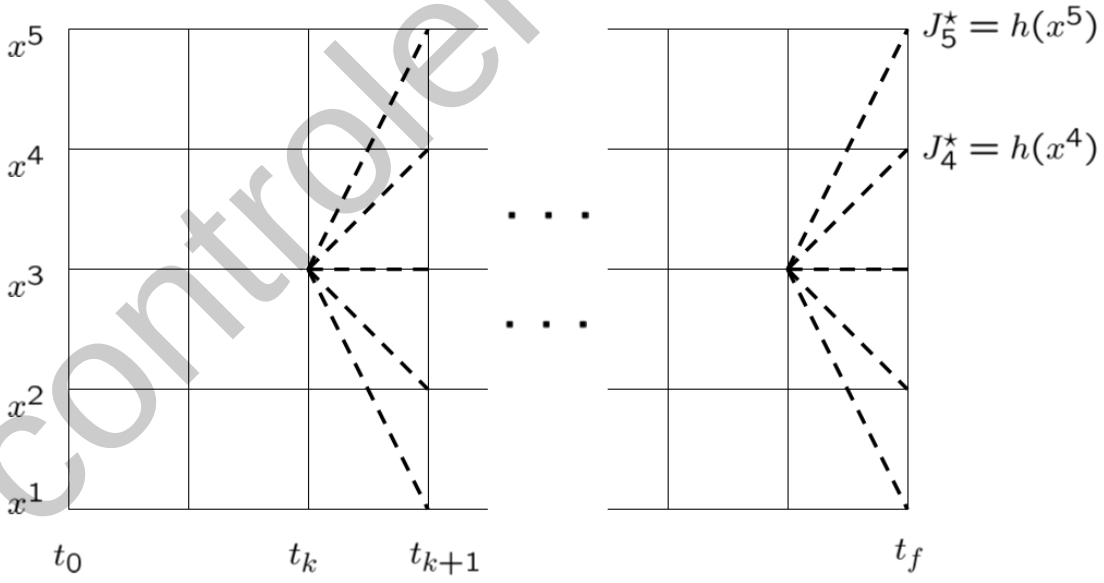
subject to

$$\begin{aligned} \dot{x} &= a(x, u, t) \\ x(t_0) &= \text{fixed} \\ t_f &= \text{fixed} \end{aligned}$$

  - Other constraints on $x(t)$ and $u(t)$ can also be included.

- **Step 1** of solution approach is to develop a grid over space/time.
  - Then look at possible final states $x_i(t_f)$ and evaluate final costs
  - For example, can discretize the state into 5 possible cases $x^1, \ldots, x^5$

$$J_i^\star = h(x_{t_f}^i)\ ,\ \forall\ i$$

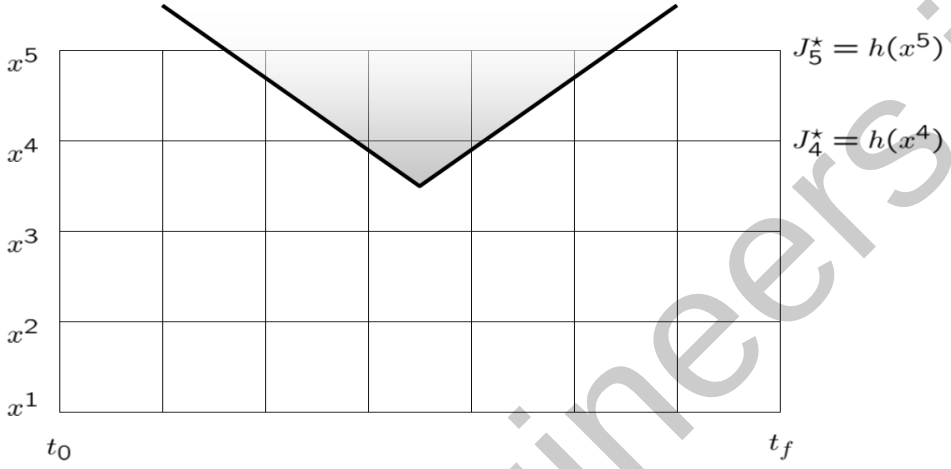


- **Step 2**: back up 1 step in time and consider all possible ways of completing the problem.
  - To evaluate the cost of a control action, must approximate the integral in the cost.

- Consider the scenario where you are at state x^i at time t_k, and apply control u_k^{ij} to move to state x^j at time $t_{k+1} = t_k + \Delta t$.
  - Approximate cost is
$$\int_{t_k}^{t_{k+1}} g(x(t), u(t), t)\, dt \approx g(x_k^i, u_k^{ij}, t_k)\Delta t$$
  - Can solve for control inputs directly from system model:
$$x_{k+1}^j \approx x_k^i + a(x_k^i, u_k^{ij}, t_k)\Delta t \quad \Rightarrow a(x_k^i, u_k^{ij}, t_k) = \frac{x_{k+1}^j - x_k^i}{\Delta t}$$
    which can be solved to find u_k^{ij}.
  - Process is especially simple if the control inputs are affine:
$$\dot{x} = f(x, t) + q(x, t)u$$
    which gives
$$u_k^{ij} = q(x_k^i, t_k)^{-1}\left[\frac{x_{k+1}^j - x_k^i}{\Delta t} - f(x_k^i, t_k)\right]$$
- So for any combination of x_k^i and x_{k+1}^j can evaluate the incremental cost $\Delta J(x_k^i, x_{k+1}^j)$ of making this state transition
- Assuming already know the optimal path from each new terminal point (x_{k+1}^j), can establish optimal path to take from x_k^i using
$$J^\star(x_k^i, t_k) = \min_{x_{k+1}^j}\left[\Delta J(x_k^i, x_{k+1}^j) + J^\star(x_{k+1}^j)\right]$$
  - Then for each x_k^i, output is:
    - ◇ Best x_{k+1}^i to pick, because it gives lowest cost
    - ◇ Control input required to achieve this best cost.
- Then work backwards in time until you reach x_{t_0}, when only one value of x is allowed because of the given initial condition.

Other Considerations

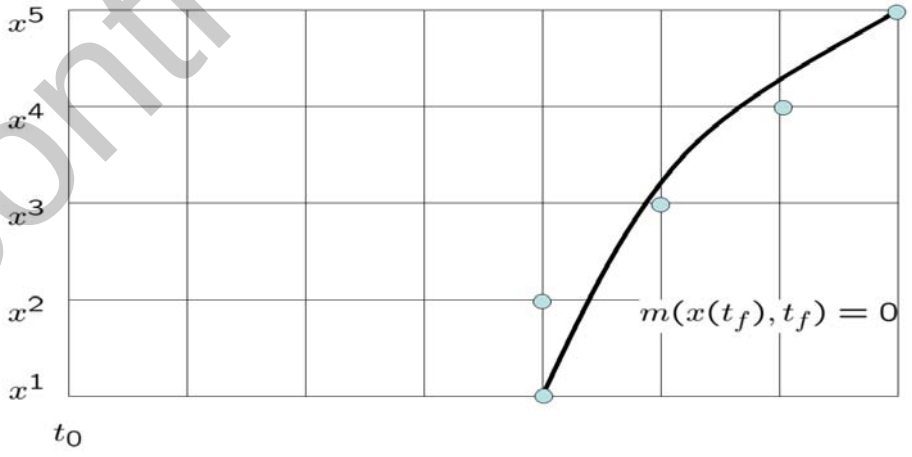
- With bounds on the control, then certain state transitions might not be allowed from 1 time-step to the next.
- With constraints on the state, certain values of $x(t)$ might not be allowed at certain times t.



- Extends to free end time problems, where

$$\min J = h(x(t_f), t_f) + \int_{t_0}^{t_f} g(x(t), u(t), t))\, dt$$

with some additional constraint on the final state $m(x(t_f), t_f) = 0$.



  - Gives group of points that (approximately) satisfy the terminal constraint
  - Can evaluate cost for each, and work backwards from there.

- Process extends to higher dimensional problems where the state is a vector.
  - Just have to define a grid of points in $\mathbf{x}$ and t, which for two dimensions would look like:

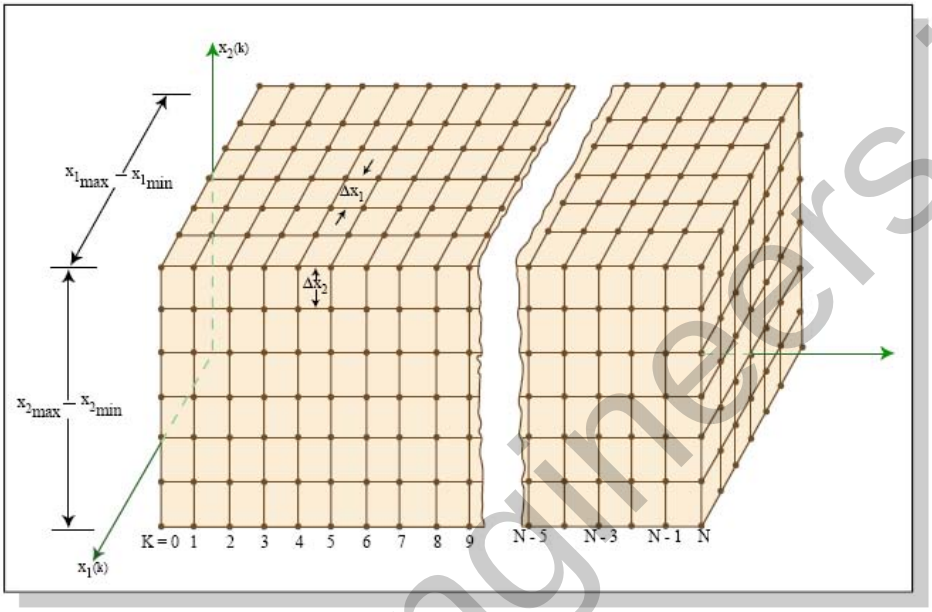


Figure by MIT OpenCourseWare.

Figure 3.2: At any time t_k, have a two dimensional array of grid points.

- Previous formulation picked x's and used those to determine the u's.
  - For more general problems, might be better off picking the u's and using those to determine the propagated x's

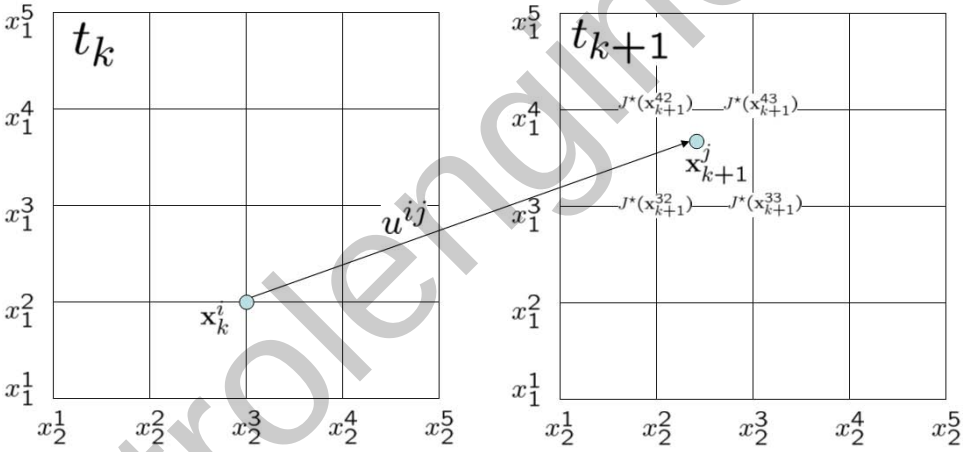
$$\begin{aligned} J^\star(x_k^i, t_k) &= \min_{u_k^{ij}} \left[\Delta J(x_k^i, u_k^{ij}) + J^\star(x_{k+1}^j, t_{k+1}) \right] \\ &= \min_{u_k^{ij}} \left[g(x_k^i, u_k^{ij}, t_k)\Delta t + J^\star(x_{k+1}^j, t_{k+1}) \right] \end{aligned}$$

  - To do this, must quantize the control inputs as well.
  - But then likely that terminal points from one time step to the next will not lie on the state discrete points $\Rightarrow$ must interpolate the cost to go between them.

- **Option 1:** find the control that moves the state from a point on one grid to a point on another.
- **Option 2:** quantize the control inputs, and then evaluate the resulting state for all possible inputs

$$\mathbf{x}_{k+1}^j = \mathbf{x}_k^i + \mathbf{a}(\mathbf{x}_k^i, \mathbf{u}_k^{ij}, t_k)\Delta t$$

  - Issue at that point is that $\mathbf{x}_{k+1}^j$ probably will not agree with the t_{k+1} grid points $\Rightarrow$ must interpolate the available $J^\star$.
  - See, for example, R.E.Larson "A survey of dynamic programming computational procedures", IEEE TAC Dec 1967 (on web) or section 3.6 in Kirk.



- Do this for all admissible $\mathbf{u}_k^{ij}$ and resulting $\mathbf{x}_{k+1}^j$, and then take

$$J^\star(\mathbf{x}_k^i, t_k) = \min_{\mathbf{u}_k^{ij}} J(\mathbf{x}_k^i, \mathbf{u}_k^{ij}, t_k)$$

- Main problem with dynamic programming is how badly it scales.
  - Given N_t points in time and N_x quantized states of dimension n
  - Number of points to consider is $N = N_t N_x^n$
    $\Rightarrow$ "**Curse of Dimensionality**" – R. Bellman, Dynamic Programming (1957) – now from Dover.

DP Example

- See Kirk pg.59:

$$J = x^2(T) + \lambda \int_0^T u^2(t)\, dt$$

with $\dot{x} = ax + u$, where $0 \le x \le 1.5$ and $-1 \le u \le 1$

- Must quantize the state within the allowable values and time within the range $t \in [0, 2]$ using N=2, $\Delta t = T/N = 1$.
  - Approximate the continuous system as:

$$\dot{x} \approx \frac{x(t + \Delta t) - x(t)}{\Delta t} = ax(t) + u(t)$$

    which gives that

$$x_{k+1} = (1 + a\Delta t)x_k + (\Delta t)u_k$$

  - Very common discretization process (Euler integration approximation) that works well if Δt is small

- Use approximate calculation from previous section – cost becomes

$$J = x^2(T) + \lambda \sum_{k=0}^{N-1} u_k^2 \Delta t$$

- Take $\lambda = 2$ and $a = 0$ to simplify things a bit.
  - With $0 \le x(k) \le 1.5$, take x quantized into four possible values $x_k \in \{0, 0.5, 1.0, 1.5\}$
  - With control bounded $|u(k)| \le 1$, assume it is quantized into five possible values: $u_k \in \{-1, -0.5, 0, 0.5, 1\}$

- Start – evaluate cost associated with all possible terminal states

x_2^j	$J_2^\star = h(x_2^j) = (x_2^j)^2$
0	0
0.5	0.25
1	1
1.5	2.25

- Given x_1 and possible x_2, can evaluate the control effort required to make that transition:

$u(1)$	$x_2^j = x_1^i + u(1)$			
x_1^i	**0**	**0.5**	**1**	**1.5**
0	0	0.5	1	**1.5**
0.5	-0.5	0	0.5	1
1	-1	-0.5	0	0.5
1.5	**-1.5**	-1	-0.5	0

which can be used to compute the cost increments:

ΔJ_{12}^{ij}	x_2^j			
x_1^i	**0**	**0.5**	**1**	**1.5**
0	0	0.5	2	XX
0.5	0.5	0	0.5	2
1	2	0.5	0	0.5
1.5	XX	2	0.5	0

and costs at time $t = 1$ given by $J_1 = \Delta J_{12}^{ij} + J_2^\star(x_2^j)$

J_1	x_2^j			
x_1^i	**0**	**0.5**	**1**	**1.5**
0	**0**	0.75	3	XX
0.5	0.5	**0.25**	1.5	4.25
1	2	**0.75**	1	2.75
1.5	XX	2.25	**1.5**	2.25

- Take min across each row to determine best action at each possible $x_1 \Rightarrow J_1^\star(x_1^j)$

x_1^i	$\rightarrow$	x_2^j
0	$\rightarrow$	0
0.5	$\rightarrow$	0.5
1	$\rightarrow$	0.5
1.5	$\rightarrow$	1

- Can repeat the process to find the costs at time $t = 0$ which are $J_0 = \Delta J_{01}^{ij} + J_1^\star(x_1^j)$

J_0	x_1^j			
x_0^i	**0**	**0.5**	**1**	**1.5**
0	**0**	0.75	2.75	XX
0.5	0.5	**0.25**	1.25	3.5
1	2	**0.75**	0.75	2
1.5	XX	2.25	**1.25**	1.5

and again, taking min across the rows gives the best actions:

x_0^i	$\rightarrow$	x_1^j
0	$\rightarrow$	0
0.5	$\rightarrow$	0.5
1	$\rightarrow$	0.5
1.5	$\rightarrow$	1

- So now we have a complete strategy for how to get from any x_0^i to the best x_2 to minimize the cost
  - This process can be highly automated, and this clumsy presentation is typically not needed.

Discrete LQR

- For most cases, dynamic programming must be solved numerically – often quite challenging.
- A few cases can be solved analytically – discrete LQR (linear quadratic regulator) is one of them
- **Goal:** select control inputs to minimize

$$J = \frac{1}{2}\mathbf{x}_N^T H \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k]$$

so that

$$g_d(\mathbf{x}_k, \mathbf{u}_k) = \frac{1}{2}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k\right)$$

subject to the dynamics

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k$$

– Assume that $H = H^T \geq 0$, $Q = Q^T \geq 0$, and $R = R^T > 0$

– Including any other constraints greatly complicates problem

- Clearly $J_N^\star[\mathbf{x}_N] = \frac{1}{2}\mathbf{x}_N^T H \mathbf{x}_N \Rightarrow$ now need to find $J_{N-1}^\star[\mathbf{x}_{N-1}]$

$$\begin{aligned} J_{N-1}^\star[\mathbf{x}_{N-1}] &= \min_{\mathbf{u}_{N-1}} \{g_d(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}) + J_N^\star[\mathbf{x}_N]\} \\ &= \min_{\mathbf{u}_{N-1}} \frac{1}{2}\left\{\mathbf{x}_{N-1}^T Q_{N-1}\mathbf{x}_{N-1} + \mathbf{u}_{N-1}^T R_{N-1}\mathbf{u}_{N-1} + \mathbf{x}_N^T H \mathbf{x}_N]\right\} \end{aligned}$$

- Note that $\mathbf{x}_N = A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}$, so that

$$\begin{aligned} J_{N-1}^\star[\mathbf{x}_{N-1}] = \min_{\mathbf{u}_{N-1}} \frac{1}{2}\Big\{&\mathbf{x}_{N-1}^T Q_{N-1}\mathbf{x}_{N-1} + \mathbf{u}_{N-1}^T R_{N-1}\mathbf{u}_{N-1} \\ &+ \{A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}\}^T H \{A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}\}\Big\} \end{aligned}$$

- Take derivative with respect to the control inputs

$$\frac{\partial J^{\star}_{N-1}[\mathbf{x}_{N-1}]}{\partial \mathbf{u}_{N-1}} = \mathbf{u}^T_{N-1}R_{N-1} + \{A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}\}^T HB_{N-1}$$

- Take transpose and set equal to 0, yields

$$\left[R_{N-1} + B^T_{N-1}HB_{N-1}\right]\mathbf{u}_{N-1} + B^T_{N-1}HA_{N-1}\mathbf{x}_{N-1} = 0$$

- Which suggests a couple of key things:
  - The best control action at time $N-1$, is a linear state feedback on the state at time $N-1$:

$$\begin{aligned}\mathbf{u}^{\star}_{N-1} &= -\left[R_{N-1} + B^T_{N-1}HB_{N-1}\right]^{-1} B^T_{N-1}HA_{N-1}\mathbf{x}_{N-1} \\ &\equiv -F_{N-1}\mathbf{x}_{N-1}\end{aligned}$$

  - Furthermore, can show that

$$\frac{\partial^2 J^{\star}_{N-1}[\mathbf{x}_{N-1}]}{\partial \mathbf{u}^2_{N-1}} = R_{N-1} + B^T_{N-1}HB_{N-1} > 0$$

    so that the stationary point is a minimum

- With this control decision, take another look at

$$\begin{aligned}J^{\star}_{N-1}[\mathbf{x}_{N-1}] &= \frac{1}{2}\mathbf{x}^T_{N-1}\Big\{Q_{N-1} + F^T_{N-1}R_{N-1}F_{N-1} + \\ &\qquad \{A_{N-1} - B_{N-1}F_{N-1}\}^T H\{A_{N-1} - B_{N-1}F_{N-1}\}\Big\}\mathbf{x}_{N-1} \\ &\equiv \frac{1}{2}\mathbf{x}^T_{N-1}P_{N-1}\mathbf{x}_{N-1}\end{aligned}$$

  - Note that $P_N = H$, which suggests a convenient form for gain F:

$$F_{N-1} = \left[R_{N-1} + B^T_{N-1}P_NB_{N-1}\right]^{-1} B^T_{N-1}P_NA_{N-1} \qquad (3.20)$$

- Now can continue using induction – assume that at time k the control will be of the form $\mathbf{u}_k^\star = -F_k \mathbf{x}_k$ where

$$F_k = \left[R_k + B_k^T P_{k+1} B_k\right]^{-1} B_k^T P_{k+1} A_k$$

and $J_k^\star[\mathbf{x}_k] = \frac{1}{2}\mathbf{x}_k^T P_k \mathbf{x}_k$ where

$$P_k = Q_k + F_k^T R_k F_k + \{A_k - B_k F_k\}^T P_{k+1} \{A_k - B_k F_k\}$$

– Recall that both equations are solved backwards from $k+1$ to k.

- Now consider time $k-1$, with

$$J_{k-1}^\star[\mathbf{x}_{k-1}] = \min_{\mathbf{u}_{k-1}} \left\{ \frac{1}{2}\mathbf{x}_{k-1}^T Q_{k-1} \mathbf{x}_{k-1} + \mathbf{u}_{k-1}^T R_{k-1} \mathbf{u}_{k-1} + J_k^\star[\mathbf{x}_k] \right\}$$

- Taking derivative with respect to $\mathbf{u}_{k-1}$ gives,

$$\frac{\partial J_{k-1}^\star[\mathbf{x}_{k-1}]}{\partial \mathbf{u}_{k-1}} = \mathbf{u}_{k-1}^T R_{k-1} + \{A_{k-1}\mathbf{x}_{k-1} + B_{k-1}\mathbf{u}_{k-1}\}^T P_k B_{k-1}$$

so that the best control input is

$$\begin{aligned} \mathbf{u}_{k-1}^\star &= -\left[R_{k-1} + B_{k-1}^T P_k B_{k-1}\right]^{-1} B_{k-1}^T P_k A_{k-1} \mathbf{x}_{k-1} \\ &= -F_{k-1}\mathbf{x}_{k-1} \end{aligned}$$

- Substitute this control into the expression for $J_{k-1}^\star[\mathbf{x}_{k-1}]$ to show that

$$J_{k-1}^\star[\mathbf{x}_{k-1}] = \frac{1}{2}\mathbf{x}_{k-1}^T P_{k-1} \mathbf{x}_{k-1}$$

and

$$\begin{aligned} P_{k-1} &= Q_{k-1} + F_{k-1}^T R_{k-1} F_{k-1} + \\ &\quad \{A_{k-1} - B_{k-1}F_{k-1}\}^T P_k \{A_{k-1} - B_{k-1}F_{k-1}\} \end{aligned}$$

- Thus the same properties hold at time $k-1$ and k, and N and $N-1$ in particular, so they will always be true.

Algorithm

- Can summarize the above in the algorithm:

$$(i) \quad P_N = H$$
$$(ii) \quad F_k = \left[R_k + B_k^T P_{k+1} B_k\right]^{-1} B_k^T P_{k+1} A_k$$
$$(iii) \quad P_k = Q_k + F_k^T R_k F_k + \{A_k - B_k F_k\}^T P_{k+1} \{A_k - B_k F_k\}$$

  cycle through steps (ii) and (iii) from $N-1 \rightarrow 0$.

- Notes:
  - The result is a control schedule that is time varying, even if A, B, Q, and R are constant.
  - Clear that P_k and F_k are independent of the state and can be computed ahead of time, off-line.
  - Possible to eliminate the F_k part of the cycle, and just cycle through P_k

$$P_k = Q_k + A_k^T \left\{ P_{k+1} - P_{k+1} B_k \left[R_k + B_k^T P_{k+1} B_k\right]^{-1} B_k^T P_{k+1} \right\} A_k$$

  - Initial assumption $R_k > 0 \; \forall \; k$ can be relaxed, but we must ensure that $\left[R_{k+1} + B_k^T Q_{k+1} B_k\right] > 0$. [2]

- In the expression:

$$J^\star(x_k^i, t_k) = \min_{u_k^{ij}} \left[g(x_k^i, u_k^{ij}, t_k)\Delta t + J^\star(x_{k+1}^j, t_{k+1}) \right]$$

  the term $J^\star(x_{k+1}^j, t_{k+1})$ plays the role of a "**cost-to-go**", which is a key concept in DP and other control problems.

[2] Anderson and Moore, *Optimal Control: Linear Quadratic Methods*, pg. 30

Suboptimal Control

- The optimal initial cost is $J_0^\star[\mathbf{x}_0] = \frac{1}{2}\mathbf{x}_0^T P_0 \mathbf{x}_0$. One question: how would the cost of a different controller strategy compare?

$$\mathbf{u}_k = -G_k \mathbf{x}_k$$

- Can substitute this controller into the cost function and compute

$$\begin{aligned} J &= \frac{1}{2}\mathbf{x}_N^T H \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}[\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k] \\ \Rightarrow\ J_G &= \frac{1}{2}\mathbf{x}_N^T H \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\mathbf{x}_k^T[Q_k + G_k^T R_k G_k]\mathbf{x}_k \end{aligned}$$

  where

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k = (A_k - B_k G_k)\mathbf{x}_k$$

- Note that:

$$\frac{1}{2}\sum_{k=0}^{N-1}\left[\mathbf{x}_{k+1}^T S_{k+1}\mathbf{x}_{k+1} - \mathbf{x}_k^T S_k \mathbf{x}_k\right] = \frac{1}{2}\left[\mathbf{x}_N^T S_N \mathbf{x}_N - \mathbf{x}_0^T S_0 \mathbf{x}_0\right]$$

- So can rearrange the cost function as

$$\begin{aligned} J_G &= \frac{1}{2}\mathbf{x}_N^T H \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left\{\mathbf{x}_k^T[Q_k + G_k^T R_k G_k - S_k]\mathbf{x}_k\right. \\ &\quad \left. + \mathbf{x}_{k+1}^T S_{k+1}\mathbf{x}_{k+1}\right\} - \frac{1}{2}\left[\mathbf{x}_N^T S_N \mathbf{x}_N - \mathbf{x}_0^T S_0 \mathbf{x}_0\right] \end{aligned}$$

  – Now substitute for $\mathbf{x}_{k+1} = (A_k - B_k G_k)\mathbf{x}_k$, and define S_k so that

$$\begin{aligned} S_N &= H \\ S_k &= Q_k + G_k^T R_k G_k + \{A_k - B_k G_k\}^T S_{k+1}\{A_k - B_k G_k\} \end{aligned}$$

  which is another recursion, that gives

$$J_G = \frac{1}{2}\mathbf{x}_0^T S_0 \mathbf{x}_0$$

- So that for a given $\mathbf{x}_0$, we can compare P_0 and S_0 to evaluate the extent to which the controller is suboptimal.

Steady State

- Assume[3]
  - Time invariant problem (LTI) – i.e., A, B, Q, R are constant
  - System $[A, B]$ stabilizable – uncontrollable modes are stable.
- For any H, then as $N \to \infty$, the recursion for P tends to a constant solution with $P_{ss} \geq 0$ that is bounded and satisfies (set $P_k \equiv P_{k+1}$)

$$P_{ss} = Q + A^T \left\{ P_{ss} - P_{ss}B \left[R + B^T P_{ss} B \right]^{-1} B^T P_{ss} \right\} A \quad (3.21)$$

  - Discrete form of the famous **Algebraic Riccati Equation**
  - Typically hard to solve analytically, but easy to solve numerically.
  - Can be many PSD solutions of (3.21), recursive solution will be one.
- Let $Q = C^T C \geq 0$, which is equivalent to having cost measurements $\mathbf{z} = C\mathbf{x}$ and state penalty $\mathbf{z}^T\mathbf{z} = \mathbf{x}^T C^T C \mathbf{x} = \mathbf{x}^T Q \mathbf{x}$. If $[A, C]$ detectable, then:
  - Independent of H, recursion for P has a **unique** steady state solution $P_{ss} \geq 0$ that is the unique PSD solution of (3.21).
  - The associated steady state gain is

$$F_{ss} = \left[R + B^T P_{ss} B \right]^{-1} B^T P_{ss} A$$

    and using F_{ss}, the closed-loop system $\mathbf{x}_{k+1} = (A - BF_{ss})\mathbf{x}_k$ is **asymptotically stable**, i.e.,

$$|\lambda(A - BF_{ss})| < 1$$

  - Detectability required to ensure that all unstable modes penalized in state cost.
- If, in addition, $[A, C]$ observable[4], then there is a unique $P_{ss} > 0$

[3] See Lewis and Syrmos, *Optimal Control*, Thm 2.4-2 and Kwakernaak and Sivan, *Linear Optimal Control Systems*, Thm 6.31

[4] Guaranteed if $Q > 0$

Discrete LQR Example

- Integrator scalar system $\dot{x} = u$, which gives

$$x_{k+1} = x_k + u_k \Delta t$$

so that $A = 1$, $B = \Delta t = 1$ and

$$J = \frac{1}{4}x(N)^2 + \frac{1}{2}\sum_{k=0}^{N-1}[x_k^2 + u_k^2]$$

so that $N = 10$, $Q = R = 1$, $H = 1/2$ (numbers in code/figures might differ)

- Note that this is a discrete system, and the rules for stability are different – need $|\lambda_i(A - BF)| < 1$.
  - Open loop system is marginally stable, and a gain $1 > F > 0$ will stabilize the system.

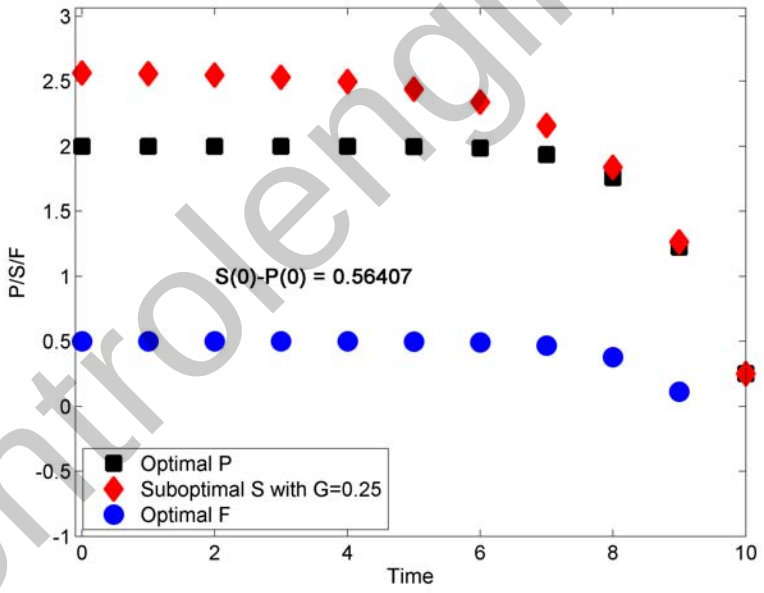


Figure 3.3: discrete LQR comparison to constant gain, $G = -0.25$

- Plot shows discrete LQR results: clear that the P_k settles to a constant value very quickly
  - Rate of reaching steady state depends on Q/R. For Q/R large reaches steady state quickly
- (Very) Suboptimal F gives an obviously worse cost

- But a reasonable choice of a constant F in this case gives nearly optimal results.

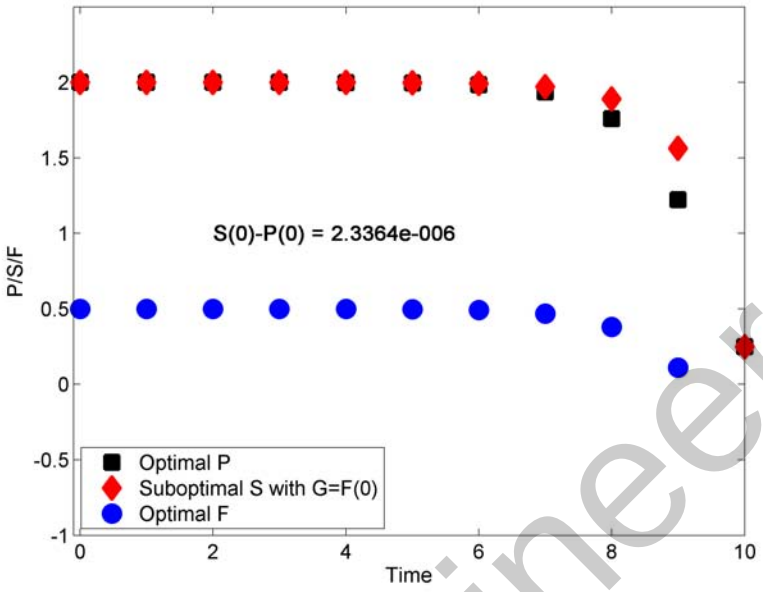


Figure 3.4: discrete LQR comparison to constant gain, $G = F(0)$

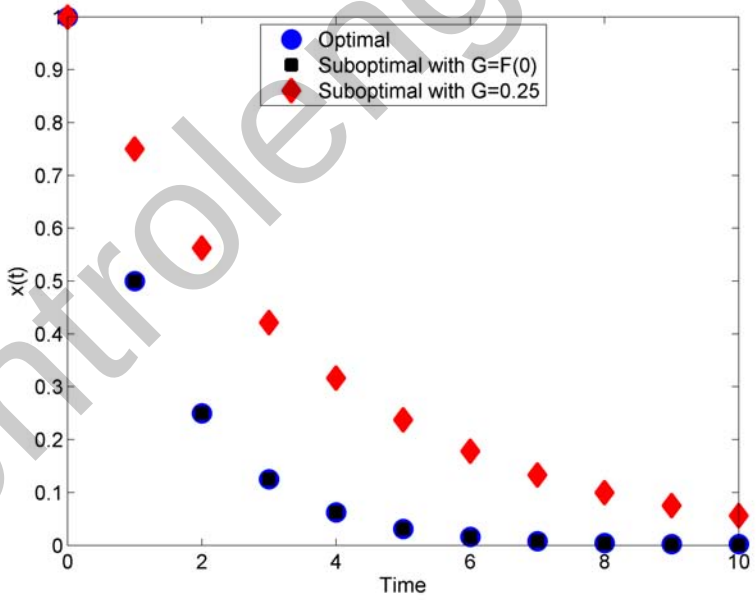


Figure 3.5: State response comparison

- State response consistent

Gain Insights

- Note that from Eq. 3.20 we know that

$$F_{N-1} = \left[R_{N-1} + B_{N-1}^T P_N B_{N-1}\right]^{-1} B_{N-1}^T P_N A_{N-1}$$

which for the scalar case reduces to

$$F_{N-1} = \frac{B_{N-1} P_N A_{N-1}}{R_{N-1} + B_{N-1}^2 P_N}$$

- So if there is a high weighting on the terminal state, then $H \to \infty$ and P_N is large. Thus

$$F_{N-1} \to \frac{B_{N-1} P_N A_{N-1}}{B_{N-1}^2 P_N} \to \frac{A}{B}$$

and

$$x_N = (A - BF)x_{N-1} = (A - B\frac{A}{B})x_{N-1} = 0$$

regardless of the value of x_{N-1}. This is a **nilpotent** controller.

- If control penalty set very small, so that $R \to 0$ (Q/R large), then

$$F_{N-1} \to \frac{B_{N-1} P_N A_{N-1}}{B_{N-1}^2 P_N} \to \frac{A}{B}$$

and $x_N = 0$ as well.

  - State penalized, but control isn't, so controller will exert as much effort as necessary to make x small.
  - In fact, this will typically make $x(1) = 0$ regardless of $x(0)$ if there are no limits on the control effort.

Discrete scalar LQR

```
% 16.323 Spring 2008
% Jonathan How
% integ.m: integrator system
%
clear all
close all
%A=1;B=1;Q=1;R=1;H=0.5;N=5;
A=1;B=1;Q=1;R=2;H=.25;N=10;

P(N+1)=H; % shift indices to avoid index of 0
for j=N-1:-1:0
    i=j+1; % shift indices to avoid index of 0
    F(i)=inv(R+B'*P(i+1)*B)*B'*P(i+1)*A;
    P(i)=(A-B*F(i))'*P(i+1)*(A-B*F(i))+F(i)'*R*F(i)+Q;
end

% what if we used a fixed gain of F(0), which stabilizes
S(N+1)=H; % shift indices to avoid index of 0
for j=N-1:-1:0
    i=j+1; % shift indices to avoid index of 0
    G(i)=F(1);
    S(i)=(A-B*G(i))'*S(i+1)*(A-B*G(i))+G(i)'*R*G(i)+Q;
end

time=[0:1:N];
figure(1);clf
plot(time,P,'ks','MarkerSize',12,'MarkerFaceColor','k')
hold on
plot(time,S,'rd','MarkerSize',12,'MarkerFaceColor','r')
plot(time(1:N),F,'bo','MarkerSize',12,'MarkerFaceColor','b')
hold off
legend('Optimal P','Suboptimal S with G=F(0)','Optimal F','Location','SouthWest')
xlabel('Time')
ylabel('P/S/F')
text(2,1,['S(0)-P(0) = ',num2str(S(1)-P(1))])
axis([-.1 N -1 max(max(P),max(S))+.5] )
print -dpng -r300 integ.png

% what if we used a fixed gain of G=0.25, which stabilizes
S(N+1)=H; % shift indices to avoid index of 0
for j=N-1:-1:0
    i=j+1; % shift indices to avoid index of 0
    G(i)=.25;
    S(i)=(A-B*G(i))'*S(i+1)*(A-B*G(i))+G(i)'*R*G(i)+Q;
end

figure(2)
%plot(time,P,'ks',time,S,'rd',time(1:N),F,'bo','MarkerSize',12)
plot(time,P,'ks','MarkerSize',12,'MarkerFaceColor','k')
hold on
plot(time,S,'rd','MarkerSize',12,'MarkerFaceColor','r')
plot(time(1:N),F,'bo','MarkerSize',12,'MarkerFaceColor','b')
hold off
legend('Optimal P','Suboptimal S with G=0.25','Optimal F','Location','SouthWest')
text(2,1,['S(0)-P(0) = ',num2str(S(1)-P(1))])
axis([-.1 N -1 max(max(P),max(S))+.5] )
ylabel('P/S/F')
xlabel('Time')
print -dpng -r300 integ2

% state response
x0=1;xo=x0;xs1=x0;xs2=x0;
for j=0:N-1;
    k=j+1;
    xo(k+1)=(A-B*F(k))*xo(k);
    xs1(k+1)=(A-B*F(1))*xs1(k);
```

```
67      xs2(k+1)=(A-B*G(1))*xs2(k);
68  end
69  figure(3)
70  plot(time,xo,'bo','MarkerSize',12,'MarkerFaceColor','b')
71  hold on
72  plot(time,xs1,'ks','MarkerSize',9,'MarkerFaceColor','k')
73  plot(time,xs2,'rd','MarkerSize',12,'MarkerFaceColor','r')
74  hold off
75  legend('Optimal','Suboptimal with G=F(0)','Suboptimal with G=0.25','Location','North')
76  %axis([-.1 5 -1 3] )
77  ylabel('x(t)')
78  xlabel('Time')
79  print -dpng -r300 integ3.png;
80
```

Appendix

- **Def:** LTI **system** is **controllable** if, for every $\mathbf{x}^\star(t)$ and every finite $T > 0$, there exists an input function $\mathbf{u}(t)$, $0 < t \leq T$, such that the system state goes from $\mathbf{x}(0) = 0$ to $\mathbf{x}(T) = \mathbf{x}^\star$.
  - Starting at 0 is not a special case – if we can get to any state in finite time from the origin, then we can get from any initial condition to that state in finite time as well. [5]

- **Thm:** LTI system is controllable iff it has no uncontrollable states.
  - Necessary and sufficient condition for controllability is that

$$\texttt{rank}\ \mathcal{M}_c \triangleq \texttt{rank}\ \left[\begin{array}{cccc} B & AB & A^2B & \cdots & A^{n-1}B \end{array}\right] = n$$

- **Def:** LTI **system** is **observable** if the initial state $\mathbf{x}(0)$ can be uniquely deduced from the knowledge of the input $\mathbf{u}(t)$ and output $\mathbf{y}(t)$ for all t between 0 and any finite $T > 0$.
  - If $\mathbf{x}(0)$ can be deduced, then we can reconstruct $\mathbf{x}(t)$ exactly because we know $\mathbf{u}(t) \Rightarrow$ we can find $\mathbf{x}(t)\ \forall\ t$.

- **Thm:** LTI system is observable iff it has no unobservable states.
  - We normally just say that the **pair (A,C) is observable.**
  - Necessary and sufficient condition for observability is that

$$\texttt{rank}\ \mathcal{M}_o \triangleq \texttt{rank}\ \left[\begin{array}{c} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{array}\right] = n$$

[5] This controllability from the origin is often called **reachability**.

16.323 Lecture 4

HJB Equation

- DP in continuous time
- HJB Equation
- Continuous LQR

Factoids: for symmetric R

$$\frac{\partial \mathbf{u}^T R \mathbf{u}}{\partial \mathbf{u}} = 2\mathbf{u}^T R$$

$$\frac{\partial R\mathbf{u}}{\partial \mathbf{u}} = R$$

DP in Continuous Time

- Have analyzed a couple of approximate solutions to the classic control problem of minimizing:

$$\min J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\, dt$$

subject to

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \\ \mathbf{x}(t_0) &= \text{given} \\ \mathbf{m}(\mathbf{x}(t_f), t_f) &= 0 \text{ set of terminal conditions} \\ \mathbf{u}(t) &\in \mathcal{U} \text{ set of possible constraints} \end{aligned}$$

- Previous approaches discretized in time, state, and control actions
  - Useful for implementation on a computer, but now want to consider the exact solution in continuous time
  - Result will be a nonlinear partial differential equation called the **Hamilton-Jacobi-Bellman** equation (**HJB**) – a key result.

- First step: consider cost over the interval $[t, t_f]$, where $t \le t_f$ of any control sequence $\mathbf{u}(\tau)$, $t \le \tau \le t_f$

$$J(\mathbf{x}(t), t, \mathbf{u}(\tau)) = h(\mathbf{x}(t_f), t_f) + \int_{t}^{t_f} g(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau)\, d\tau$$

  - Clearly the goal is to pick $\mathbf{u}(\tau)$, $t \le \tau \le t_f$ to minimize this cost.

$$J^\star(\mathbf{x}(t), t) = \min_{\substack{\mathbf{u}(\tau) \in \mathcal{U} \\ t \le \tau \le t_f}} J(\mathbf{x}(t), t, \mathbf{u}(\tau))$$

- Approach:
  - Split time interval $[t, t_f]$ into $[t, t+\Delta t]$ and $[t+\Delta t, t_f]$, and are specifically interested in the case where $\Delta t \to 0$
  - Identify the optimal cost-to-go $J^\star(\mathbf{x}(t+\Delta t), t+\Delta t)$
  - Determine the "stage cost" in time $[t, t+\Delta t]$
  - Combine above to find best strategy from time t.
  - Manipulate result into HJB equation.

- Split:

$$J^\star(\mathbf{x}(t), t) = \min_{\substack{\mathbf{u}(\tau)\in\mathcal{U} \\ t\le\tau\le t_f}} \left\{ h(\mathbf{x}(t_f), t_f) + \int_t^{t_f} g(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau))\, d\tau \right\}$$

$$= \min_{\substack{\mathbf{u}(\tau)\in\mathcal{U} \\ t\le\tau\le t_f}} \left\{ h(\mathbf{x}(t_f), t_f) + \int_t^{t+\Delta t} g(\mathbf{x}, \mathbf{u}, \tau)\, d\tau + \int_{t+\Delta t}^{t_f} g(\mathbf{x}, \mathbf{u}, \tau)\, d\tau \right\}$$

- Implicit here that at time $t+\Delta t$, the system will be at state $\mathbf{x}(t+\Delta t)$.
  - But from the **principle of optimality**, we can write that the optimal cost-to-go from this state is:

$$J^\star(\mathbf{x}(t+\Delta t), t+\Delta t)$$

- Thus can rewrite the cost calculation as:

$$J^\star(\mathbf{x}(t), t) = \min_{\substack{\mathbf{u}(\tau)\in\mathcal{U} \\ t\le\tau\le t+\Delta t}} \left\{ \int_t^{t+\Delta t} g(\mathbf{x}, \mathbf{u}, \tau)\, d\tau + J^\star(\mathbf{x}(t+\Delta t), t+\Delta t) \right\}$$

- Assuming $J^\star(\mathbf{x}(t+\Delta t), t+\Delta t)$ has bounded second derivatives in both arguments, can expand this cost as a Taylor series about $\mathbf{x}(t), t$

$$J^\star(\mathbf{x}(t+\Delta t), t+\Delta t) \approx J^\star(\mathbf{x}(t), t) + \left[\frac{\partial J^\star}{\partial t}(\mathbf{x}(t), t)\right] \Delta t + \left[\frac{\partial J^\star}{\partial \mathbf{x}}(\mathbf{x}(t), t)\right] (\mathbf{x}(t+\Delta t) - \mathbf{x}(t))$$

  - Which for small Δt can be compactly written as:

$$J^\star(\mathbf{x}(t+\Delta t), t+\Delta t) \approx J^\star(\mathbf{x}(t), t) + J_t^\star(\mathbf{x}(t), t)\Delta t + J_{\mathbf{x}}^\star(\mathbf{x}(t), t)\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)\Delta t$$

- Substitute this into the cost calculation with a small Δt to get

$$J^\star(\mathbf{x}(t), t) = \min_{\mathbf{u}(t)\in\mathcal{U}} \{g(\mathbf{x}(t), \mathbf{u}(t), t)\Delta t + J^\star(\mathbf{x}(t), t) + J_t^\star(\mathbf{x}(t), t)\Delta t + J_{\mathbf{x}}^\star(\mathbf{x}(t), t)\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)\Delta t\}$$

- Extract the terms that are independent of $\mathbf{u}(t)$ and cancel

$$0 = J_t^\star(\mathbf{x}(t), t) + \min_{\mathbf{u}(t)\in\mathcal{U}} \{g(\mathbf{x}(t), \mathbf{u}(t), t) + J_{\mathbf{x}}^\star(\mathbf{x}(t), t)\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)\}$$

  - This is a partial differential equation in $J^\star(\mathbf{x}(t), t)$ that is solved backwards in time with an initial condition that

$$J^\star(\mathbf{x}(t_f), t_f) = h(\mathbf{x}(t_f))$$

  for $\mathbf{x}(t_f)$ and t_f combinations that satisfy $m(\mathbf{x}(t_f), t_f) = 0$

HJB Equation

- For simplicity, define the **Hamiltonian**

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, J^\star_{\mathbf{x}}, t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + J^\star_{\mathbf{x}}(\mathbf{x}(t), t)\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)$$

  then the **HJB equation** is

$$-J^\star_t(\mathbf{x}(t), t) = \min_{\mathbf{u}(t)\in\mathcal{U}} \mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), J^\star_{\mathbf{x}}(\mathbf{x}(t), t), t)$$

  - A very powerful result, that is both a **necessary and sufficient** condition for optimality
  - But one that is hard to solve/use in general.

- Some references on numerical solution methods:
  - M. G. Crandall, L. C. Evans, and P.-L. Lions, "Some properties of viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 282, no. 2, pp. 487–502, 1984.
  - M. Bardi and I. Capuzzo-Dolcetta (1997), "Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations," *Systems & Control: Foundations & Applications,* Birkhauser, Boston.
- Can use it to directly solve the continuous LQR problem

HJB Simple Example

- Consider the system with dynamics

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{u}$$

for which $A + A^T = 0$ and $\|\mathbf{u}\| \leq 1$, and the cost function

$$J = \int_0^{t_f} dt = t_f$$

- Then the Hamiltonian is

$$\mathcal{H} = 1 + J_{\mathbf{x}}^\star(A\mathbf{x} + \mathbf{u})$$

and the constrained minimization of $\mathcal{H}$ with respect to $\mathbf{u}$ gives

$$\mathbf{u}^\star = -(J_{\mathbf{x}}^\star)^T / \|J_{\mathbf{x}}^\star\|$$

- Thus the HJB equation is:

$$-J_t^\star = 1 + J_{\mathbf{x}}^\star(A\mathbf{x}) - \|J_{\mathbf{x}}^\star\|$$

- As a candidate solution, take $J^\star(\mathbf{x}) = \mathbf{x}^T\mathbf{x}/\|\mathbf{x}\| = \|\mathbf{x}\|$, which is not an explicit function of t, so

$$J_{\mathbf{x}}^\star = \frac{\mathbf{x}^T}{\|\mathbf{x}\|} \quad \text{and} \quad J_t^\star = 0$$

which gives:

$$\begin{aligned} 0 &= 1 + \frac{\mathbf{x}^T}{\|\mathbf{x}\|}(A\mathbf{x}) - \frac{\|\mathbf{x}\|}{\|\mathbf{x}\|} \\ &= \frac{1}{\|\mathbf{x}\|}(\mathbf{x}^T A\mathbf{x}) \\ &= \frac{1}{\|\mathbf{x}\|}\frac{1}{2}\mathbf{x}^T(A + A^T)\mathbf{x} = 0 \end{aligned}$$

so that the HJB is satisfied and the optimal control is:

$$\mathbf{u}^\star = -\frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Continuous LQR

- Specialize to a linear system model and a quadratic cost function

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$

$$J = \frac{1}{2}\mathbf{x}(t_f)^T H \mathbf{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f} \left\{\mathbf{x}(t)^T R_{\rm xx}(t)\mathbf{x}(t) + \mathbf{u}(t)^T R_{\rm uu}(t)\mathbf{u}(t)\right\} dt$$

  - Assume that t_f fixed and there are no bounds on $\mathbf{u}$,
  - Assume $H, R_{\rm xx}(t) \geq 0$ and $R_{\rm uu}(t) > 0$, then

$$\begin{aligned}\mathcal{H}(\mathbf{x}, \mathbf{u}, J^\star_{\mathbf{x}}, t) \;=\; & \frac{1}{2}\left[\mathbf{x}(t)^T R_{\rm xx}(t)\mathbf{x}(t) + \mathbf{u}(t)^T R_{\rm uu}(t)\mathbf{u}(t)\right] \\ & + J^\star_{\mathbf{x}}(\mathbf{x}(t), t)\left[A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)\right]\end{aligned}$$

- Now need to find the minimum of $\mathcal{H}$ with respect to $\mathbf{u}$, which will occur at a stationary point that we can find using (no constraints)

$$\frac{\partial \mathcal{H}}{\partial \mathbf{u}} = \mathbf{u}(t)^T R_{\rm uu}(t) + J^\star_{\mathbf{x}}(\mathbf{x}(t), t)B(t) = 0$$

  - Which gives the **optimal control law:**

$$\mathbf{u}^\star(t) = -R^{-1}_{\rm uu}(t)B(t)^T J^\star_{\mathbf{x}}(\mathbf{x}(t), t)^T$$

  - Since

$$\frac{\partial^2 \mathcal{H}}{\partial \mathbf{u}^2} = R_{\rm uu}(t) > 0$$

  then this defines a global minimum.

- Given this control law, can rewrite the Hamiltonian as:

$$\mathcal{H}(\mathbf{x}, \mathbf{u}^\star, J^\star_{\mathbf{x}}, t) =$$

$$\frac{1}{2}\left[\mathbf{x}(t)^T R_{\rm xx}(t)\mathbf{x}(t) + J^\star_{\mathbf{x}}(\mathbf{x}(t),t)B(t)R^{-1}_{\rm uu}(t)R_{\rm uu}(t)R^{-1}_{\rm uu}(t)B(t)^T J^\star_{\mathbf{x}}(\mathbf{x}(t),t)^T\right]$$
$$+J^\star_{\mathbf{x}}(\mathbf{x}(t),t)\left[A(t)\mathbf{x}(t) - B(t)R^{-1}_{\rm uu}(t)B(t)^T J^\star_{\mathbf{x}}(\mathbf{x}(t),t)^T\right]$$

$$= \frac{1}{2}\mathbf{x}(t)^T R_{\rm xx}(t)\mathbf{x}(t) + J^\star_{\mathbf{x}}(\mathbf{x}(t),t)A(t)\mathbf{x}(t)$$
$$-\frac{1}{2}J^\star_{\mathbf{x}}(\mathbf{x}(t),t)B(t)R^{-1}_{\rm uu}(t)B(t)^T J^\star_{\mathbf{x}}(\mathbf{x}(t),t)^T$$

- Might be difficult to see where this is heading, but note that the boundary condition for this PDE is:

$$J^\star(\mathbf{x}(t_f), t_f) = \frac{1}{2}\mathbf{x}^T(t_f)H\mathbf{x}(t_f)$$

  - So a candidate solution to investigate is to maintain a quadratic form for this cost for all time t. So could assume that

$$J^\star(\mathbf{x}(t),t) = \frac{1}{2}\mathbf{x}^T(t)P(t)\mathbf{x}(t), \qquad P(t) = P^T(t)$$

    and see what conditions we must impose on $P(t)$. [6]

  - Note that in this case, $J^\star$ is a function of the variables $\mathbf{x}$ and t[7]

$$\frac{\partial J^\star}{\partial \mathbf{x}} = \mathbf{x}^T(t)P(t)$$

$$\frac{\partial J^\star}{\partial t} = \frac{1}{2}\mathbf{x}^T(t)\dot{P}(t)\mathbf{x}(t)$$

- To use HJB equation need to evaluate:

$$-J^\star_t(\mathbf{x}(t),t) = \min_{\mathbf{u}(t)\in\mathcal{U}} \mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), J^\star_{\mathbf{x}}, t)$$

[6] See AM, pg. 21 on how to avoid having to make this assumption.

[7] Partial derivatives taken wrt one variable assuming the other is fixed. Note that there are 2 independent variables in this problem x and t. x is time-varying, but it is not a function of t.

- Substitute candidate solution into HJB:

$$\begin{aligned}-\frac{1}{2}\mathbf{x}(t)^T\dot{P}(t)\mathbf{x}(t) &= \frac{1}{2}\mathbf{x}(t)^T R_{\rm xx}(t)\mathbf{x}(t) + \mathbf{x}^T P(t)A(t)\mathbf{x}(t) \\ &\quad -\frac{1}{2}\mathbf{x}^T(t)P(t)B(t)R_{\rm uu}^{-1}(t)B(t)^T P(t)\mathbf{x}(t) \\ &= \frac{1}{2}\mathbf{x}(t)^T R_{\rm xx}(t)\mathbf{x}(t) + \frac{1}{2}\mathbf{x}^T(t)\{P(t)A(t) + A(t)^T P(t)\}\mathbf{x}(t) \\ &\quad -\frac{1}{2}\mathbf{x}^T(t)P(t)B(t)R_{\rm uu}^{-1}(t)B(t)^T P(t)\mathbf{x}(t)\end{aligned}$$

  which must be true for all $\mathbf{x}(t)$, so we require that $P(t)$ solve

$$\boxed{\begin{aligned}-\dot{P}(t) &= P(t)A(t) + A(t)^T P(t) + R_{\rm xx}(t) - P(t)B(t)R_{\rm uu}^{-1}(t)B(t)^T P(t) \\ P(t_f) &= H\end{aligned}}$$

- If $P(t)$ solves this **Differential Riccati Equation**, then the HJB equation is satisfied by the candidate $J^\star(\mathbf{x}(t),t)$ and the resulting control is **optimal**.

- Key thing about this $J^\star$ solution is that, since $J^\star_{\mathbf{x}} = \mathbf{x}^T(t)P(t)$, then

$$\begin{aligned}\mathbf{u}^\star(t) &= -R_{\rm uu}^{-1}(t)B(t)^T J^\star_{\mathbf{x}}(\mathbf{x}(t),t)^T \\ &= -R_{\rm uu}^{-1}(t)B(t)^T P(t)\mathbf{x}(t)\end{aligned}$$

  - Thus **optimal feedback control is a linear state feedback** with gain

$$F(t) = R_{\rm uu}^{-1}(t)B(t)^T P(t) \Rightarrow \mathbf{u}(t) = -F(t)\mathbf{x}(t)$$

    ◇ Can be solved for ahead of time.

- As before, can evaluate the performance of some arbitrary time-varying feedback gain $\mathbf{u} = -G(t)\mathbf{x}(t)$, and the result is that

$$J_G = \frac{1}{2}\mathbf{x}^T S(t)\mathbf{x}$$

where $S(t)$ solves

$$\begin{aligned} -\dot{S}(t) &= \{A(t) - B(t)G(t)\}^T S(t) + S(t)\{A(t) - B(t)G(t)\} \\ &\quad + R_{\rm xx}(t) + G(t)^T R_{\rm uu}(t) G(t) \\ S(t_f) &= H \end{aligned}$$

  - Since this must be true for arbitrary G, then would expect that this reduces to Riccati Equation if $G(t) \equiv R_{\rm uu}^{-1}(t)B^T(t)S(t)$

- If we assume LTI dynamics and let $t_f \to \infty$, then at any finite time t, would expect the Differential Riccati Equation to settle down to a steady state value (if it exists) which is the solution of

$$PA + A^T P + R_{\rm xx} - PBR_{\rm uu}^{-1}B^T P = 0$$

  - Called the **(Control) Algebraic Riccati Equation (CARE)**
  - Typically assume that $R_{\rm xx} = C_z^T R_{\rm zz} C_z$, $R_{\rm zz} > 0$ associated with performance output variable $\mathbf{z}(t) = C_z\mathbf{x}(t)$

LQR Observations

- With terminal penalty, $H = 0$, the solution to the differential Riccati Equation (DRE) approaches a constant iff the system has no poles that are unstable, uncontrollable[8], and observable[9] by $\mathbf{z}(t)$
  - If a constant steady state solution to the DRE exists, then it is a positive semi-definite, symmetric solution of the CARE.

- If $[A, B, C_z]$ is both stabilizable and detectable (i.e. all modes are stable or seen in the cost function), then:
  - Independent of $H \geq 0$, the steady state solution P_{ss} of the DRE approaches the **unique** PSD symmetric solution of the CARE.

- If a steady state solution exists P_{ss} to the DRE, then the closed-loop system using the static form of the feedback

$$\mathbf{u}(t) = -R_{\text{uu}}^{-1} B^T P_{ss} \mathbf{x}(t) = -F_{ss} \mathbf{x}(t)$$

  is **asymptotically stable** if and only if the system $[A, B, C_z]$ is stabilizable and detectable.
  - This steady state control minimizes the infinite horizon cost function $\lim_{t_f \to \infty} J$ for all $H \geq 0$

- The solution P_{ss} is **positive definite** if and only if the system $[A, B, C_z]$ is stabilizable and completely observable.

- See Kwakernaak and Sivan, page 237, Section 3.4.3.

[8] 16.31 Notes on Controllability

[9] 16.31 Notes on Observability

Scalar LQR Example

- A scalar system with dynamics $\dot{x} = ax + bu$ and with cost ($R_{\rm xx} > 0$ and $R_{\rm uu} > 0$)

$$J = \int_0^\infty (R_{\rm xx} x^2(t) + R_{\rm uu} u^2(t))\, dt$$

- This simple system represents one of the few cases for which the differential Riccati equation can be solved analytically:

$$P(\tau) = \frac{(aP_{t_f} + R_{\rm xx})\sinh(\beta\tau) + \beta P_{t_f}\cosh(\beta\tau)}{(b^2 P_{t_f}/R_{\rm uu} - a)\sinh(\beta\tau) + \beta\cosh(\beta\tau)}$$

where $\tau = t_f - t$, $\beta = \sqrt{a^2 + b^2(R_{\rm xx}/R_{\rm uu})}$.

  - Note that for given a and b, ratio $R_{\rm xx}/R_{\rm uu}$ determines the time constant of the transient in $P(t)$ (determined by β).

- The steady-state P solves the CARE:

$$2aP_{ss} + R_{\rm xx} - P_{ss}^2 b^2/R_{\rm uu} = 0$$

which gives (take positive one) that

$$P_{ss} = \frac{a + \sqrt{a^2 + b^2 R_{\rm xx}/R_{\rm uu}}}{b^2/R_{\rm uu}} = \frac{a+\beta}{b^2/R_{\rm uu}} = \frac{a+\beta}{b^2/R_{\rm uu}}\left(\frac{-a+\beta}{-a+\beta}\right) > 0$$

- With $P_{t_f} = 0$, the solution of the differential equation reduces to:

$$P(\tau) = \frac{R_{\rm xx}\sinh(\beta\tau)}{(-a)\sinh(\beta\tau) + \beta\cosh(\beta\tau)}$$

where as $\tau \to t_f (\to \infty)$ then $\sinh(\beta\tau) \to \cosh(\beta\tau) \to e^{\beta\tau}/2$, so

$$P(\tau) = \frac{R_{\rm xx}\sinh(\beta\tau)}{(-a)\sinh(\beta\tau) + \beta\cosh(\beta\tau)} \to \frac{R_{\rm xx}}{(-a)+\beta} = P_{ss}$$

- Then the steady state feedback controller is $u(t) = -Kx(t)$ where

$$K_{ss} = R_{\rm uu}^{-1} b P_{ss} = \frac{a + \sqrt{a^2 + b^2 R_{\rm xx}/R_{\rm uu}}}{b}$$

- The closed-loop dynamics are

$$\begin{aligned} \dot{x} &= (a - bK_{ss})x = A_{cl}x(t) \\ &= \left(a - \frac{b}{b}(a + \sqrt{a^2 + b^2 R_{\rm xx}/R_{\rm uu}}) \right) x \\ &= -\sqrt{a^2 + b^2 R_{\rm xx}/R_{\rm uu}}\; x \end{aligned}$$

  which are clearly stable.

- As $R_{\rm xx}/R_{\rm uu} \to \infty$, $A_{cl} \approx -|b|\sqrt{R_{\rm xx}/R_{\rm uu}}$
  - **Cheap control** problem
  - Note that smaller $R_{\rm uu}$ leads to much faster response.

- As $R_{\rm xx}/R_{\rm uu} \to 0$, $K \approx (a + |a|)/b$
  - **Expensive control** problem
  - If $a < 0$ (open-loop stable), $K \approx 0$ and $A_{cl} = a - bK \approx a$
  - If $a > 0$ (OL unstable), $K \approx 2a/b$ and $A_{cl} = a - bK \approx -a$

- Note that in the expensive control case, the controller tries to do as little as possible, but it must stabilize the unstable open-loop system.
  - Observation: optimal definition of "as little as possible" is to put the closed-loop pole at the reflection of the open-loop pole about the imaginary axis.

Numerical P Integration

- To numerically integrate solution of P, note that we can use standard Matlab integration tools if we can rewrite the DRE in vector form.
  - Define a vec operator so that

$$\mathtt{vec}(P) = \begin{bmatrix} P_{11} \\ P_{12} \\ \vdots \\ P_{1n} \\ P_{22} \\ P_{23} \\ \vdots \\ P_{nn} \end{bmatrix} \equiv y$$

  - The $\mathtt{unvec}(y)$ operation is the straightforward
  - Can now write the DRE as differential equation in the variable y

- Note that with $\tau = t_f - t$, then $d\tau = -dt$,
  - $t = t_f$ corresponds to $\tau = 0$, $t = 0$ corresponds to $\tau = t_f$
  - Can do the integration forward in time variable $\tau : 0 \rightarrow t_f$

- Then define a Matlab function as

```
doty = function(y);
global A B Rxx Ruu %
P=unvec(y); %
% assumes that P derivative wrt to tau (so no negative)
dot P = (P*A + A^T*P+Rxx-P*B*Ruu^{-1}*B^T*P);%
doty = vec(dotP); %
return
```

  - Which is integrated from $\tau = 0$ with initial condition H
  - Code uses a more crude form of integration

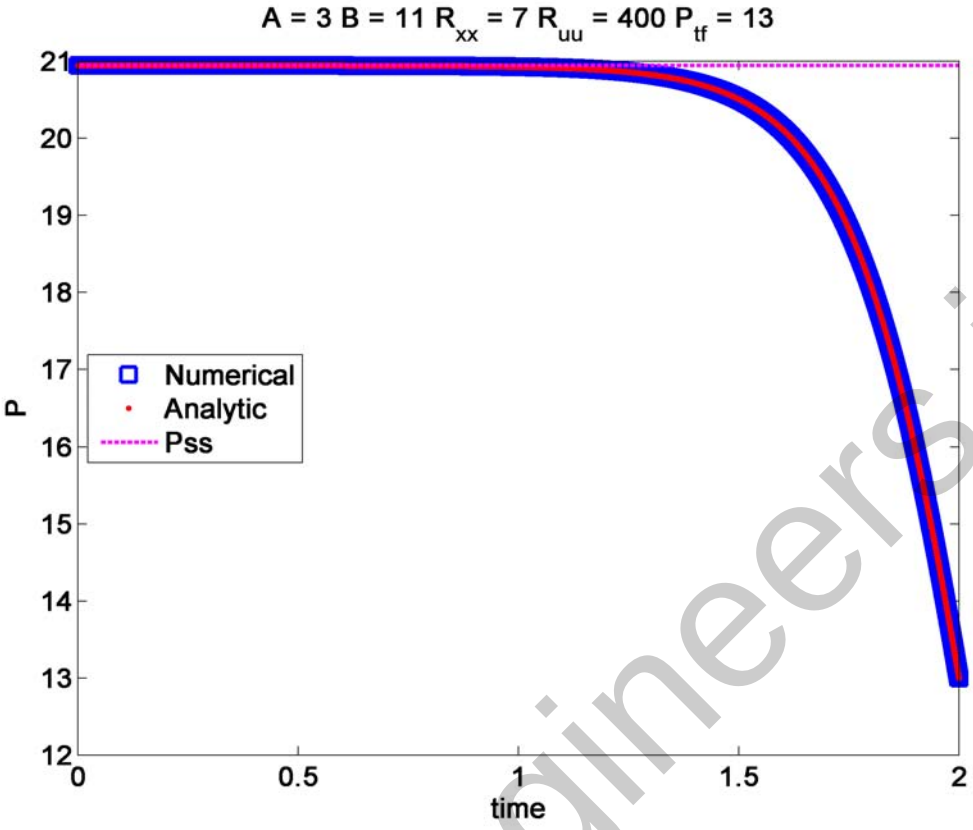


Figure 4.1: Comparison of numerical and analytical P

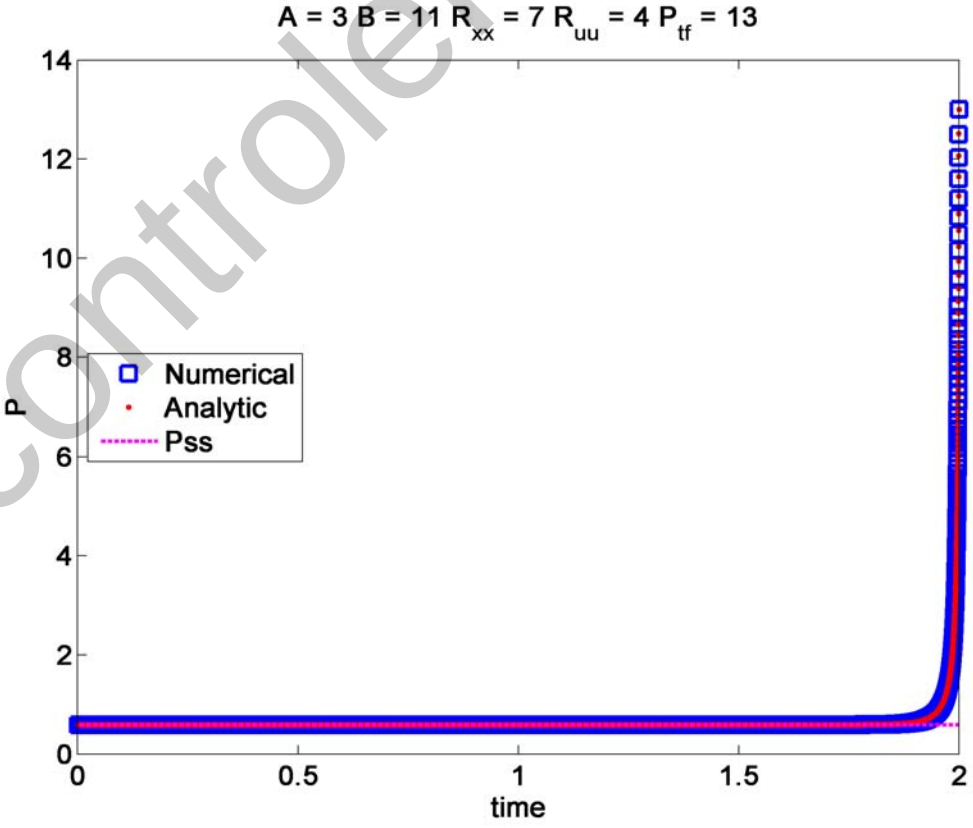


Figure 4.2: Comparison showing response with much larger R_{xx}/R_{uu}

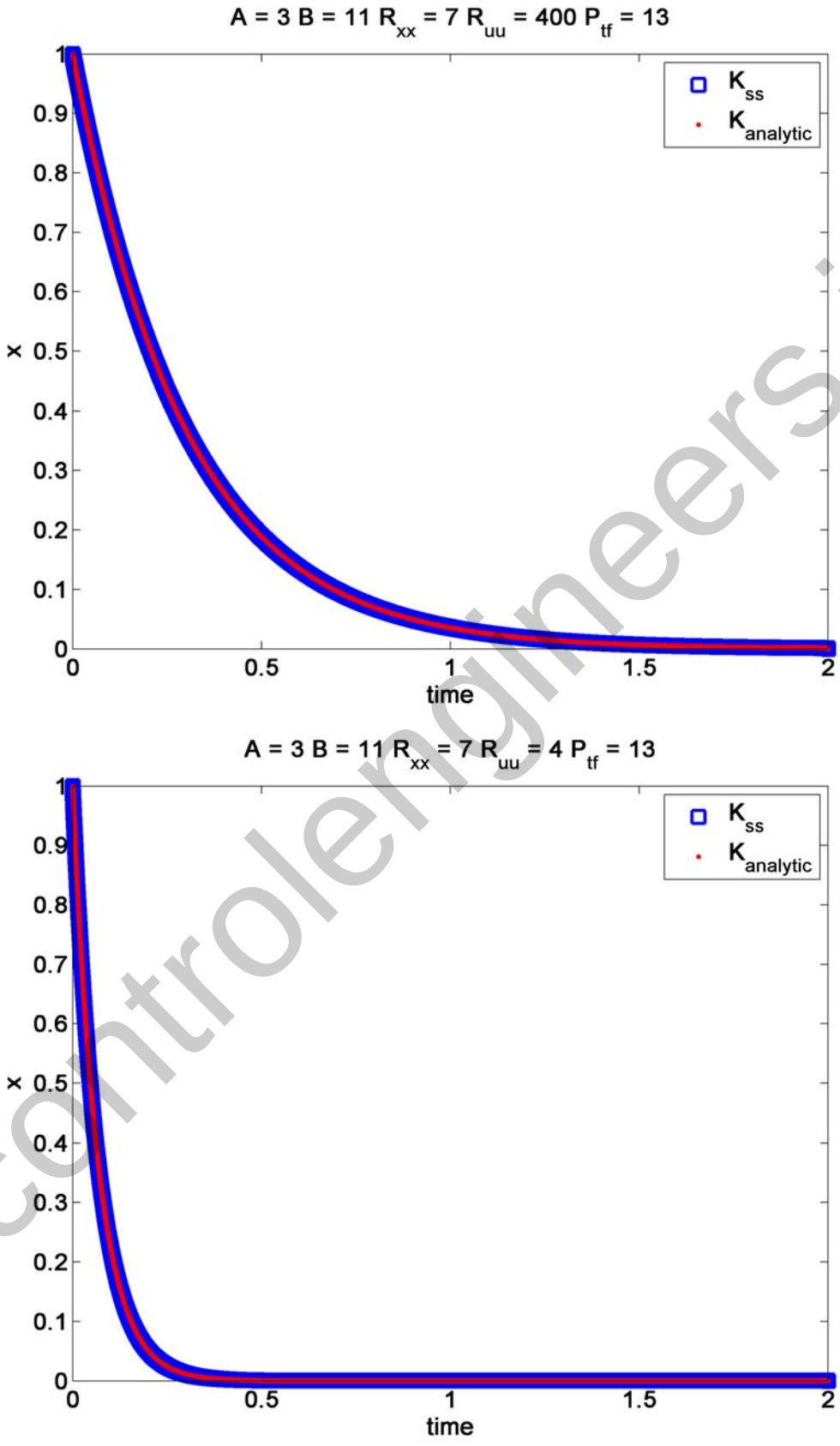


Figure 4.3: State response with high and low R_{uu}. State response with time-varying gain almost indistinguishable – highly dynamic part of x response ends before significant variation in P.

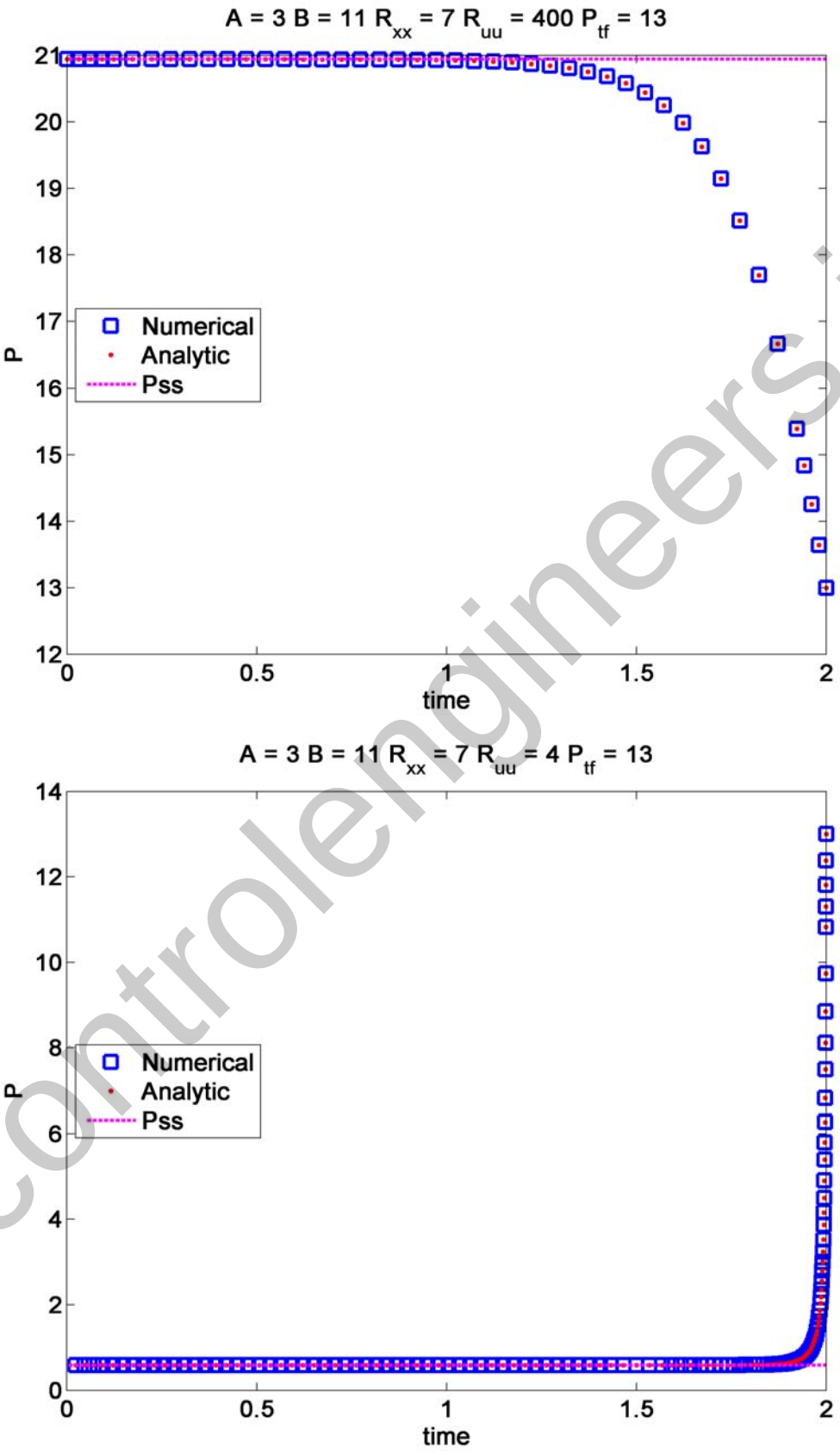


Figure 4.4: Comparison of numerical and analytical P using a better integration scheme

Numerical Calculation of P

```
% Simple LQR example showing time varying P and gains
% 16.323 Spring 2008
% Jonathan How
% reg2.m
clear all;close all;
set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontWeight','demi')
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontWeight','demi')
global A B Rxx Ruu

A=3;B=11;Rxx=7;Ptf=13;tf=2;dt=.0001;
Ruu=20^2;
Ruu=2^2;

% integrate the P backwards (crude form)
time=[0:dt:tf];
P=zeros(1,length(time));K=zeros(1,length(time));Pcurr=Ptf;
for kk=0:length(time)-1
  P(length(time)-kk)=Pcurr;
  K(length(time)-kk)=inv(Ruu)*B'*Pcurr;
  Pdot=-Pcurr*A-A'*Pcurr-Rxx+Pcurr*B*inv(Ruu)*B'*Pcurr;
  Pcurr=Pcurr-dt*Pdot;
end

options=odeset('RelTol',1e-6,'AbsTol',1e-6)
[tau,y]=ode45(@doty,[0 tf],vec(Ptf));
Tnum=[];Pnum=[];Fnum=[];
for i=1:length(tau)
    Tnum(length(tau)-i+1)=tf-tau(i);
    temp=unvec(y(i,:));
    Pnum(length(tau)-i+1,:,:)=temp;
    Fnum(length(tau)-i+1,:)=-inv(Ruu)*B'*temp;
end

% get the SS result from LQR
[klqr,Plqr]=lqr(A,B,Rxx,Ruu);

% Analytical pred
beta=sqrt(A^2+Rxx/Ruu*B^2);
t=tf-time;
Pan=((A*Ptf+Rxx)*sinh(beta*t)+beta*Ptf*cosh(beta*t))./...
    ((B^2*Ptf/Ruu-A)*sinh(beta*t)+beta*cosh(beta*t));
Pan2=((A*Ptf+Rxx)*sinh(beta*(tf-Tnum))+beta*Ptf*cosh(beta*(tf-Tnum)))./...
    ((B^2*Ptf/Ruu-A)*sinh(beta*(tf-Tnum))+beta*cosh(beta*(tf-Tnum)));

figure(1);clf
plot(time,P,'bs',time,Pan,'r.',[0 tf],[1 1]*Plqr,'m--')
title(['A = ',num2str(A),' B = ',num2str(B),' R_{xx} = ',num2str(Rxx),...
    ' R_{uu} = ',num2str(Ruu),' P_{tf} = ',num2str(Ptf)])
legend('Numerical','Analytic','Pss','Location','West')
xlabel('time');ylabel('P')
if Ruu > 10
    print -r300 -dpng reg2_1.png;
else
    print -r300 -dpng reg2_2.png;
end

figure(3);clf
plot(Tnum,Pnum,'bs',Tnum,Pan2,'r.',[0 tf],[1 1]*Plqr,'m--')
title(['A = ',num2str(A),' B = ',num2str(B),' R_{xx} = ',num2str(Rxx),...
    ' R_{uu} = ',num2str(Ruu),' P_{tf} = ',num2str(Ptf)])
legend('Numerical','Analytic','Pss','Location','West')
xlabel('time');ylabel('P')
if Ruu > 10
    print -r300 -dpng reg2_13.png;
else
    print -r300 -dpng reg2_23.png;
end
```

```
68
69  Pan2=inline('((A*Ptf+Rxx)*sinh(beta*t)+beta*Ptf*cosh(beta*t))/((B^2*Ptf/Ruu-A)*sinh(beta*t)+beta*cosh(beta*t))');
70  x1=zeros(1,length(time));x2=zeros(1,length(time));
71  xcurr1=[1]';xcurr2=[1]';
72  for kk=1:length(time)-1
73    x1(:,kk)=xcurr1;  x2(:,kk)=xcurr2;
74    xdot1=(A-B*Ruu^(-1)*B'*Pan2(A,B,Ptf,Ruu,Rxx,beta,tf-(kk-1)*dt))*x1(:,kk);
75    xdot2=(A-B*klqr)*x2(:,kk);
76    xcurr1=xcurr1+xdot1*dt;
77    xcurr2=xcurr2+xdot2*dt;
78  end
79
80  figure(2);clf
81  plot(time,x2,'bs',time,x1,'r.');xlabel('time');ylabel('x')
82  title(['A = ',num2str(A),' B = ',num2str(B),' R_{xx} = ',num2str(Rxx),...
83         ' R_{uu} = ',num2str(Ruu),' P_{tf} = ',num2str(Ptf)])
84  legend('K_{ss}','K_{analytic}','Location','NorthEast')
85  if Ruu > 10
86      print -r300 -dpng reg2_11.png;
87  else
88      print -r300 -dpng reg2_22.png;
89  end
```

```
1   function [doy]=doty(t,y);
2   global A B Rxx Ruu;
3   P=unvec(y);
4   dotP=P*A+A'*P+Rxx-P*B*Ruu^(-1)*B'*P;
5   doy=vec(dotP);
6   return
```

```
1   function y=vec(P);
2
3   y=[];
4   for ii=1:length(P);
5       y=[y;P(ii,ii:end)'];
6   end
7
8   return
```

```
1   function P=unvec(y);
2
3   N=max(roots([1 1 -2*length(y)]));
4   P=[];kk=N;kk0=1;
5   for ii=1:N;
6       P(ii,ii:N)=[y(kk0+[0:kk-1])]';
7       kk0=kk0+kk;
8       kk=kk-1;
9   end
10  P=(P+P')-diag(diag(P));
11  return
```

Finite Time LQR Example

- Simple system with $t_0 = 0$ and $t_f = 10$sec.

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

$$2J = x^T(10) \begin{bmatrix} 0 & 0 \\ 0 & h \end{bmatrix} x(10) + \int_0^{10} \left\{ x^T(t) \begin{bmatrix} q & 0 \\ 0 & 0 \end{bmatrix} x(t) + ru^2(t) \right\} dt$$

- Compute gains using both time-varying $P(t)$ and steady-state value.

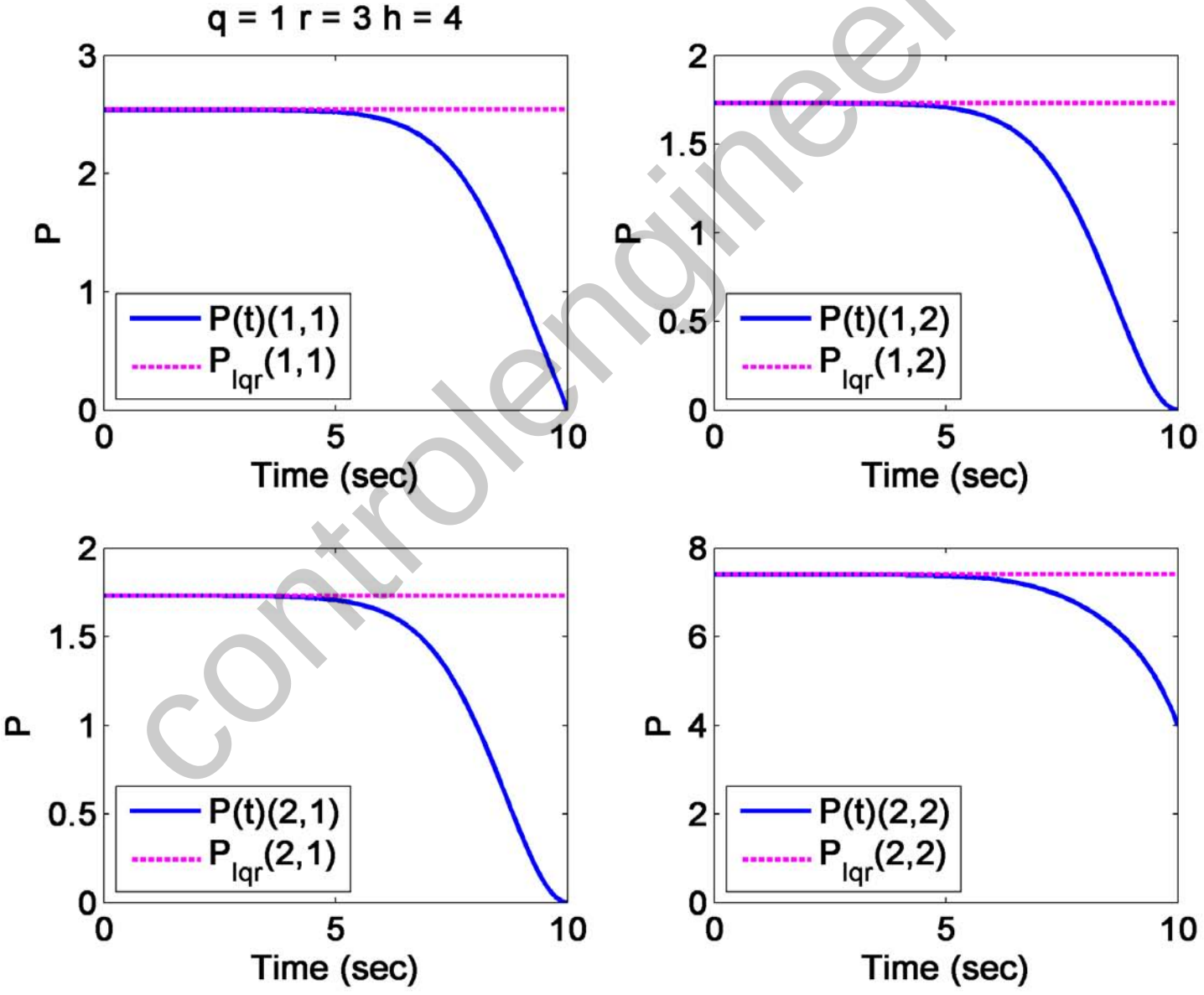


Figure 4.5: Set $q = 1$, $r = 3$, $h = 4$

- Find state solution $x(0) = [1 \quad 1]^T$ using both sets of gains

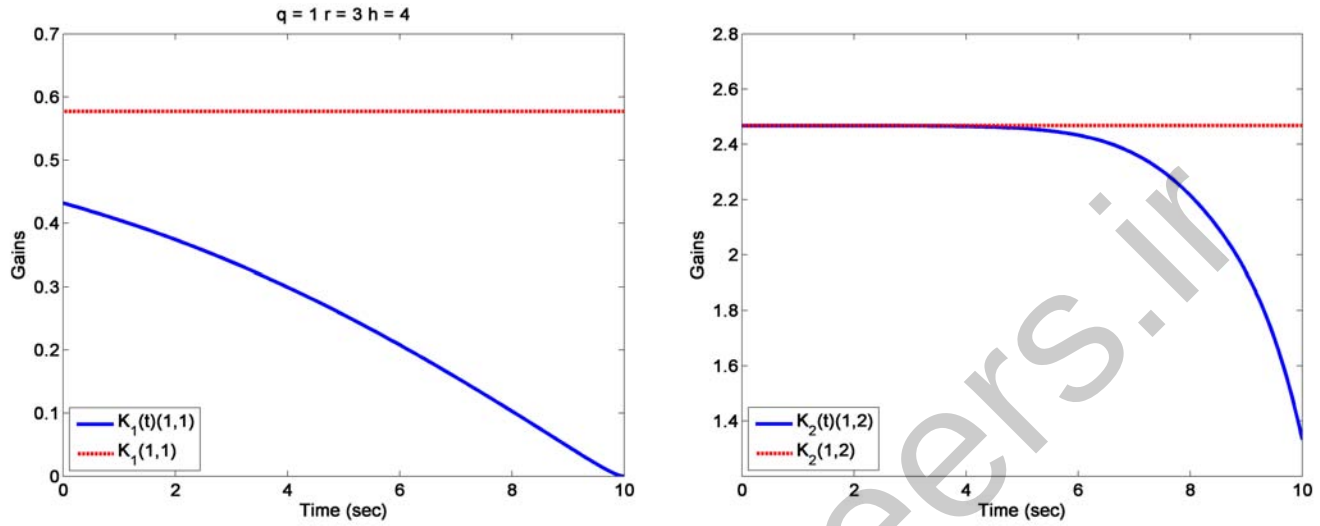


Figure 4.6: Time-varying and constant gains - $K_{lqr} = [0.5774 \; 2.4679]$

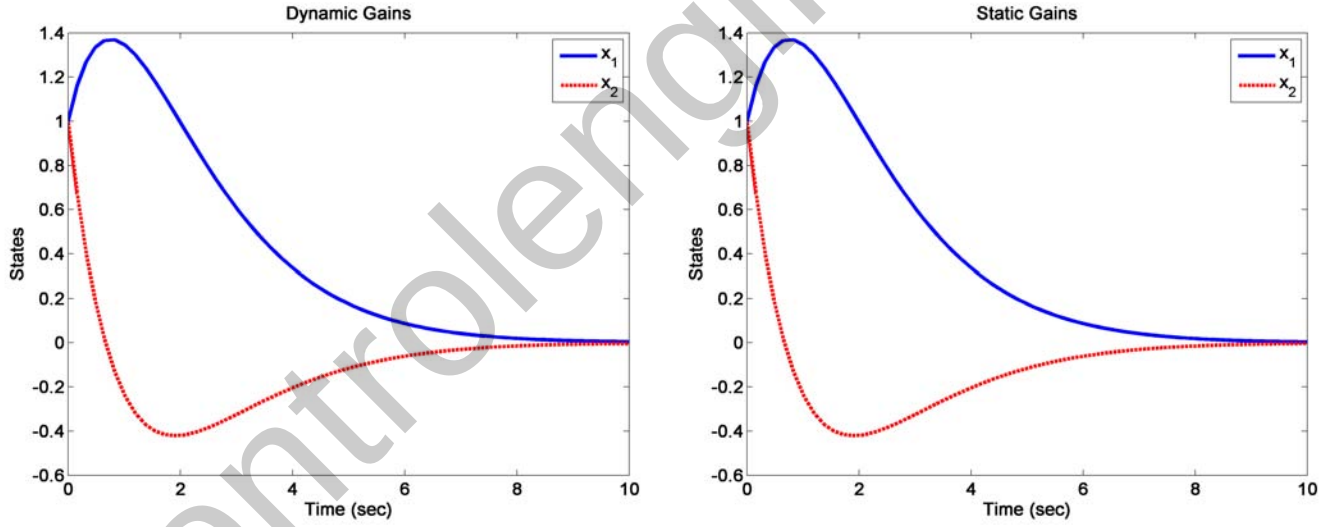


Figure 4.7: State response - Constant gain and time-varying gain almost indistinguishable because the transient dies out before the time at which the gains start to change – effectively a steady state problem.

- For most applications, the static gains are more than adequate - it is only when the terminal conditions are important in a short-time horizon problem that the time-varying gains should be used.
  - **Significant savings** in implementation complexity & computation.

Finite Time LQR Example

```
% Simple LQR example showing time varying P and gains
% 16.323 Spring 2008
% Jonathan How
% reg1.m
%
clear all;%close all;
set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontWeight','demi')
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontWeight','demi')
global A B Rxx Ruu
jprint = 0;

h=4;q=1;r=3;
A=[0 1;0 1];B=[0 1]';tf=10;dt=.01;
Ptf=[0 0;0 h];Rxx=[q 0;0 0];Ruu=r;
Ptf=[0 0;0 1];Rxx=[q 0;0 100];Ruu=r;

% alternative calc of Ricc solution
H=[A -B*B'/r ; -Rxx -A'];
[V,D]=eig(H); % check order of eigenvalues
Psi11=V(1:2,1:2);
Psi21=V(3:4,1:2);
Ptest=Psi21*inv(Psi11);

if 0

% integrate the P backwards (crude)
time=[0:dt:tf];
P=zeros(2,2,length(time));
K=zeros(1,2,length(time));
Pcurr=Ptf;
for kk=0:length(time)-1
  P(:,:,length(time)-kk)=Pcurr;
  K(:,:,length(time)-kk)=inv(Ruu)*B'*Pcurr;
  Pdot=-Pcurr*A-A'*Pcurr-Rxx+Pcurr*B*inv(Ruu)*B'*Pcurr;
  Pcurr=Pcurr-dt*Pdot;
end

else
% integrate forwards (ODE)
options=odeset('RelTol',1e-6,'AbsTol',1e-6)
[tau,y]=ode45(@doty,[0 tf],vec(Ptf),options);
Tnum=[];Pnum=[];Fnum=[];
for i=1:length(tau)
    time(length(tau)-i+1)=tf-tau(i);
    temp=unvec(y(i,:));
    P(:,:,length(tau)-i+1)=temp;
    K(:,:,length(tau)-i+1)=inv(Ruu)*B'*temp;
end

end % if 0

% get the SS result from LQR
[klqr,Plqr]=lqr(A,B,Rxx,Ruu);

x1=zeros(2,1,length(time));
x2=zeros(2,1,length(time));
xcurr1=[1 1]';
xcurr2=[1 1]';
for kk=1:length(time)-1
  dt=time(kk+1)-time(kk);
  x1(:,:,kk)=xcurr1;
  x2(:,:,kk)=xcurr2;
  xdot1=(A-B*K(:,:,kk))*x1(:,:,kk);
  xdot2=(A-B*klqr)*x2(:,:,kk);
  xcurr1=xcurr1+xdot1*dt;
  xcurr2=xcurr2+xdot2*dt;
end
```

```
x1(:,:,length(time))=xcurr1;
x2(:,:,length(time))=xcurr2;

figure(5);clf
subplot(221)
plot(time,squeeze(K(1,1,:)),[0 10],[1 1]*klqr(1),'m--','LineWidth',2)
legend('K_1(t)','K_1')
xlabel('Time (sec)');ylabel('Gains')
title(['q = ',num2str(1),' r = ',num2str(r),' h = ',num2str(h)])
subplot(222)
plot(time,squeeze(K(1,2,:)),[0 10],[1 1]*klqr(2),'m--','LineWidth',2)
legend('K_2(t)','K_2')
xlabel('Time (sec)');ylabel('Gains')
subplot(223)
plot(time,squeeze(x1(1,1,:)),time,squeeze(x1(2,1,:)),'m--','LineWidth',2),
legend('x_1','x_2')
xlabel('Time (sec)');ylabel('States');title('Dynamic Gains')
subplot(224)
plot(time,squeeze(x2(1,1,:)),time,squeeze(x2(2,1,:)),'m--','LineWidth',2),
legend('x_1','x_2')
xlabel('Time (sec)');ylabel('States');title('Static Gains')

figure(6);clf
subplot(221)
plot(time,squeeze(P(1,1,:)),[0 10],[1 1]*Plqr(1,1),'m--','LineWidth',2)
legend('P(t)(1,1)','P_{lqr}(1,1)','Location','SouthWest')
xlabel('Time (sec)');ylabel('P')
title(['q = ',num2str(1),' r = ',num2str(r),' h = ',num2str(h)])
subplot(222)
plot(time,squeeze(P(1,2,:)),[0 10],[1 1]*Plqr(1,2),'m--','LineWidth',2)
legend('P(t)(1,2)','P_{lqr}(1,2)','Location','SouthWest')
xlabel('Time (sec)');ylabel('P')
subplot(223)
plot(time,squeeze(P(2,1,:)),[0 10],[1 1]*squeeze(Plqr(2,1)),'m--','LineWidth',2),
legend('P(t)(2,1)','P_{lqr}(2,1)','Location','SouthWest')
xlabel('Time (sec)');ylabel('P')
subplot(224)
plot(time,squeeze(P(2,2,:)),[0 10],[1 1]*squeeze(Plqr(2,2)),'m--','LineWidth',2),
legend('P(t)(2,2)','P_{lqr}(2,2)','Location','SouthWest')
xlabel('Time (sec)');ylabel('P')
axis([0 10 0 8])
if jprint;    print -dpng -r300 reg1_6.png
end

figure(1);clf
plot(time,squeeze(K(1,1,:)),[0 10],[1 1]*klqr(1),'r--','LineWidth',3)
legend('K_1(t)(1,1)','K_1(1,1)','Location','SouthWest')
xlabel('Time (sec)');ylabel('Gains')
title(['q = ',num2str(1),' r = ',num2str(r),' h = ',num2str(h)])
print -dpng -r300 reg1_1.png
figure(2);clf
plot(time,squeeze(K(1,2,:)),[0 10],[1 1]*klqr(2),'r--','LineWidth',3)
legend('K_2(t)(1,2)','K_2(1,2)','Location','SouthWest')
xlabel('Time (sec)');ylabel('Gains')
if jprint;    print -dpng -r300 reg1_2.png
end

figure(3);clf
plot(time,squeeze(x1(1,1,:)),time,squeeze(x1(2,1,:)),'r--','LineWidth',3),
legend('x_1','x_2')
xlabel('Time (sec)');ylabel('States');title('Dynamic Gains')
if jprint;    print -dpng -r300 reg1_3.png
end

figure(4);clf
plot(time,squeeze(x2(1,1,:)),time,squeeze(x2(2,1,:)),'r--','LineWidth',3),
legend('x_1','x_2')
xlabel('Time (sec)');ylabel('States');title('Static Gains');
if jprint;    print -dpng -r300 reg1_4.png
end
```

Weighting Matrix Selection

- A good rule of thumb when selecting the weighting matrices R_{xx} and R_{uu} is to normalize the signals:

$$R_{\text{xx}} = \begin{bmatrix} \frac{\alpha_1^2}{(x_1)_{\max}^2} & & & \\ & \frac{\alpha_2^2}{(x_2)_{\max}^2} & & \\ & & \ddots & \\ & & & \frac{\alpha_n^2}{(x_n)_{\max}^2} \end{bmatrix}$$

$$R_{\text{uu}} = \rho \begin{bmatrix} \frac{\beta_1^2}{(u_1)_{\max}^2} & & & \\ & \frac{\beta_2^2}{(u_2)_{\max}^2} & & \\ & & \ddots & \\ & & & \frac{\beta_m^2}{(u_m)_{\max}^2} \end{bmatrix}$$

- The $(x_i)_{\max}$ and $(u_i)_{\max}$ represent the largest desired response/control input for that component of the state/actuator signal.
- The $\sum_i \alpha_i^2 = 1$ and $\sum_i \beta_i^2 = 1$ are used to add an additional relative weighting on the various components of the state/control
- ρ is used as the last relative weighting between the control and state penalties $\Rightarrow$ gives us a relatively concrete way to discuss the relative size of R_{xx} and R_{uu} and their ratio $R_{\text{xx}}/R_{\text{uu}}$
- Note: to **directly** compare the continuous and discrete LQR, you must modify the weighting matrices for the discrete case, as outlined here using `lqrd`.

16.323 Lecture 5

Calculus of Variations

- Calculus of Variations
- Most books cover this material well, but Kirk Chapter 4 does a particularly nice job.
- See here for online reference.

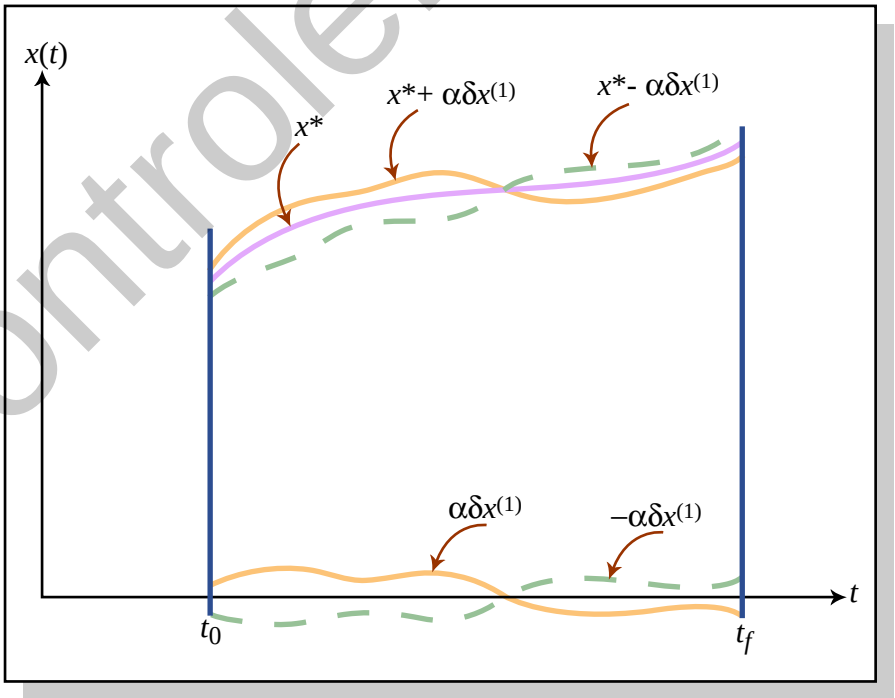


Figure by MIT OpenCourseWare.

Calculus of Variations

- **Goal:** Develop alternative approach to solve general optimization problems for continuous systems – **variational calculus**
  - Formal approach will provide new insights for constrained solutions, and a more direct path to the solution for other problems.

- **Main issue** – General control problem, the cost is a function of functions $\mathbf{x}(t)$ and $\mathbf{u}(t)$.

$$\min J = h(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t))\, dt$$

subject to

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ \mathbf{x}(t_0), t_0 &\ \text{given} \\ m(\mathbf{x}(t_f), t_f) &= 0 \end{aligned}$$

  - Call $J(\mathbf{x}(t), \mathbf{u}(t))$ a **functional**.

- Need to investigate how to find the optimal values of a functional.
  - For a function, we found the gradient, and set it to zero to find the stationary points, and then investigated the higher order derivatives to determine if it is a maximum or minimum.
  - Will investigate something similar for functionals.

- **Maximum and Minimum of a Function**
  - A function $f(\mathbf{x})$ has a local minimum at $\mathbf{x}^\star$ if
    $$f(\mathbf{x}) \geq f(\mathbf{x}^\star)$$
    for all admissible $\mathbf{x}$ in $\|\mathbf{x} - \mathbf{x}^\star\| \leq \epsilon$
  - Minimum can occur at (i) stationary point, (ii) at a boundary, or (iii) a point of discontinuous derivative.
  - If only consider stationary points of the differentiable function $f(\mathbf{x})$, then statement equivalent to requiring that differential of f satisfy:
    $$df = \frac{\partial f}{\partial \mathbf{x}} d\mathbf{x} = 0$$
    for all small $d\mathbf{x}$, which gives the same necessary condition from Lecture 1
    $$\frac{\partial f}{\partial \mathbf{x}} = 0$$

- Note that this definition used norms to compare two vectors. Can do the same thing with functions $\Rightarrow$ distance between two functions
  $$d = \|\mathbf{x}_2(t) - \mathbf{x}_1(t)\|$$
  where
  1. $\|\mathbf{x}(t)\| \geq 0$ for all $\mathbf{x}(t)$, and $\|\mathbf{x}(t)\| = 0$ only if $\mathbf{x}(t) = 0$ for all t in the interval of definition.
  2. $\|a\mathbf{x}(t)\| = |a|\|\mathbf{x}(t)\|$ for all real scalars a.
  3. $\|\mathbf{x}_1(t) + \mathbf{x}_2(t)\| \leq \|\mathbf{x}_1(t)\| + \|\mathbf{x}_2(t)\|$

- Common function norm:
  $$\|\mathbf{x}(t)\|_2 = \left(\int_{t_0}^{t_f} \mathbf{x}(t)^T \mathbf{x}(t) dt \right)^{1/2}$$

- **Maximum and Minimum of a Functional**
  - A functional $J(\mathbf{x}(t))$ has a local minimum at $\mathbf{x}^\star(t)$ if

$$J(\mathbf{x}(t)) \geq J(\mathbf{x}^\star(t))$$

    for all admissible $\mathbf{x}(t)$ in $\|\mathbf{x}(t) - \mathbf{x}^\star(t)\| \leq \epsilon$

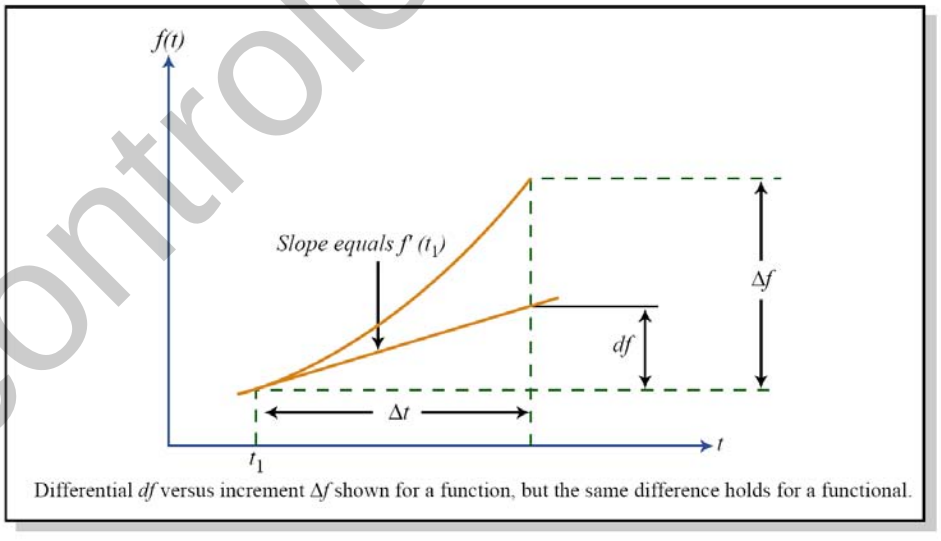
- Now define something equivalent to the differential of a function - called a **variation** of a functional.
  - An **increment** of a functional

$$\Delta J(\mathbf{x}(t), \delta\mathbf{x}(t)) = J(\mathbf{x}(t) + \delta\mathbf{x}(t)) - J(\mathbf{x}(t))$$

  - A **variation** of the functional is a linear approximation of this increment:

$$\Delta J(\mathbf{x}(t), \delta\mathbf{x}(t)) = \delta J(\mathbf{x}(t), \delta\mathbf{x}(t)) + H.O.T.$$

    i.e. $\delta J(\mathbf{x}(t), \delta\mathbf{x}(t))$ is linear in $\delta\mathbf{x}(t)$.



Differential *df* versus increment Δ*f* shown for a function, but the same difference holds for a functional.

Figure 5.1: Differential df versus increment Δf shown for a function, but the same difference holds for a functional.

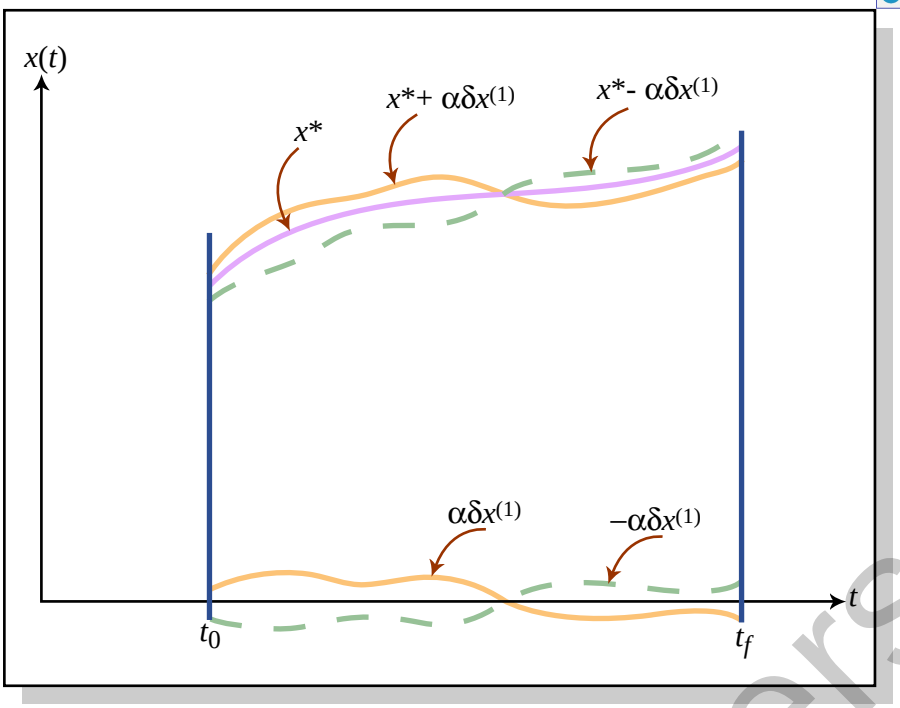


Figure by MIT OpenCourseWare.

Figure 5.2: Visualization of perturbations to function $x(t)$ by $\delta x(t)$ – it is a potential change in the value of x over the entire time period of interest. Typically require that if $x(t)$ is in some class (i.e., continuous), that $x(t) + \delta x(t)$ is also in that class.

- **Fundamental Theorem of the Calculus of Variations**
  - Let $\mathbf{x}$ be a function of t in the class Ω, and $J(\mathbf{x})$ be a differentiable functional of $\mathbf{x}$. Assume that the functions in Ω are not constrained by any boundaries.
  - If $\mathbf{x}^\star$ is an extremal function, then the variation of J must vanish on $\mathbf{x}^\star$, i.e. for all admissible $\delta\mathbf{x}$,

$$\delta J(\mathbf{x}(t), \delta\mathbf{x}(t)) = 0$$

  - Proof is in Kirk, page 121, but it is relatively straightforward.

- How compute the variation? If $J(\mathbf{x}(t)) = \int_{t_0}^{t_f} f(\mathbf{x}(t))dt$ where f has cts first and second derivatives with respect to $\mathbf{x}$, then

$$\begin{aligned}\delta J(\mathbf{x}(t), \delta\mathbf{x}) &= \int_{t_0}^{t_f} \left\{\frac{\partial f(\mathbf{x}(t))}{\partial \mathbf{x}(t)}\right\} \delta\mathbf{x}dt + f(\mathbf{x}(t_f))\delta t_f - f(\mathbf{x}(t_0))\delta t_0 \\ &= \int_{t_0}^{t_f} f_{\mathbf{x}}(\mathbf{x}(t))\delta\mathbf{x}dt + f(\mathbf{x}(t_f))\delta t_f - f(\mathbf{x}(t_0))\delta t_0\end{aligned}$$

Variation Examples: Scalar

- For more general problems, first consider the cost evaluated on a scalar function $x(t)$ with t_0, t_f and the curve endpoints fixed.

$$J(x(t)) = \int_{t_0}^{t_f} g(x(t), \dot{x}(t), t)dt$$

$$\Rightarrow \quad \delta J(x(t), \delta x) = \int_{t_0}^{t_f} \left[g_x(x(t), \dot{x}(t), t)\delta x + g_{\dot{x}}(x(t), \dot{x}(t), t)\delta \dot{x} \right] dt$$

  - Note that

$$\delta \dot{x} = \frac{d}{dt}\delta x$$

  so δx and $\delta \dot{x}$ **are not independent**.

- Integrate by parts:

$$\int udv \equiv uv - \int vdu$$

  with $u = g_{\dot{x}}$ and $dv = \delta \dot{x} dt$ to get:

$$\begin{aligned}
\delta J(x(t), \delta x) &= \int_{t_0}^{t_f} g_x(x(t), \dot{x}(t), t)\delta x dt + [g_{\dot{x}}(x(t), \dot{x}(t), t)\delta x]_{t_0}^{t_f} \\
&\quad - \int_{t_0}^{t_f} \frac{d}{dt} g_{\dot{x}}(x(t), \dot{x}(t), t)\delta x dt \\
&= \int_{t_0}^{t_f} \left[g_x(x(t), \dot{x}(t), t) - \frac{d}{dt} g_{\dot{x}}(x(t), \dot{x}(t), t) \right] \delta x(t) dt \\
&\quad + [g_{\dot{x}}(x(t), \dot{x}(t), t)\delta x]_{t_0}^{t_f}
\end{aligned}$$

- Since $x(t_0),\ x(t_f)$ given, then $\delta x(t_0) = \delta x(t_f) = 0$, yielding

$$\delta J(x(t), \delta x) = \int_{t_0}^{t_f} \left[g_x(x(t), \dot{x}(t), t) - \frac{d}{dt} g_{\dot{x}}(x(t), \dot{x}(t), t) \right] \delta x(t) dt$$

- Recall need $\delta J = 0$ for all admissible $\delta x(t)$, which are arbitrary within $(t_0, t_f) \Rightarrow$ the (first order) necessary condition for a maximum or minimum is called **Euler Equation**:

$$\frac{\partial g(x(t), \dot{x}(t), t)}{\partial x} - \frac{d}{dt}\left(\frac{\partial g(x(t), \dot{x}(t), t)}{\partial \dot{x}}\right) = 0$$

- **Example**: Find the curve that gives the shortest distance between 2 points in a plane (x_0, y_0) and (x_f, y_f).
  - Cost function – sum of differential arc lengths:

$$\begin{aligned} J &= \int_{x_0}^{x_f} ds = \int_{x_0}^{x_f} \sqrt{(dx)^2 + (dy)^2} \\ &= \int_{x_0}^{x_f} \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx \end{aligned}$$

  - Take y as dependent variable, and x as independent one

$$\frac{dy}{dx} \to \dot{y}$$

  - New form of the cost:

$$J = \int_{x_0}^{x_f} \sqrt{1 + \dot{y}^2}\, dx \to \int_{x_0}^{x_f} g(\dot{y}) dx$$

  - Take partials: $\partial g / \partial y = 0$, and

$$\begin{aligned} \frac{d}{dx}\left(\frac{\partial g}{\partial \dot{y}}\right) &= \frac{d}{d\dot{y}}\left(\frac{\partial g}{\partial \dot{y}}\right)\frac{d\dot{y}}{dx} \\ &= \frac{d}{d\dot{y}}\left(\frac{\dot{y}}{(1+\dot{y}^2)^{1/2}}\right)\ddot{y} = \frac{\ddot{y}}{(1+\dot{y}^2)^{3/2}} = 0 \end{aligned}$$

    which implies that $\ddot{y} = 0$

  - Most general curve with $\ddot{y} = 0$ is a line $y = c_1 x + c_2$

Vector Functions

- Can generalize the problem by including several (N) functions $x_i(t)$ and possibly free endpoints

$$J(\mathbf{x}(t)) = \int_{t_0}^{t_f} g(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)dt$$

with t_0, t_f, $\mathbf{x}(t_0)$ fixed.

- Then (drop the arguments for brevity)

$$\delta J(\mathbf{x}(t), \delta\mathbf{x}) = \int_{t_0}^{t_f} \left[g_{\mathbf{x}}\delta\mathbf{x}(t) + g_{\dot{\mathbf{x}}}\delta\dot{\mathbf{x}}(t)\right] dt$$

  – Integrate by parts to get:

$$\delta J(\mathbf{x}(t), \delta\mathbf{x}) = \int_{t_0}^{t_f} \left[g_{\mathbf{x}} - \frac{d}{dt}g_{\dot{\mathbf{x}}}\right] \delta\mathbf{x}(t)dt + g_{\dot{\mathbf{x}}}(\mathbf{x}(t_f), \dot{\mathbf{x}}(t_f), t_f)\delta\mathbf{x}(t_f)$$

- The requirement then is that for $t \in (t_0, t_f)$, $\mathbf{x}(t)$ must satisfy

$$\frac{\partial g}{\partial \mathbf{x}} - \frac{d}{dt}\frac{\partial g}{\partial \dot{\mathbf{x}}} = 0$$

where $\mathbf{x}(t_0) = \mathbf{x}_0$ which are the given N boundary conditions, and the remaining N more BC follow from:

– $\mathbf{x}(t_f) = \mathbf{x}_f$ if $\mathbf{x}_f$ is given as fixed,

– If $\mathbf{x}(t_f)$ are free, then

$$\frac{\partial g(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{\mathbf{x}}(t_f)} = 0$$

- Note that we could also have a mixture, where parts of $\mathbf{x}(t_f)$ are given as fixed, and other parts are free – just use the rules above on each component of $x_i(t_f)$

Free Terminal Time

- Now consider a slight variation: the goal is to minimize

$$J(\mathbf{x}(t)) = \int_{t_0}^{t_f} g(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)dt$$

with t_0, $\mathbf{x}(t_0)$ fixed, t_f free, and various constraints on $\mathbf{x}(t_f)$

- Compute variation of the functional considering 2 candidate solutions:
  - $\mathbf{x}(t)$, which we consider to be a perturbation of the optimal $\mathbf{x}^\star(t)$ (that we need to find)

$$\delta J(\mathbf{x}^\star(t), \delta\mathbf{x}) = \int_{t_0}^{t_f} \left[g_{\mathbf{x}}\delta\mathbf{x}(t) + g_{\dot{\mathbf{x}}}\delta\dot{\mathbf{x}}(t)\right] dt + g(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f)\delta t_f$$

  - Integrate by parts to get:

$$\begin{aligned}\delta J(\mathbf{x}^\star(t), \delta\mathbf{x}) = & \int_{t_0}^{t_f} \left[g_{\mathbf{x}} - \frac{d}{dt}g_{\dot{\mathbf{x}}}\right] \delta\mathbf{x}(t)dt \\ & + g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f)\delta\mathbf{x}(t_f) \\ & + g(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f)\delta t_f\end{aligned}$$

- Looks standard so far, but we have to be careful how we handle the terminal conditions

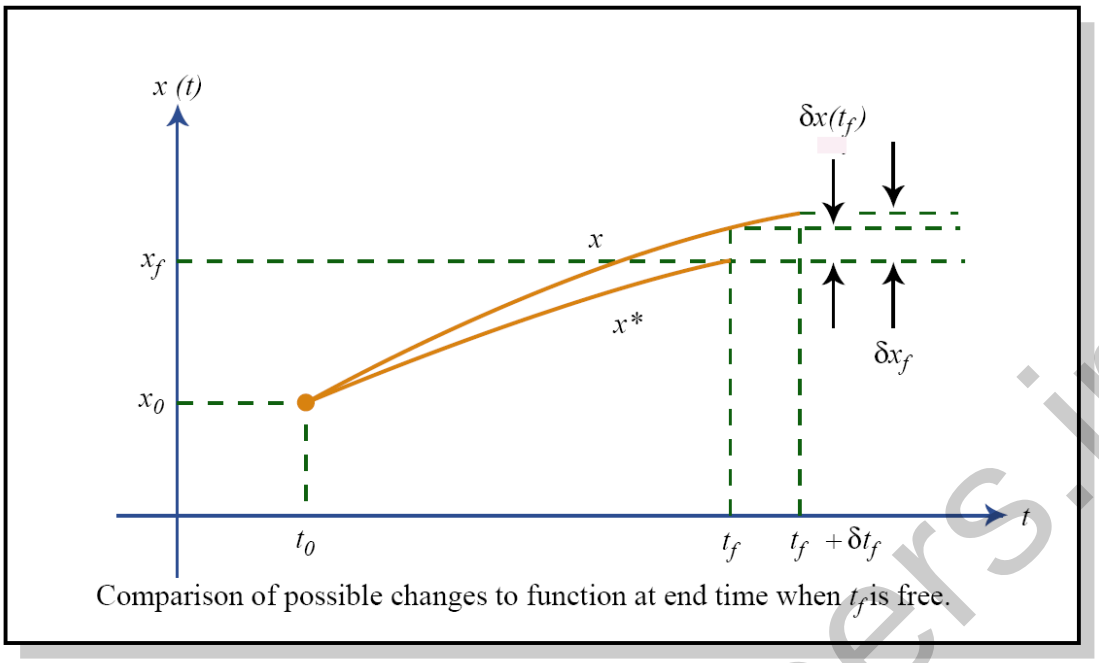


Figure by MIT OpenCourseWare.

Figure 5.3: Comparison of possible changes to function at end time when t_f is free.

- By definition, $\delta\mathbf{x}(t_f)$ is the difference between two admissible functions at time t_f (in this case the optimal solution $\mathbf{x}^\star$ and another candidate $\mathbf{x}$).
  - But in this case, must also account for possible changes to δt_f.
  - Define $\delta\mathbf{x}_f$ as being the difference between the ends of the two possible functions – **total possible change** in the final state:

$$\delta\mathbf{x}_f \approx \delta\mathbf{x}(t_f) + \dot{\mathbf{x}}^\star(t_f)\delta t_f$$

  so $\delta\mathbf{x}(t_f) \neq \delta\mathbf{x}_f$ in general.

- Substitute to get

$$\delta J(\mathbf{x}^\star(t), \delta\mathbf{x}) = \int_{t_0}^{t_f} \left[g_{\mathbf{x}} - \frac{d}{dt} g_{\dot{\mathbf{x}}} \right] \delta\mathbf{x}(t)dt + g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f)\delta\mathbf{x}_f$$

$$+ \left[g(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f) - g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f)\dot{\mathbf{x}}^\star(t_f) \right] \delta t_f$$

- Independent of the terminal constraint, the conditions on the solution $\mathbf{x}^\star(t)$ to be an extremal for this case are that it satisfy the Euler equations

$$g_{\mathbf{x}}(\mathbf{x}^\star(t), \dot{\mathbf{x}}^\star(t), t) - \frac{d}{dt} g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t), \dot{\mathbf{x}}^\star(t), t) = 0$$

  - Now consider the additional constraints on the individual elements of $\mathbf{x}^\star(t_f)$ and t_f to find the other boundary conditions

- Type of terminal constraints determines how we treat $\delta\mathbf{x}_f$ and δt_f
  1. Unrelated
  2. Related by a simple function $\mathbf{x}(t_f) = \Theta(t_f)$
  3. Specified by a more complex constraint $\mathbf{m}(\mathbf{x}(t_f), t_f) = 0$

- **Type 1:** If t_f and $\mathbf{x}(t_f)$ are free but unrelated, then $\delta\mathbf{x}_f$ and δt_f are independent and arbitrary $\Rightarrow$ their coefficients must both be zero.

$$\begin{aligned} g_{\mathbf{x}}(\mathbf{x}^\star(t), \dot{\mathbf{x}}^\star(t), t) - \frac{d}{dt} g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t), \dot{\mathbf{x}}^\star(t), t) &= 0 \\ g(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f) - g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f)\dot{\mathbf{x}}^\star(t_f) &= 0 \\ g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f) &= 0 \end{aligned}$$

  - Which makes it clear that this is a **two-point boundary value problem**, as we now have conditions at both t_0 and t_f

- **Type 2:** If t_f and $\mathbf{x}(t_f)$ are free but related as $\mathbf{x}(t_f) = \Theta(t_f)$, then

$$\delta \mathbf{x}_f = \frac{d\Theta}{dt}(t_f)\delta t_f$$

  - Substitute and collect terms gives

$$\begin{aligned}\delta J &= \int_{t_0}^{t_f} \left[g_{\mathbf{x}} - \frac{d}{dt} g_{\dot{\mathbf{x}}} \right] \delta \mathbf{x} dt + \left[g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f) \frac{d\Theta}{dt}(t_f) \right. \\ &+ \left. g(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f) - g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f)\dot{\mathbf{x}}^\star(t_f) \right] \delta t_f\end{aligned}$$

  - Set coefficient of δt_f to zero (it is arbitrary) $\Rightarrow$ full conditions

$$\begin{aligned} g_{\mathbf{x}}(\mathbf{x}^\star(t), \dot{\mathbf{x}}^\star(t), t) - \frac{d}{dt} g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t), \dot{\mathbf{x}}^\star(t), t) &= 0 \\ g_{\dot{\mathbf{x}}}(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f) \left[\frac{d\Theta}{dt}(t_f) - \dot{\mathbf{x}}^\star(t_f) \right] + g(\mathbf{x}^\star(t_f), \dot{\mathbf{x}}^\star(t_f), t_f) &= 0 \end{aligned}$$

  - Last equation called the **Transversality Condition**

- To handle third type of terminal condition, must address solution of constrained problems.

Image removed due to copyright restrictions.

Figure 5.4: Summary of possible terminal constraints (Kirk, page 151)

Example: 5–1

- Find the shortest curve from the origin to a specified line.

- **Goal:** minimize the cost functional (See page 5–6)

$$J = \int_{t_0}^{t_f} \sqrt{1 + \dot{x}^2(t)}\, dt$$

given that $t_0 = 0$, $x(0) = 0$, and t_f and $x(t_f)$ are free, but $x(t_f)$ must line on the line

$$\theta(t) = -5t + 15$$

- Since $g(x, \dot{x}, t)$ is only a function of $\dot{x}$, Euler equation reduces to

$$\frac{d}{dt}\left[\frac{\dot{x}^\star(t)}{[1 + \dot{x}^\star(t)^2]^{1/2}}\right] = 0$$

which after differentiating and simplifying, gives $\ddot{x}^\star(t) = 0 \Rightarrow$ answer is a straight line

$$x^\star(t) = c_1 t + c_0$$

but since $x(0) = 0$, then $c_0 = 0$

- Transversality condition gives

$$\left[\frac{\dot{x}^\star(t_f)}{[1 + \dot{x}^\star(t_f)^2]^{1/2}}\right][-5 - \dot{x}^\star(t_f)] + [1 + \dot{x}^\star(t_f)^2]^{1/2} = 0$$

that simplifies to

$$[\dot{x}^\star(t_f)]\,[-5 - \dot{x}^\star(t_f)] + [1 + \dot{x}^\star(t_f)^2] = -5\dot{x}^\star(t_f) + 1 = 0$$

so that $\dot{x}^\star(t_f) = c_1 = 1/5$

  - Not a surprise, as this gives the slope of a line orthogonal to the constraint line.

- To find final time: $x(t_f) = -5t_f + 15 = t_f/5$ which gives $t_f \approx 2.88$

Example: 5–2

- Had the terminal constraint been a bit more challenging, such as

$$\Theta(t) = \frac{1}{2}([t-5]^2 - 1) \Rightarrow \frac{d\Theta}{dt} = t - 5$$

- Then the transversality condition gives

$$\left[\frac{\dot{x}^\star(t_f)}{[1+\dot{x}^\star(t_f)^2]^{1/2}}\right][t_f - 5 - \dot{x}^\star(t_f)] + [1+\dot{x}^\star(t_f)^2]^{1/2} = 0$$

$$[\dot{x}^\star(t_f)]\,[t_f - 5 - \dot{x}^\star(t_f)] + [1+\dot{x}^\star(t_f)^2] = 0$$

$$c_1\,[t_f - 5] + 1 = 0$$

- Now look at $x^\star(t)$ and $\Theta(t)$ at t_f

$$x^\star(t_f) = -\frac{t_f}{(t_f-5)} = \frac{1}{2}([t_f-5]^2 - 1)$$

which gives $t_f = 3$, $c_1 = 1/2$ and $x^\star(t_f) = t/2$

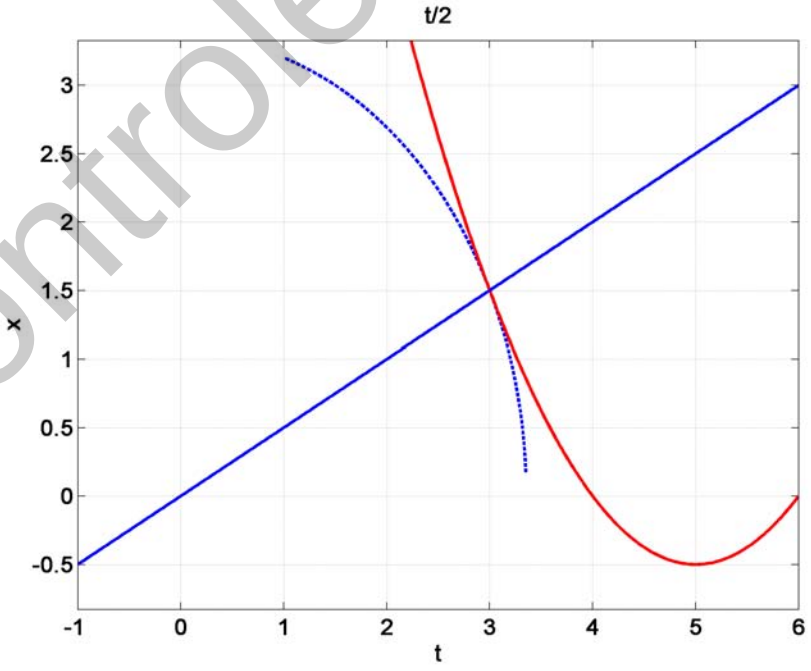


Figure 5.5: Quadratic terminal constraint.

Corner Conditions

- Key generalization of the preceding is to allow the possibility that the solutions not be as **smooth**
  - Assume that $\mathbf{x}(t)$ cts, but allow discontinuities in $\dot{\mathbf{x}}(t)$, which occur at **corners**.
  - Naturally occur when intermediate state constraints imposed, or with jumps in the control signal.

- **Goal:** with t_0, t_f, $\mathbf{x}(t_0)$, and $\mathbf{x}(t_f)$ fixed, minimize cost functional

$$J(\mathbf{x}(t),t) = \int_{t_0}^{t_f} g(\mathbf{x}(t),\dot{\mathbf{x}}(t),t)dt$$

  - Assume g has cts first/second derivatives wrt all arguments
  - Even so, $\dot{\mathbf{x}}$ discontinuity could lead to a discontinuity in g.

- Assume that $\dot{\mathbf{x}}$ has a discontinuity at some time $t_1 \in (t_0, t_f)$, which is not fixed (or typically known). Divide cost into 2 regions:

$$J(\mathbf{x}(t),t) = \int_{t_0}^{t_1} g(\mathbf{x}(t),\dot{\mathbf{x}}(t),t)dt + \int_{t_1}^{t_f} g(\mathbf{x}(t),\dot{\mathbf{x}}(t),t)dt$$

- Expand as before – note that t_1 is not fixed

$$\begin{aligned}\delta J \;=\; & \int_{t_0}^{t_1} \left[\frac{\partial g}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial g}{\partial \dot{\mathbf{x}}}\delta\dot{\mathbf{x}}\right] dt + g(t_1^-)\delta t_1 \\ & + \int_{t_1}^{t_f} \left[\frac{\partial g}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial g}{\partial \dot{\mathbf{x}}}\delta\dot{\mathbf{x}}\right] dt - g(t_1^+)\delta t_1\end{aligned}$$

- Now IBP

$$\begin{aligned}\delta J &= \int_{t_0}^{t_1}\left[g_{\mathbf{x}} - \frac{d}{dt}\left(g_{\dot{\mathbf{x}}}\right)\right]\delta\mathbf{x}dt + g(t_1^-)\delta t_1 + g_{\dot{\mathbf{x}}}(t_1^-)\delta\mathbf{x}(t_1^-)\\ &+ \int_{t_1}^{t_f}\left[g_{\mathbf{x}} - \frac{d}{dt}\left(g_{\dot{\mathbf{x}}}\right)\right]\delta\mathbf{x}dt - g(t_1^+)\delta t_1 - g_{\dot{\mathbf{x}}}(t_1^+)\delta\mathbf{x}(t_1^+)\end{aligned}$$

- As on 5–9, must constrain $\delta\mathbf{x}_1$, which is the total variation in the solution at time t_1

$$\begin{aligned}\text{from lefthand side} \quad \delta\mathbf{x}_1 &= \delta\mathbf{x}(t_1^-) + \dot{\mathbf{x}}(t_1^-)\delta t_1\\ \text{from righthand side} \quad \delta\mathbf{x}_1 &= \delta\mathbf{x}(t_1^+) + \dot{\mathbf{x}}(t_1^+)\delta t_1\end{aligned}$$

  - Continuity requires that these two expressions for $\delta\mathbf{x}_1$ be equal
  - Already know that it is possible that $\dot{\mathbf{x}}(t_1^-) \neq \dot{\mathbf{x}}(t_1^+)$, so possible that $\delta\mathbf{x}(t_1^-) \neq \delta\mathbf{x}(t_1^+)$ as well.

- Substitute:

$$\begin{aligned}\delta J &= \int_{t_0}^{t_1}\left[g_{\mathbf{x}} - \frac{d}{dt}\left(g_{\dot{\mathbf{x}}}\right)\right]\delta\mathbf{x}dt + \left[g(t_1^-) - g_{\dot{\mathbf{x}}}(t_1^-)\dot{\mathbf{x}}(t_1^-)\right]\delta t_1 + g_{\dot{\mathbf{x}}}(t_1^-)\delta\mathbf{x}_1\\ &+ \int_{t_1}^{t_f}\left[g_{\mathbf{x}} - \frac{d}{dt}\left(g_{\dot{\mathbf{x}}}\right)\right]\delta\mathbf{x}dt - \left[g(t_1^+) - g_{\dot{\mathbf{x}}}(t_1^+)\dot{\mathbf{x}}(t_1^+)\right]\delta t_1 - g_{\dot{\mathbf{x}}}(t_1^+)\delta\mathbf{x}_1\end{aligned}$$

- Necessary conditions are then:

$$\begin{aligned}g_{\mathbf{x}} - \frac{d}{dt}\left(g_{\dot{\mathbf{x}}}\right) &= 0 \quad t \in (t_0, t_f)\\ g_{\dot{\mathbf{x}}}(t_1^-) &= g_{\dot{\mathbf{x}}}(t_1^+)\\ g(t_1^-) - g_{\dot{\mathbf{x}}}(t_1^-)\dot{\mathbf{x}}(t_1^-) &= g(t_1^+) - g_{\dot{\mathbf{x}}}(t_1^+)\dot{\mathbf{x}}(t_1^+)\end{aligned}$$

  - Last two are the **Weierstrass-Erdmann** conditions

- Necessary conditions given for a special set of the terminal conditions, but the form of the internal conditions unchanged by different terminal constraints
  - With several corners, there are a set of constraints for each
  - Can be used to demonstrate that there isn't a corner

- Typical instance that induces corners is intermediate time constraints of the form $\mathbf{x}(t_1) = \boldsymbol{\theta}(t_1)$.
  - i.e., the solution must touch a specified curve at some point in time during the solution.

- Slightly complicated in this case, because the constraint couples the allowable variations in $\delta\mathbf{x}_1$ and δt since

$$\delta\mathbf{x}_1 = \frac{d\boldsymbol{\theta}}{dt}\delta t_1 = \dot{\boldsymbol{\theta}}\delta t_1$$

  - But can eliminate $\delta\mathbf{x}_1$ in favor of δt_1 in the expression for δJ to get new corner condition:

$$g(t_1^-) + g_{\dot{\mathbf{x}}}(t_1^-)\left[\dot{\boldsymbol{\theta}}(t_1^-) - \dot{\mathbf{x}}(t_1^-)\right] = g(t_1^+) + g_{\dot{\mathbf{x}}}(t_1^+)\left[\dot{\boldsymbol{\theta}}(t_1^+) - \dot{\mathbf{x}}(t_1^+)\right]$$

  - So now $g_{\dot{\mathbf{x}}}(t_1^-) = g_{\dot{\mathbf{x}}}(t_1^+)$ no longer needed, but have $\mathbf{x}(t_1) = \boldsymbol{\theta}(t_1)$

Corner Example

- Find shortest length path joining the points $x = 0, t = -2$ and $x = 0, t = 1$ that touches the curve $x = t^2 + 3$ at some point

- In this case, $J = \int_{-2}^{1} \sqrt{1 + \dot{x}^2} dt$ with $x(1) = x(-2) = 0$ and $x(t_1) = t_1^2 + 3$

- Note that since g is only a function of $\dot{x}$, then solution $x(t)$ will only be linear in each segment (see 5–13)

$$\begin{aligned} \text{segment } 1 \quad & x(t) = a + bt \\ \text{segment } 2 \quad & x(t) = c + dt \end{aligned}$$

  – Terminal conditions: $x(-2) = a - 2b = 0$ and $x(1) = c + d = 0$

- Apply corner condition:

$$\sqrt{1 + \dot{x}(t_1^-)^2} + \frac{\dot{x}(t_1^-)}{\sqrt{1 + \dot{x}(t_1^-)^2}} \left[2t_1^- - \dot{x}(t_1^-) \right]$$
$$= \frac{1 + 2t_1^- \dot{x}(t_1^-)}{\sqrt{1 + \dot{x}(t_1^-)^2}} = \frac{1 + 2t_1^+ \dot{x}(t_1^+)}{\sqrt{1 + \dot{x}(t_1^+)^2}}$$

  which gives:

$$\frac{1 + 2bt_1}{\sqrt{1 + b^2}} = \frac{1 + 2dt_1}{\sqrt{1 + d^2}}$$

- Solve using `fsolve` to get:

$$a = 3.0947, b = 1.5474, c = 2.8362, d = -2.8362, t_1 = -0.0590$$

```
function F=myfunc(x); %
% x=[a b c d t1]; %
F=[x(1)-2*x(2);
    x(3)+x(4);
    (1+2*x(2)*x(5))/(1+x(2)^2)^(1/2) - (1+2*x(4)*x(5))/(1+x(4)^2)^(1/2);
    x(1)+x(2)*x(5) - (x(5)^2+3);
    x(3)+x(4)*x(5) - (x(5)^2+3)];
return %
x = fsolve('myfunc',[2 1 2 -2 0]')
```

Constrained Solutions

- Now consider variations of the basic problem that include constraints.
- For example, if the goal is to find the extremal function $\mathbf{x}^\star$ that minimizes

$$J(\mathbf{x}(t),t) = \int_{t_0}^{t_f} g(\mathbf{x}(t),\dot{\mathbf{x}}(t),t)dt$$

subject to the constraint that a given set of n differential equations be satisfied

$$\mathbf{f}(\mathbf{x}(t),\dot{\mathbf{x}}(t),t) = 0$$

where we assume that $\mathbf{x} \in \mathcal{R}^{n+m}$ (take t_f and $\mathbf{x}(t_f)$ to be fixed)

- As with the basic optimization problems in Lecture 2, proceed by augmenting cost with the constraints using Lagrange multipliers
  - Since the constraints must be satisfied at all time, these multipliers are also assumed to be functions of time.

$$J_a(\mathbf{x}(t),t) = \int_{t_0}^{t_f} \left\{ g(\mathbf{x},\dot{\mathbf{x}},t) + \mathbf{p}(t)^T \mathbf{f}(\mathbf{x},\dot{\mathbf{x}},t) \right\} dt$$

  - Does not change the cost if the constraints are satisfied.
  - Time varying Lagrange multipliers give more degrees of freedom in specifying how the constraints are added.

- Take variation of augmented functional considering perturbations to both $\mathbf{x}(t)$ and $\mathbf{p}(t)$

$$\begin{aligned}&\delta J(\mathbf{x}(t),\delta\mathbf{x}(t),\mathbf{p}(t),\delta\mathbf{p}(t))\\ &= \int_{t_0}^{t_f} \left\{ \left[g_{\mathbf{x}} + \mathbf{p}^T \mathbf{f}_{\mathbf{x}} \right] \delta\mathbf{x}(t) + \left[g_{\dot{\mathbf{x}}} + \mathbf{p}^T \mathbf{f}_{\dot{\mathbf{x}}} \right] \delta\dot{\mathbf{x}}(t) + \mathbf{f}^T \delta\mathbf{p}(t) \right\} dt\end{aligned}$$

- As before, integrate by parts to get:

$$\delta J(\mathbf{x}(t), \delta\mathbf{x}(t), \mathbf{p}(t), \delta\mathbf{p}(t)) = \int_{t_0}^{t_f} \left(\left\{ \left[g_{\mathbf{x}} + \mathbf{p}^T \mathbf{f}_{\mathbf{x}} \right] - \frac{d}{dt} \left[g_{\dot{\mathbf{x}}} + \mathbf{p}^T \mathbf{f}_{\dot{\mathbf{x}}} \right] \right\} \delta\mathbf{x}(t) + \mathbf{f}^T \delta\mathbf{p}(t) \right) dt$$

- To simplify things a bit, define

$$g_a(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) \equiv g(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) + \mathbf{p}(t)^T \mathbf{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)$$

- On the extremal, the variation must be zero, but since $\delta\mathbf{x}(t)$ and $\delta\mathbf{p}(t)$ can be arbitrary, can only occur if

$$\boxed{\begin{aligned} \frac{\partial g_a(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \mathbf{x}} - \frac{d}{dt}\left(\frac{\partial g_a(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{\mathbf{x}}} \right) &= 0 \\ \mathbf{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) &= 0 \end{aligned}}$$

  - which are obviously a generalized version of the Euler equations obtained before.

- Note similarity of the definition of g_a here with the Hamiltonian on page 4–4.

- Will find that this generalization carries over to future optimizations as well.

General Terminal Conditions

- Now consider Type 3 constraints on 5–10, which are a very general form with t_f free and $\mathbf{x}(t_f)$ given by a condition:

$$\mathbf{m}(\mathbf{x}(t_f), t_f) = 0$$

- Constrained optimization, so as before, augment the cost functional

$$J(\mathbf{x}(t), t) = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) dt$$

with the constraint using Lagrange multipliers:

$$J_a(\mathbf{x}(t), \boldsymbol{\nu}, t) = h(\mathbf{x}(t_f), t_f) + \boldsymbol{\nu}^T \mathbf{m}(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) dt$$

- Considering changes to $\mathbf{x}(t)$, t_f, $\mathbf{x}(t_f)$ and $\boldsymbol{\nu}$, the variation for J_a is

$$\begin{aligned}
\delta J_a &= h_{\mathbf{x}}(t_f)\delta\mathbf{x}_f + h_{t_f}\delta t_f + \mathbf{m}^T(t_f)\delta\boldsymbol{\nu} + \boldsymbol{\nu}^T\left(\mathbf{m}_{\mathbf{x}}(t_f)\delta\mathbf{x}_f + \mathbf{m}_{t_f}(t_f)\delta t_f\right) \\
&\quad + \int_{t_0}^{t_f} \left[g_{\mathbf{x}}\delta\mathbf{x} + g_{\dot{\mathbf{x}}}\delta\dot{\mathbf{x}}\right] dt + g(t_f)\delta t_f \\
&= \left[h_{\mathbf{x}}(t_f) + \boldsymbol{\nu}^T\mathbf{m}_{\mathbf{x}}(t_f)\right]\delta\mathbf{x}_f + \left[h_{t_f} + \boldsymbol{\nu}^T\mathbf{m}_{t_f}(t_f) + g(t_f)\right]\delta t_f \\
&\quad + \mathbf{m}^T(t_f)\delta\boldsymbol{\nu} + \int_{t_0}^{t_f}\left[g_{\mathbf{x}} - \frac{d}{dt}g_{\dot{\mathbf{x}}}\right]\delta\mathbf{x}dt + g_{\dot{\mathbf{x}}}(t_f)\delta\mathbf{x}(t_f)
\end{aligned}$$

  - Now use that $\delta\mathbf{x}_f = \delta\mathbf{x}(t_f) + \dot{\mathbf{x}}(t_f)\delta t_f$ as before to get

$$\begin{aligned}
\delta J_a &= \left[h_{\mathbf{x}}(t_f) + \boldsymbol{\nu}^T\mathbf{m}_{\mathbf{x}}(t_f) + g_{\dot{\mathbf{x}}}(t_f)\right]\delta\mathbf{x}_f \\
&\quad + \left[h_{t_f} + \boldsymbol{\nu}^T\mathbf{m}_{t_f}(t_f) + g(t_f) - g_{\dot{\mathbf{x}}}(t_f)\dot{\mathbf{x}}(t_f)\right]\delta t_f + \mathbf{m}^T(t_f)\delta\boldsymbol{\nu} \\
&\quad + \int_{t_0}^{t_f}\left[g_{\mathbf{x}} - \frac{d}{dt}g_{\dot{\mathbf{x}}}\right]\delta\mathbf{x}dt
\end{aligned}$$

- Looks like a bit of a mess, but we can clean it up a bit using

$$w(\mathbf{x}(t_f), \boldsymbol{\nu}, t_f) = h(\mathbf{x}(t_f), t_f) + \boldsymbol{\nu}^T \mathbf{m}(\mathbf{x}(t_f), t_f)$$

  to get

$$\begin{aligned}\delta J_a &= \left[w_{\mathbf{x}}(t_f) + g_{\dot{\mathbf{x}}}(t_f)\right] \delta \mathbf{x}_f \\ &+ \left[w_{t_f} + g(t_f) - g_{\dot{\mathbf{x}}}(t_f)\dot{\mathbf{x}}(t_f)\right] \delta t_f + \mathbf{m}^T(t_f)\delta\boldsymbol{\nu} \\ &+ \int_{t_0}^{t_f} \left[g_{\mathbf{x}} - \frac{d}{dt} g_{\dot{\mathbf{x}}}\right] \delta \mathbf{x} dt\end{aligned}$$

  – Given the extra degrees of freedom in the multipliers, can treat all of the variations as independent $\Rightarrow$ all coefficients must be zero to achieve $\delta J_a = 0$

- So the necessary conditions are

$$\boxed{\begin{aligned} g_{\mathbf{x}} - \frac{d}{dt} g_{\dot{\mathbf{x}}} &= 0 \quad (\texttt{dim}\ n) \\ w_{\mathbf{x}}(t_f) + g_{\dot{\mathbf{x}}}(t_f) &= 0 \quad (\texttt{dim}\ n) \\ w_{t_f} + g(t_f) - g_{\dot{\mathbf{x}}}(t_f)\dot{\mathbf{x}}(t_f) &= 0 \quad (\texttt{dim}\ 1) \end{aligned}}$$

  – With $\mathbf{x}(t_0) = \mathbf{x}_0$ ($\texttt{dim}\ n$) and $\mathbf{m}(\mathbf{x}(t_f), t_f) = 0$ ($\texttt{dim}\ m$) combined with last 2 conditions $\Rightarrow$ $2n + m + 1$ constraints

  – Solution of Eulers equation has $2n$ constants of integration for $x(t)$, and must find $\boldsymbol{\nu}$ ($\texttt{dim}\ m$) and t_f $\Rightarrow$ $2n + m + 1$ unknowns

- Some special cases:

  – If t_f is fixed, $h = h(\mathbf{x}(t_f))$, $\mathbf{m} \rightarrow \mathbf{m}(\mathbf{x}(t_f))$ and we lose the last condition in box – others remain unchanged

  – If t_f is fixed, $\mathbf{x}(t_f)$ free, then there is no $\mathbf{m}$, no $\boldsymbol{\nu}$ and w reduces to h.

- Kirk's book also considers several other type of constraints.

16.323 Lecture 6

Calculus of Variations applied to Optimal Control

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \\ \dot{\mathbf{p}} &= -H_{\mathbf{x}}^T \\ H_{\mathbf{u}} &= 0 \end{aligned}$$

Optimal Control Problems

- Are now ready to tackle the optimal control problem
  - Start with simple terminal constraints

$$J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt$$

    with the system dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)$$

  - t_0, $\mathbf{x}(t_0)$ fixed
  - t_f free
  - $\mathbf{x}(t_f)$ are fixed or free by element
- Note that this looks a bit different because we have $\mathbf{u}(t)$ in the integrand, but consider that with a simple substitution, we get

$$\tilde{g}(\mathbf{x}, \dot{\mathbf{x}}, t) \overset{\dot{\mathbf{x}}=\mathbf{a}(\mathbf{x},\mathbf{u},t)}{\longrightarrow} \hat{g}(\mathbf{x}, \mathbf{u}, t)$$

- Note that the differential equation of the dynamics acts as a constraint that we must adjoin using a Lagrange multiplier, as before:

$$J_a = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \left[g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T \{\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) - \dot{\mathbf{x}}\} \right] dt$$

- Find the variation:[10]

$$\begin{aligned}\delta J_a = \; & h_{\mathbf{x}} \delta \mathbf{x}_f + h_{t_f} \delta t_f + \int_{t_0}^{t_f} \left[g_{\mathbf{x}} \delta \mathbf{x} + g_{\mathbf{u}} \delta \mathbf{u} + (\mathbf{a} - \dot{\mathbf{x}})^T \delta \mathbf{p}(t) \right. \\ & \left. + \mathbf{p}^T(t) \{\mathbf{a}_{\mathbf{x}} \delta \mathbf{x} + \mathbf{a}_{\mathbf{u}} \delta \mathbf{u} - \delta \dot{\mathbf{x}}\} \right] dt + \left[g + \mathbf{p}^T (\mathbf{a} - \dot{\mathbf{x}}) \right] (t_f) \delta t_f\end{aligned}$$

- Clean this up by defining the **Hamiltonian**: (See 4–4)

$$H(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) = g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t) \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)$$

[10] Take partials wrt each of the variables that the integrand is a function of.

- Then

$$\begin{aligned}\delta J_a &= h_{\mathbf{x}}\delta\mathbf{x}_f + \left[h_{t_f} + g + \mathbf{p}^T(\mathbf{a}-\dot{\mathbf{x}})\right](t_f)\delta t_f \\ &\quad + \int_{t_0}^{t_f}\left[H_{\mathbf{x}}\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (\mathbf{a}-\dot{\mathbf{x}})^T\delta\mathbf{p}(t) - \mathbf{p}^T(t)\delta\dot{\mathbf{x}}\right]dt\end{aligned}$$

- To proceed, note that by integrating by parts [11] we get:

$$\begin{aligned}-\int_{t_0}^{t_f}\mathbf{p}^T(t)\delta\dot{\mathbf{x}}dt &= -\int_{t_0}^{t_f}\mathbf{p}^T(t)d\delta\mathbf{x} \\ &= -\mathbf{p}^T\delta\mathbf{x}\Big|_{t_0}^{t_f} + \int_{t_0}^{t_f}\left(\frac{d\mathbf{p}(t)}{dt}\right)^T\delta\mathbf{x}dt \\ &= -\mathbf{p}^T(t_f)\delta\mathbf{x}(t_f) + \int_{t_0}^{t_f}\dot{\mathbf{p}}^T(t)\delta\mathbf{x}dt \\ &= -\mathbf{p}^T(t_f)\left(\delta\mathbf{x}_f - \dot{\mathbf{x}}(t_f)\delta t_f\right) + \int_{t_0}^{t_f}\dot{\mathbf{p}}^T(t)\delta\mathbf{x}dt\end{aligned}$$

- So now can rewrite the variation as:

$$\begin{aligned}\delta J_a &= h_{\mathbf{x}}\delta\mathbf{x}_f + \left[h_{t_f} + g + \mathbf{p}^T(\mathbf{a}-\dot{\mathbf{x}})\right](t_f)\delta t_f \\ &\quad + \int_{t_0}^{t_f}\left[H_{\mathbf{x}}\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (\mathbf{a}-\dot{\mathbf{x}})^T\delta\mathbf{p}(t)\right]dt - \int_{t_0}^{t_f}\mathbf{p}^T(t)\delta\dot{\mathbf{x}}dt \\ &= \left(h_{\mathbf{x}} - \mathbf{p}^T(t_f)\right)\delta\mathbf{x}_f + \left[h_{t_f} + g + \mathbf{p}^T(\mathbf{a}-\dot{\mathbf{x}}) + \mathbf{p}^T\dot{\mathbf{x}}\right](t_f)\delta t_f \\ &\quad + \int_{t_0}^{t_f}\left[\left(H_{\mathbf{x}} + \dot{\mathbf{p}}^T\right)\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (\mathbf{a}-\dot{\mathbf{x}})^T\delta\mathbf{p}(t)\right]dt\end{aligned}$$

[11] $\int udv \equiv uv - \int vdu$

- So necessary conditions for $\delta J_a = 0$ are that for $t \in [t_0, t_f]$

$$\begin{array}{rcll} \dot{\mathbf{x}} & = & \mathbf{a}(\mathbf{x}, \mathbf{u}, t) & (\texttt{dim } n) \\ \dot{\mathbf{p}} & = & -H_{\mathbf{x}}^T & (\texttt{dim } n) \\ H_{\mathbf{u}} & = & 0 & (\texttt{dim } m) \end{array}$$

  - With the boundary condition (lost if t_f is fixed) that

$$h_{t_f} + g + \mathbf{p}^T \mathbf{a} = h_{t_f} + H(t_f) = 0$$

  - Add the boundary constraints that $\mathbf{x}(t_0) = \mathbf{x}_0$ (dim n)
  - If $\mathbf{x}_i(t_f)$ is fixed, then $\mathbf{x}_i(t_f) = x_{i_f}$
  - If $\mathbf{x}_i(t_f)$ is free, then $\mathbf{p}_i(t_f) = \dfrac{\partial h}{\partial x_i}(t_f)$ for a total (dim n)

- These necessary conditions have $2n$ differential and m algebraic equations with $2n+1$ unknowns (if t_f free), found by imposing the $(2n+1)$ boundary conditions.

- Note the symmetry in the differential equations:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x},\mathbf{u},t) = \left(\frac{\partial H}{\partial \mathbf{p}}\right)^T \\ \dot{\mathbf{p}} &= -\left(\frac{\partial H}{\partial \mathbf{x}}\right)^T = -\frac{\partial (g+\mathbf{p}^T\mathbf{a})^T}{\partial \mathbf{x}} \\ &= -\left(\frac{\partial \mathbf{a}}{\partial \mathbf{x}}\right)^T \mathbf{p} - \left(\frac{\partial g}{\partial \mathbf{x}}\right)^T\end{aligned}$$

  - So the dynamics of $\mathbf{p}$, called the **costate**, are **linearized system dynamics** (negative transpose – dual)

$$\left(\frac{\partial \mathbf{a}}{\partial \mathbf{x}}\right) = \begin{bmatrix} \frac{\partial a_1}{\partial x_1} & \cdots & \frac{\partial a_1}{\partial x_n} \\ & \ddots & \\ \frac{\partial a_n}{\partial x_1} & \cdots & \frac{\partial a_n}{\partial x_n} \end{bmatrix}$$

- These necessary conditions are extremely important, and we will be using them for the rest of the term.

Control with General Terminal Conditions

- Can develop similar conditions in the case of more general terminal conditions with t_f free and

$$\mathbf{m}(\mathbf{x}(t_f), t_f) = 0$$

- Follow the same procedure on 6–1 using the insights provided on 5–21 (using the g_a form on 5–20) to form

$$w(\mathbf{x}(t_f), \boldsymbol{\nu}, t_f) = h(\mathbf{x}(t_f), t_f) + \boldsymbol{\nu}^T \mathbf{m}(\mathbf{x}(t_f), t_f)$$

- Work through the math, and get the necessary conditions are

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \quad (\texttt{dim } n) \quad (6.22) \\ \dot{\mathbf{p}} &= -H_{\mathbf{x}}^T \quad (\texttt{dim } n) \quad (6.23) \\ H_{\mathbf{u}} &= 0 \quad (\texttt{dim } m) \quad (6.24) \end{aligned}$$

  - With the boundary condition (lost if t_f fixed)

$$H(t_f) + w_{t_f}(t_f) = 0$$

  - And $\mathbf{m}(\mathbf{x}(t_f), t_f) = 0$, with $\mathbf{x}(t_0)$ and t_0 given.
  - With (since $\mathbf{x}(t_f)$ is not directly given)

$$\mathbf{p}(t_f) = \left[\frac{\partial w}{\partial \mathbf{x}}(t_f)\right]^T$$

- Collapses to form on 6–3 if $\mathbf{m}$ not present – i.e., does not constrain $\mathbf{x}(t_f)$

Example 6–1

- Simple double integrator system starting at $y(0) = 10$, $\dot{y}(0) = 0$, must drive to origin $y(t_f) = \dot{y}(t_f) = 0$ to minimize the cost $(b > 0)$

$$J = \frac{1}{2}\alpha t_f^2 + \frac{1}{2}\int_0^{t_f} bu^2(t)dt$$

- Define the dynamics with $x_1 = y$, $x_2 = \dot{y}$ so that

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t) \qquad A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- With $\mathbf{p}(t) = [p_1(t) \;\; p_2(t)]^T$, define the Hamiltonian

$$H = g + \mathbf{p}^T(t)\mathbf{a} = \frac{1}{2}bu^2 + \mathbf{p}^T(t)\left(A\mathbf{x}(t) + Bu(t)\right)$$

- The necessary conditions are then that:

$$\begin{aligned} \dot{\mathbf{p}} &= -H_{\mathbf{x}}^T, \quad \rightarrow & \dot{p}_1 &= -\frac{\partial H}{\partial x_1} = 0 \rightarrow p_1(t) = c_1 \\ & & \dot{p}_2 &= -\frac{\partial H}{\partial x_2} = -p_1 \rightarrow p_2(t) = -c_1 t + c_2 \\ H_u &= bu + p_2 = 0 \quad \rightarrow & u &= -\frac{p_2}{b} = -\frac{c_2}{b} + \frac{c_1}{b}t \end{aligned}$$

- Now impose the boundary conditions:

$$\begin{aligned} H(t_f) + h_t(t_f) &= \frac{1}{2}bu^2(t_f) + p_1(t_f)x_2(t_f) + p_2(t_f)u(t_f) + \alpha t_f = 0 \\ &= \frac{1}{2}bu^2(t_f) + (-bu(t_f))u(t_f) + \alpha t_f \\ &= -\frac{1}{2}bu^2(t_f) + \alpha t_f = 0 \;\rightarrow\; t_f = \frac{1}{2b\alpha}\left(-c_2 + c_1 t_f\right)^2 \end{aligned}$$

- Now go back to the state equations:

$$\dot{x}_2(t) = -\frac{c_2}{b} + \frac{c_1}{b}t \quad \rightarrow \quad x_2(t) = c_3 - \frac{c_2}{b}t + \frac{c_1}{2b}t^2$$

and since $x_2(0) = 0$, $c_3 = 0$, and

$$\dot{x}_1(t) = x_2(t) \quad \rightarrow \quad x_1(t) = c_4 - \frac{c_2}{2b}t^2 + \frac{c_1}{6b}t^3$$

and since $x_1(0) = 10$, $c_4 = 10$

- Now note that

$$\begin{aligned} x_2(t_f) &= -\frac{c_2}{b}t_f + \frac{c_1}{2b}t_f^2 = 0 \\ x_1(t_f) &= 10 - \frac{c_2}{2b}t_f^2 + \frac{c_1}{6b}t_f^3 = 0 \\ &= 10 - \frac{c_2}{6b}t_f^2 = 0 \quad \rightarrow \quad c_2 = \frac{60b}{t_f^2}, \qquad c_1 = \frac{120b}{t_f^3} \end{aligned}$$

  - But that gives us:

$$t_f = \frac{1}{2b\alpha}\left(-\frac{60b}{t_f^2} + \frac{120b}{t_f^3}t_f\right)^2 = \frac{(60b)^2}{2b\alpha t_f^4}$$

    so that $t_f^5 = 1800b/\alpha$ or $t_f \approx 4.48(b/\alpha)^{1/5}$, which makes sense because t_f goes down as α goes up.

  - Finally, $c_2 = 2.99b^{3/5}\alpha^{2/5}$ and $c_1 = 1.33b^{2/5}\alpha^{3/5}$

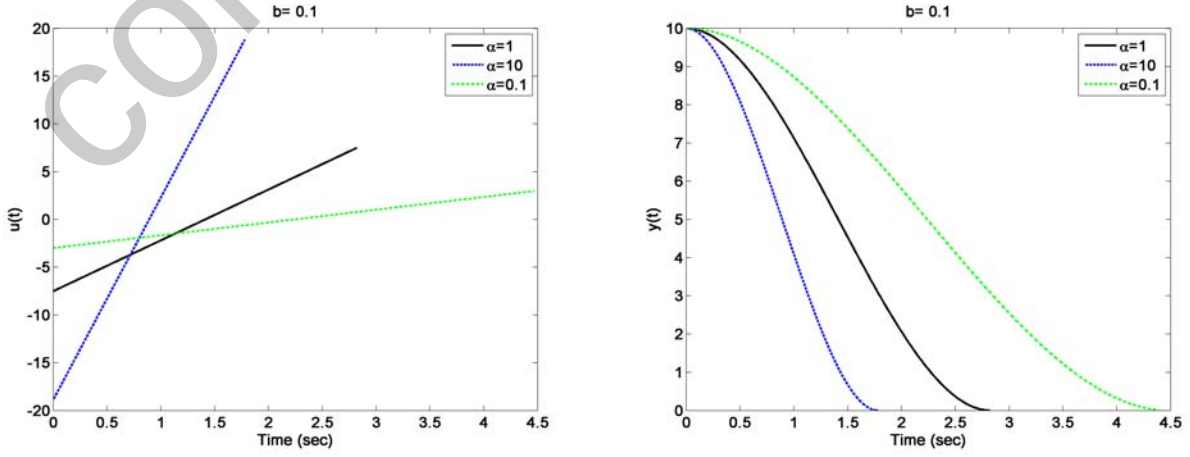


Figure 6.1: Example 6–1

Example 6–1

```
%
% Simple opt example showing impact of weight on t_f
% 16.323 Spring 2008
% Jonathan How
% opt1.m
%
clear all;close all;
set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontWeight','demi')
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontWeight','demi')
%
A=[0 1;0 0];B=[0 1]';C=eye(2);D=zeros(2,1);
G=ss(A,B,C,D);
X0=[10 0]';
b=0.1;

alp=1;
tf=(1800*b/alp)^0.2;
c1=120*b/tf^3;
c2=60*b/tf^2;
time=[0:1e-2:tf];
u=(-c2+c1*time)/b;
[y1,t1]=lsim(G,u,time,X0);

figure(1);clg
plot(time,u,'k-','LineWidth',2);hold on
alp=10;
tf=(1800*b/alp)^0.2;
c1=120*b/tf^3;
c2=60*b/tf^2;
time=[0:1e-2:tf];
u=(-c2+c1*time)/b;
[y2,t2]=lsim(G,u,time,X0);
plot(time,u,'b--','LineWidth',2);

alp=0.10;
tf=(1800*b/alp)^0.2;
c1=120*b/tf^3;
c2=60*b/tf^2;
time=[0:1e-2:tf];
u=(-c2+c1*time)/b;
[y3,t3]=lsim(G,u,time,X0);
plot(time,u,'g-.','LineWidth',2);hold off

legend('\alpha=1','\alpha=10','\alpha=0.1')
xlabel('Time (sec)')
ylabel('u(t)')
title(['b= ',num2str(b)])

figure(2);clg
plot(t1,y1(:,1),'k-','LineWidth',2);
hold on
plot(t2,y2(:,1),'b--','LineWidth',2);
plot(t3,y3(:,1),'g-.','LineWidth',2);
hold off
legend('\alpha=1','\alpha=10','\alpha=0.1')
xlabel('Time (sec)')
ylabel('y(t)')
title(['b= ',num2str(b)])

print -dpng -r300 -f1 opt11.png
print -dpng -r300 -f2 opt12.png
```

LQR Variational Solution

- Deterministic Linear Quadratic Regulator

  Plant:

  $$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B_u(t)\mathbf{u}(t), \qquad \mathbf{x}(t_0) = \mathbf{x}_0$$
  $$\mathbf{z}(t) = C_z(t)\mathbf{x}(t)$$

  Cost:

  $$2J_{LQR} = \int_{t_0}^{t_f} \left[\mathbf{z}^T(t)R_{zz}(t)\mathbf{z}(t) + \mathbf{u}^T(t)R_{uu}(t)\mathbf{u}(t)\right]\, dt + \mathbf{x}(t_f)^T P_{t_f}\mathbf{x}(t_f)$$

  - Where $P_{t_f} \geq 0$, $R_{zz}(t) > 0$ and $R_{uu}(t) > 0$
  - Define $R_{xx} = C_z^T R_{zz} C_z \geq 0$
  - $A(t)$ is a continuous function of time.
  - $B_u(t)$, $C_z(t)$, $R_{zz}(t)$, $R_{uu}(t)$ are piecewise continuous functions of time, and all are bounded.

- **Problem Statement:** Find input $u(t)\ \forall t \in [t_0, t_f]$ to min J_{LQR}
  - This is not necessarily specified to be a feedback controller.

- To optimize the cost, we follow the procedure of augmenting the constraints in the problem (the system dynamics) to the cost (integrand) to form the **Hamiltonian**:

  $$H = \frac{1}{2}\left(\mathbf{x}^T(t)R_{xx}\mathbf{x}(t) + \mathbf{u}^T(t)R_{uu}\mathbf{u}(t)\right) + \mathbf{p}^T(t)\left(A\mathbf{x}(t) + B_u\mathbf{u}(t)\right)$$

  - $\mathbf{p}(t) \in \mathbb{R}^{n\times 1}$ is called the **Adjoint variable** or **Costate**
  - It is the **Lagrange multiplier** in the problem.

- The necessary conditions (see 6–3) for optimality are that:

  1. $\dot{\mathbf{x}}(t) = \dfrac{\partial H}{\partial \mathbf{p}}^T = A\mathbf{x}(t) + B(t)\mathbf{u}(t)$ with $\mathbf{x}(t_0) = \mathbf{x}_0$

  2. $\dot{\mathbf{p}}(t) = -\dfrac{\partial H}{\partial \mathbf{x}}^T = -R_{\rm xx}\mathbf{x}(t) - A^T\mathbf{p}(t)$ with $\mathbf{p}(t_f) = P_{t_f}\mathbf{x}(t_f)$

  3. $\dfrac{\partial H}{\partial \mathbf{u}} = 0 \Rightarrow R_{\rm uu}\mathbf{u} + B_u^T\mathbf{p}(t) = 0$, so $\mathbf{u}^\star = -R_{\rm uu}^{-1}B_u^T\mathbf{p}(t)$

  4. As before, we can check for a minimum by looking at $\dfrac{\partial^2 H}{\partial \mathbf{u}^2} \geq 0$ (need to check that $R_{\rm uu} \geq 0$)

- Note that $\mathbf{p}(t)$ plays the same role as $J_{\mathbf{x}}^\star(\mathbf{x}(t),t)^T$ in previous solutions to the continuous LQR problem (see 4–8).
  - Main difference is there is no need to guess a solution for $J^\star(\mathbf{x}(t),t)$

- Now have:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}^\star(t) = A\mathbf{x}(t) - B_u R_{\rm uu}^{-1}B_u^T\mathbf{p}(t)$$

  which can be combined with equation for the adjoint variable

$$\dot{\mathbf{p}}(t) = -R_{\rm xx}\mathbf{x}(t) - A^T\mathbf{p}(t) = -C_z^T R_{\rm zz}C_z\mathbf{x}(t) - A^T\mathbf{p}(t)$$

$$\Rightarrow \begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} A & -B_u R_{\rm uu}^{-1}B_u^T \\ -C_z^T R_{\rm zz}C_z & -A^T \end{bmatrix}}_{H} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{p}(t) \end{bmatrix}$$

  where H is called the **Hamiltonian Matrix**.
  - Matrix describes coupled closed loop dynamics for both $\mathbf{x}$ and $\mathbf{p}$.
  - Dynamics of $\mathbf{x}(t)$ and $\mathbf{p}(t)$ are coupled, but $\mathbf{x}(t)$ known initially and $\mathbf{p}(t)$ known at terminal time, since $\mathbf{p}(t_f) = P_{t_f}\mathbf{x}(t_f)$
  - Two point boundary value problem $\Rightarrow$ typically hard to solve.

- However, in this case, we can introduce a new matrix variable $P(t)$ and show that:
  1. $\mathbf{p}(t) = P(t)\mathbf{x}(t)$
  2. It is relatively easy to find $P(t)$.

- How proceed?
  1. For the $2n$ system

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix} = \begin{bmatrix} A & -B_u R_{\text{uu}}^{-1} B_u^T \\ -C_z^T R_{\text{zz}} C_z & -A^T \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{p}(t) \end{bmatrix}$$

  define a transition matrix

$$F(t_1, t_0) = \begin{bmatrix} F_{11}(t_1, t_0) & F_{12}(t_1, t_0) \\ F_{21}(t_1, t_0) & F_{22}(t_1, t_0) \end{bmatrix}$$

  and use this to relate $\mathbf{x}(t)$ to $\mathbf{x}(t_f)$ and $\mathbf{p}(t_f)$

$$\begin{bmatrix} \mathbf{x}(t) \\ \mathbf{p}(t) \end{bmatrix} = \begin{bmatrix} F_{11}(t, t_f) & F_{12}(t, t_f) \\ F_{21}(t, t_f) & F_{22}(t, t_f) \end{bmatrix} \begin{bmatrix} \mathbf{x}(t_f) \\ \mathbf{p}(t_f) \end{bmatrix}$$

  so

$$\begin{aligned} \mathbf{x}(t) &= F_{11}(t, t_f)\mathbf{x}(t_f) + F_{12}(t, t_f)\mathbf{p}(t_f) \\ &= \left[F_{11}(t, t_f) + F_{12}(t, t_f)P_{t_f}\right] \mathbf{x}(t_f) \end{aligned}$$

  2. Now find $\mathbf{p}(t)$ in terms of $\mathbf{x}(t_f)$

$$\mathbf{p}(t) = \left[F_{21}(t, t_f) + F_{22}(t, t_f)P_{t_f}\right] \mathbf{x}(t_f)$$

  3. Eliminate $\mathbf{x}(t_f)$ to get:

$$\begin{aligned} \mathbf{p}(t) &= \left[F_{21}(t, t_f) + F_{22}(t, t_f)P_{t_f}\right] \left[F_{11}(t, t_f) + F_{12}(t, t_f)P_{t_f}\right]^{-1} \mathbf{x}(t) \\ &\triangleq P(t)\mathbf{x}(t) \end{aligned}$$

- Now have $\mathbf{p}(t) = P(t)\mathbf{x}(t)$, must find the equation for $P(t)$

$$\begin{aligned} \dot{\mathbf{p}}(t) &= \dot{P}(t)\mathbf{x}(t) + P(t)\dot{\mathbf{x}}(t) \\ \Rightarrow \quad -C_z^T R_{zz} C_z \mathbf{x}(t) - A^T \mathbf{p}(t) &= \end{aligned}$$

$$\begin{aligned} -\dot{P}(t)\mathbf{x}(t) &= C_z^T R_{zz} C_z \mathbf{x}(t) + A^T \mathbf{p}(t) + P(t)\dot{\mathbf{x}}(t) \\ &= C_z^T R_{zz} C_z \mathbf{x}(t) + A^T \mathbf{p}(t) + P(t)(A\mathbf{x}(t) - B_u R_{uu}^{-1} B_u^T \mathbf{p}(t)) \\ &= (C_z^T R_{zz} C_z + P(t)A)\mathbf{x}(t) + (A^T - P(t)B_u R_{uu}^{-1} B_u^T)\mathbf{p}(t) \\ &= (C_z^T R_{zz} C_z + P(t)A)\mathbf{x}(t) + (A^T - P(t)B_u R_{uu}^{-1} B_u^T)P(t)\mathbf{x}(t) \\ &= \left[A^T P(t) + P(t)A + C_z^T R_{zz} C_z - P(t)B_u R_{uu}^{-1} B_u^T P(t)\right]\mathbf{x}(t) \end{aligned}$$

- This must be true for arbitrary $\mathbf{x}(t)$, so $P(t)$ must satisfy

$$-\dot{P}(t) = A^T P(t) + P(t)A + C_z^T R_{zz} C_z - P(t)B_u R_{uu}^{-1} B_u^T P(t)$$

  - Which, of course, is the matrix differential **Riccati Equation**.
  - Optimal value of $P(t)$ is found by solving this equation *backwards* in time from t_f with $P(t_f) = P_{t_f}$

- The control gains are then

$$u_{\text{opt}} = -R_{\text{uu}}^{-1} B_u^T \mathbf{p}(t) = -R_{\text{uu}}^{-1} B_u^T P(t)\mathbf{x}(t) = -K(t)\mathbf{x}(t)$$

- **Optimal control inputs can in fact be computed using linear feedback on the full system state**
  - Find optimal steady state feedback gains $\mathbf{u}(t) = -K\mathbf{x}(t)$ using

$$K = \texttt{lqr}(A, B, C_z^T R_{\text{zz}} C_z, R_{\text{uu}})$$

- **Key point:** This controller works equally well for MISO and MIMO regulator designs.

Alternate Derivation of DRE

- On 6-10 we showed that:

$$P(t) = \left[F_{21}(t,t_f) + F_{22}(t,t_f)P_{t_f}\right]\left[F_{11}(t,t_f) + F_{12}(t,t_f)P_{t_f}\right]^{-1}$$

- To find the Riccati equation, note that

$$\frac{d}{dt}M^{-1}(t) = -M^{-1}(t)\dot{M}(t)M^{-1}(t)$$

which gives

$$\begin{aligned}\dot{P}(t) = &\left[\dot{F}_{21}(t,t_f) + \dot{F}_{22}(t,t_f)P_{t_f}\right]\left[F_{11}(t,t_f) + F_{12}(t,t_f)P_{t_f}\right]^{-1} \\ &- \left[F_{21}(t,t_f) + F_{22}(t,t_f)P_{t_f}\right]\left[F_{11}(t,t_f) + F_{12}(t,t_f)P_{t_f}\right]^{-1} \cdot \\ &\left[\dot{F}_{11}(t,t_f) + \dot{F}_{12}(t,t_f)P_{t_f}\right]\left[F_{11}(t,t_f) + F_{12}(t,t_f)P_{t_f}\right]^{-1}\end{aligned}$$

- Since F is the transition matrix [12] for the system (see 6–10), then

$$\frac{d}{dt}F(t,t_f) = HF(t,t_f)$$

$$\begin{bmatrix}\dot{F}_{11} & \dot{F}_{12} \\ \dot{F}_{21} & \dot{F}_{22}\end{bmatrix}(t,t_f) = \begin{bmatrix}A & -B_u R_{\text{uu}}^{-1} B_u^T \\ -R_{\text{xx}} & -A^T\end{bmatrix}(t,t_f)\begin{bmatrix}F_{11} & F_{12} \\ F_{21} & F_{22}\end{bmatrix}(t,t_f)$$

[12] Consider homogeneous system $\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t)$ with initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$. The general solution to this differential equation is given by $\mathbf{x}(t) = \Phi(t,t_0)\mathbf{x}(t_0)$ where $\Phi(t_1,t_1) = I$. Can show the following properties of the state transition matrix Φ:

1. $\Phi(t_2,t_0) = \Phi(t_2,t_1)\Phi(t_1,t_0)$, regardless of the order of the t_i
2. $\Phi(t,\tau) = \Phi(\tau,t)^{-1}$
3. $\frac{d}{dt}\Phi(t,t_0) = A(t)\Phi(t,t_0)$

- Now substitute and re-arrange:

$$\begin{aligned}
\dot{P} &= \left\{[\dot{F}_{21} + \dot{F}_{22}P_{t_f}] - P[\dot{F}_{11} + \dot{F}_{12}P_{t_f}]\right\}[F_{11} + F_{12}P_{t_f}]^{-1} \\
\dot{F}_{11} &= AF_{11} - B_u R_{\rm uu}^{-1} B_u^T F_{21} \\
\dot{F}_{12} &= AF_{12} - B_u R_{\rm uu}^{-1} B_u^T F_{22} \\
\dot{F}_{21} &= -R_{\rm xx}F_{11} - A^T F_{21} \\
\dot{F}_{22} &= -R_{\rm xx}F_{12} - A^T F_{22}
\end{aligned}$$

$$\begin{aligned}
\dot{P} &= \left\{\left(-R_{\rm xx}F_{11} - A^T F_{21} + (-R_{\rm xx}F_{12} - A^T F_{22})P_{t_f}\right)\right. \\
&\left. -P\left(AF_{11} - B_u R_{\rm uu}^{-1} B_u^T F_{21} + (AF_{12} - B_u R_{\rm uu}^{-1} B_u^T F_{22})P_{t_f}\right)\right\}[F_{11} + F_{12}P_{t_f}]^{-1}
\end{aligned}$$

- There are four terms:

$$-R_{\rm xx}(F_{11} + F_{12}P_{t_f})[F_{11} + F_{12}P_{t_f}]^{-1} = -R_{\rm xx}$$

$$-A^T(F_{21} + F_{22}P_{t_f})[F_{11} + F_{12}P_{t_f}]^{-1} = -A^T P$$

$$-PA(F_{11} + F_{12}P_{t_f})[F_{11} + F_{12}P_{t_f}]^{-1} = -PA$$

$$PB_u R_{\rm uu}^{-1} B_u^T (F_{21} + F_{22}P_{t_f})[F_{11} + F_{12}P_{t_f}]^{-1} = PB_u R_{\rm uu}^{-1} B_u^T P$$

- Which, as expected, gives that

$$-\dot{P} = A^T P + PA + R_{\rm xx} - PB_u R_{\rm uu}^{-1} B_u^T P$$

CARE Solution Algorithm

- Recall from (6–10) that

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix} = \begin{bmatrix} A & -B_u R_{\text{uu}}^{-1} B_u^T \\ -C_z^T R_{\text{zz}} C_z & -A^T \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{p}(t) \end{bmatrix}$$

- Assuming that the eigenvalues of H are unique, the Hamiltonian can be diagonalized into the form:

$$\begin{bmatrix} \dot{\mathbf{z}}_1(t) \\ \dot{\mathbf{z}}_2(t) \end{bmatrix} = \begin{bmatrix} -\Lambda & 0 \\ 0 & \Lambda \end{bmatrix} \begin{bmatrix} \mathbf{z}_1(t) \\ \mathbf{z}_2(t) \end{bmatrix}$$

  where diagonal matrix Λ is comprised of RHP eigenvalues of H.

- A similarity transformation exists between the states $\mathbf{z}_1, \mathbf{z}_2$ and $\mathbf{x}, \mathbf{p}$:

$$\begin{bmatrix} \mathbf{x}(t) \\ \mathbf{p}(t) \end{bmatrix} = \Psi \begin{bmatrix} \mathbf{z}_1(t) \\ \mathbf{z}_2(t) \end{bmatrix} \Leftrightarrow \begin{bmatrix} \mathbf{z}_1(t) \\ \mathbf{z}_2(t) \end{bmatrix} = \Psi^{-1} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{p}(t) \end{bmatrix}$$

  where

$$\Psi = \left[\begin{array}{c|c} \Psi_{11} & \Psi_{12} \\ \hline \Psi_{21} & \Psi_{22} \end{array}\right] \text{ and } \Psi^{-1} = \left[\begin{array}{c|c} (\Psi^{-1})_{11} & (\Psi^{-1})_{12} \\ \hline (\Psi^{-1})_{21} & (\Psi^{-1})_{22} \end{array}\right]$$

  and the columns of Ψ are the eigenvectors of H.

- Solving for $\mathbf{z}_2(t)$ gives

$$\begin{aligned} \mathbf{z}_2(t) = e^{\Lambda t}\mathbf{z}_2(0) &= [(\Psi^{-1})_{21}\mathbf{x}(t) + (\Psi^{-1})_{22}\mathbf{p}(t)] \\ &= [(\Psi^{-1})_{21} + (\Psi^{-1})_{22}P(t)]\mathbf{x}(t) \end{aligned}$$

  – For the cost to be finite, need $\lim_{t\to\infty} \mathbf{x}(t) = 0$, so can show that

$$\lim_{t\to\infty} \mathbf{z}_2(t) = 0$$

  – But given that the Λ dynamics in the RHP, this can only be true if $\mathbf{z}_2(0) = 0$, which means that $\mathbf{z}_2(t) = 0\ \forall t$

- With this fact, note that

$$\begin{aligned}\mathbf{x}(t) &= \Psi_{11}\mathbf{z}_1(t)\\ \mathbf{p}(t) &= \Psi_{21}\mathbf{z}_1(t)\end{aligned}$$

  which can be combined to give:

$$\mathbf{p}(t) = \Psi_{21}(\Psi_{11})^{-1}\mathbf{x}(t) \equiv P_{ss}\mathbf{x}(t)$$

- Summary of solution algorithm:
  - Find the eigenvalues and eigenvectors of H
  - Select the n eigenvectors associated with the n eigenvalues in the LHP.
  - Form Ψ_{11} and Ψ_{21}.
  - Compute the steady state solution of the Riccati equation using

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

```
% alternative calc of Riccati solution
H=[A -B*inv(Ruu)*B' ; -Rxx -A'];
[V,D]=eig(H); % check order of eigenvalues
Psi11=V(1:2,1:2);
Psi21=V(3:4,1:2);
Ptest=Psi21*inv(Psi11);
```

Optimal Cost

- Showed in earlier derivations that the optimal cost-to-go from the initial (or any state) is of the form

$$J = \frac{1}{2}\mathbf{x}^T(t_0)P(t_0)\mathbf{x}(t_0)$$

  – Relatively clean way to show it for this derivation as well.

- Start with the standard cost and add zero ($A\mathbf{x} + B_u\mathbf{u} - \dot{\mathbf{x}} = 0$)

$$\begin{aligned} J_{LQR} &= \frac{1}{2}\int_{t_0}^{t_f} \left[\mathbf{x}^T R_{\rm xx}\mathbf{x} + \mathbf{u}^T R_{\rm uu}\mathbf{u} + \mathbf{p}^T(A\mathbf{x} + B_u\mathbf{u} - \dot{\mathbf{x}})\right]\, dt \\ &\quad + \frac{1}{2}\mathbf{x}(t_f)^T P_{t_f}\mathbf{x}(t_f) \end{aligned}$$

- Now use the results of the necessary conditions to get:

$$\begin{aligned} \dot{\mathbf{p}} = -H_{\mathbf{x}}^T \qquad &\Rightarrow \mathbf{p}^T A = -\dot{\mathbf{p}}^T - \mathbf{x}^T R_{\rm xx} \\ H_{\mathbf{u}} = 0 \qquad &\Rightarrow \mathbf{p}^T B_u = -\mathbf{u}^T R_{\rm uu} \end{aligned}$$

  with $\mathbf{p}(t_f) = P_{t_f}\mathbf{x}(t_f)$

- Substitute these terms to get

$$\begin{aligned} J_{LQR} &= \frac{1}{2}\mathbf{x}(t_f)^T P_{t_f}\mathbf{x}(t_f) - \frac{1}{2}\int_{t_0}^{t_f}\left[\dot{\mathbf{p}}^T\mathbf{x} + \mathbf{p}^T\dot{\mathbf{x}}\right]\, dt \\ &= \frac{1}{2}\mathbf{x}(t_f)^T P_{t_f}\mathbf{x}(t_f) - \frac{1}{2}\int_{t_0}^{t_f}\left[\frac{d}{dt}(\mathbf{p}^T\mathbf{x})\right]\, dt \\ &= \frac{1}{2}\mathbf{x}(t_f)^T P_{t_f}\mathbf{x}(t_f) - \frac{1}{2}\left[\mathbf{p}^T(t_f)\mathbf{x}(t_f) - \mathbf{p}^T(t_0)\mathbf{x}(t_0)\right] \\ &= \frac{1}{2}\mathbf{x}(t_f)^T P_{t_f}\mathbf{x}(t_f) - \frac{1}{2}\left[\mathbf{x}^T(t_f)P_{t_f}\mathbf{x}(t_f) - \mathbf{x}^T(t_0)P(t_0)\mathbf{x}(t_0)\right] \\ &= \frac{1}{2}\mathbf{x}^T(t_0)P(t_0)\mathbf{x}(t_0) \end{aligned}$$

Pole Locations

- The closed-loop dynamics couple $\mathbf{x}(t)$ and $\mathbf{p}(t)$ and are given by

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix} = \begin{bmatrix} A & -B_u R_{\rm uu}^{-1} B_u^T \\ -C_z^T R_{\rm zz} C_z & -A^T \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{p}(t) \end{bmatrix}$$

  with the appropriate boundary conditions.

- OK, so where are the closed-loop poles of the system?
  - Answer: must be eigenvalues of Hamiltonian matrix for the system:

$$H \triangleq \begin{bmatrix} A & -B_u R_{\rm uu}^{-1} B_u^T \\ -C_z^T R_{\rm zz} C_z & -A^T \end{bmatrix}$$

    so we must solve $\det(sI - H) = 0$.

- **Key point:** For a SISO system, we can relate the closed-loop poles to a **Symmetric Root Locus** (SRL) for the transfer function

$$G_{zu}(s) = C_z(sI - A)^{-1}B_u = \frac{N(s)}{D(s)}$$

  - Poles and zeros of $G_{zu}(s)$ play an integral role in determining SRL
  - Note $G_{zu}(s)$ is the transfer function from control inputs to performance variable.

- In fact, the closed-loop poles are given by the LHP roots of

$$\Delta(s) = D(s)D(-s) + \frac{R_{\rm zz}}{R_{\rm uu}} N(s)N(-s) = 0$$

  - $D(s)D(-s) + \frac{R_{\rm zz}}{R_{\rm uu}} N(s)N(-s)$ is drawn using standard root locus rules - but it is symmetric wrt to both the real and imaginary axes.
  - For a stable system, we clearly just take the poles in the LHP.

Derivation of the SRL

- The closed-loop poles are given by the eigenvalues of

$$H \triangleq \begin{bmatrix} A & -B_u R_{\text{uu}}^{-1} B_u^T \\ -C_z^T R_{\text{zz}} C_z & -A^T \end{bmatrix} \rightarrow \qquad \det(sI - H) = 0$$

- **Note:** if A is invertible:

$$\det \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \det(A) \det(D - CA^{-1}B)$$

$$\Rightarrow \det(sI - H) = \det(sI - A) \det\left[(sI + A^T) - C_z^T R_{\text{zz}} C_z (sI - A)^{-1} B_u R_{\text{uu}}^{-1} B_u^T\right]$$

$$= \det(sI - A) \det(sI + A^T) \det\left[I - C_z^T R_{\text{zz}} C_z (sI - A)^{-1} B_u R_{\text{uu}}^{-1} B_u^T (sI + A^T)^{-1}\right]$$

- **Also:** $\det(I + ABC) = \det(I + CAB)$, and if $D(s) = \det(sI - A)$, then $D(-s) = \det(-sI - A^T) = (-1)^n \det(sI + A^T)$

$$\det(sI - H) = (-1)^n D(s) D(-s) \det\left[I + R_{\text{uu}}^{-1} B_u^T (-sI - A^T)^{-1} C_z^T R_{\text{zz}} C_z (sI - A)^{-1} B_u\right]$$

- If $G_{zu}(s) = C_z(sI - A)^{-1}B_u$, then $G_{zu}^T(-s) = B_u^T(-sI - A^T)^{-1}C_z^T$, so for SISO systems

$$\begin{aligned} \det(sI - H) &= (-1)^n D(s) D(-s) \det\left[I + R_{\text{uu}}^{-1} G_{zu}^T(-s) R_{\text{zz}} G_{zu}(s)\right] \\ &= (-1)^n D(s) D(-s) \left[I + \frac{R_{\text{zz}}}{R_{\text{uu}}} G_{zu}(-s) G_{zu}(s)\right] \\ &= (-1)^n \left[D(s) D(-s) + \frac{R_{\text{zz}}}{R_{\text{uu}}} N(s) N(-s)\right] = 0 \end{aligned}$$

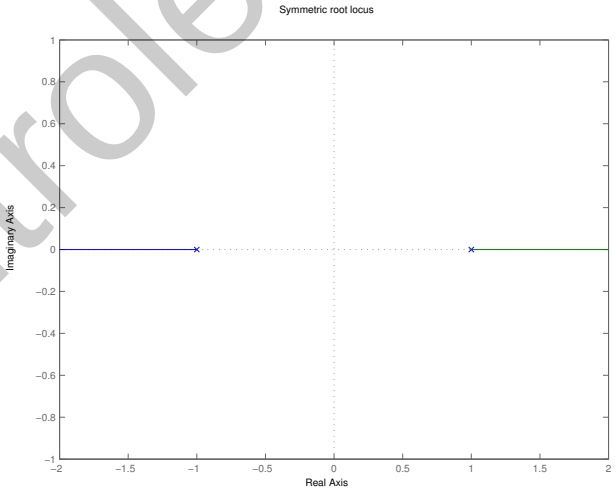
Example 6–2

- Simple example from 4–12: A scalar system with $\dot{x} = ax + bu$ with cost ($R_{xx} > 0$ and $R_{uu} > 0$) $J = \int_0^\infty (R_{zz}x^2(t) + R_{uu}u^2(t))\, dt$
- The steady-state P solves $2aP + R_{zz} - P^2b^2/R_{uu} = 0$ which gives that $P = \frac{a+\sqrt{a^2+b^2R_{zz}/R_{uu}}}{R_{uu}^{-1}b^2} > 0$
  - So that $u(t) = -Kx(t)$ where $K = R_{uu}^{-1}bP = \frac{a+\sqrt{a^2+b^2R_{zz}/R_{uu}}}{b}$
  - and the closed-loop dynamics are

$$\begin{aligned}\dot{x} &= (a - bK)x = \left(a - \frac{b}{b}(a + \sqrt{a^2 + b^2R_{zz}/R_{uu}})\right)x \\ &= -\sqrt{a^2 + b^2R_{zz}/R_{uu}}\; x = A_{cl}x(t)\end{aligned}$$

- In this case, $G_{zu}(s) = b/(s-a)$ so that $N(s) = b$ and $D(s) = (s-a)$, and the SRL is of the form:

$$(s - a)(-s - a) + \frac{R_{zz}}{R_{uu}}b^2 = 0$$



- SRL is the same whether $a < 0$ (OL stable) or $a > 0$ (OL unstable)
  - But the CLP is always the one in the LHP
  - Explains result on 4–12 about why gain $K \neq 0$ for OL unstable systems, even for expensive control problem ($R_{uu} \to \infty$)

SRL Interpretations

- For SISO case, define $R_{zz}/R_{uu} = 1/r$.

- Consider what happens as $r \rightsquigarrow \infty$ – **high control cost case**

$$\Delta(s) = D(s)D(-s) + r^{-1}N(s)N(-s) = 0 \quad \Rightarrow \quad \textbf{D(s)D(-s)=0}$$

  – So the n closed-loop poles are:
    ◇ Stable roots of the open-loop system (already in the LHP.)
    ◇ **Reflection** about the $\mathbf{j}\omega$-axis of the unstable open-loop poles.

- Consider what happens as $r \rightsquigarrow 0$ – **low control cost case**

$$\Delta(s) = D(s)D(-s) + r^{-1}N(s)N(-s) = 0 \quad \Rightarrow \quad \textbf{N(s)N(-s)=0}$$

  – Assume order of $N(s)N(-s)$ is $2m < 2n$
  – So the n closed-loop poles go to:
    ◇ The m finite zeros of the system that are in the LHP (or the reflections of the system zeros in the RHP).
    ◇ The system zeros at infinity (there are $n - m$ of these).

- The poles tending to infinity do so along very specific paths so that they form a **Butterworth Pattern**:
  – At high frequency we can ignore all but the highest powers of s in the expression for $\Delta(s) = 0$

$$\Delta(s) = 0 \quad \rightsquigarrow \quad (-1)^n s^{2n} + r^{-1}(-1)^m (b_o s^m)^2 = 0$$

$$\Rightarrow s^{2(n-m)} = (-1)^{n-m+1}\frac{b_o^2}{r}$$

- The $2(n-m)$ solutions of this expression lie on a circle of radius

$$(b_0^2/r)^{1/2(n-m)}$$

at the intersection of the radial lines with **phase from the negative real axis**:

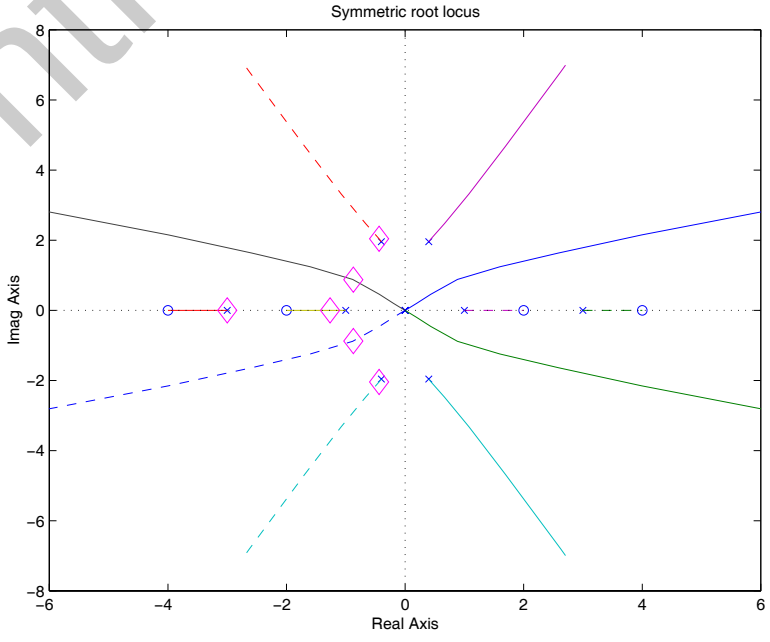
$$\pm\frac{l\pi}{n-m}\ ,\quad l=0,1,\ldots,\frac{n-m-1}{2}\ ,\qquad \textbf{(n-m) odd}$$

$$\pm\frac{(l+1/2)\pi}{n-m}\ ,\quad l=0,1,\ldots,\frac{n-m}{2}-1\ ,\qquad \textbf{(n-m) even}$$

$n-m$	Phase
1	0
2	$\pm\pi/4$
3	0, $\pm\pi/3$
4	$\pm\pi/8$, $\pm 3\pi/8$

- **Note:** Plot the SRL using the 180^o rules (normal) if $n-m$ is even and the 0^o rules if $n-m$ is odd.

Figure 6.2: $G(s)=\frac{(s-2)(s-4)}{(s-1)(s-3)(s^2+0.8s+4)s^2}$



- As noted previously, we are free to pick the state weighting matrices C_z to penalize the parts of the motion we are most concerned with.

- Simple example – consider oscillator with $\mathbf{x} = [\, p \,, \quad v \,]^T$

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -0.5 \end{bmatrix} \,, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

but we choose two cases for z

$$z = p = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x} \qquad \text{and} \qquad z = v = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}$$

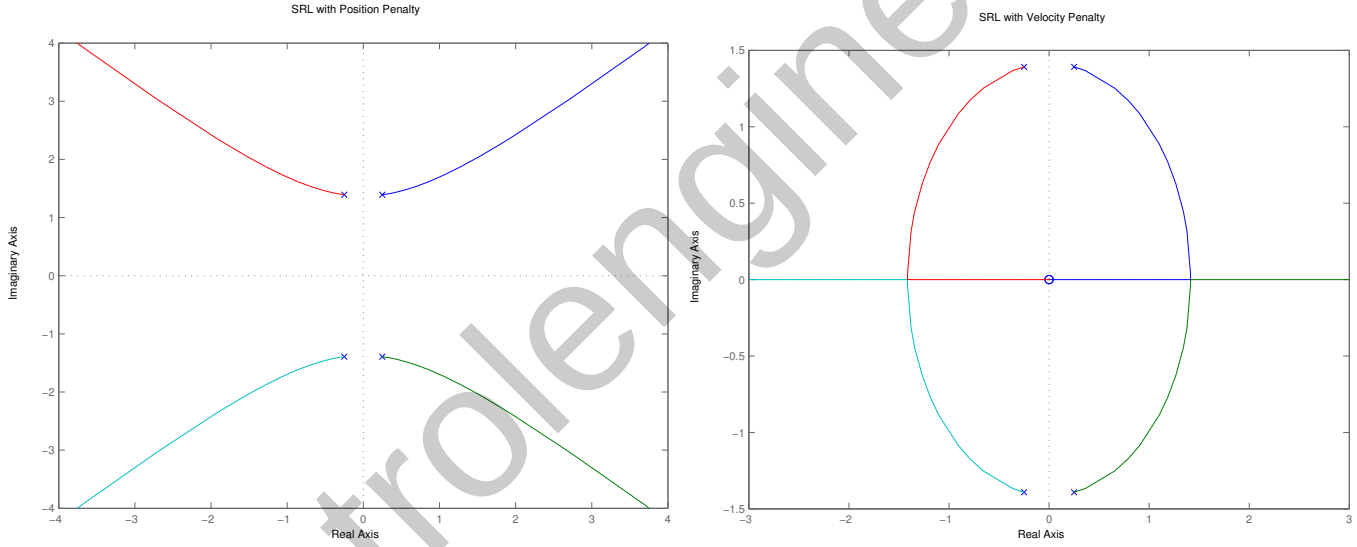


Figure 6.3: SRL with position (left) and velocity penalties (right)

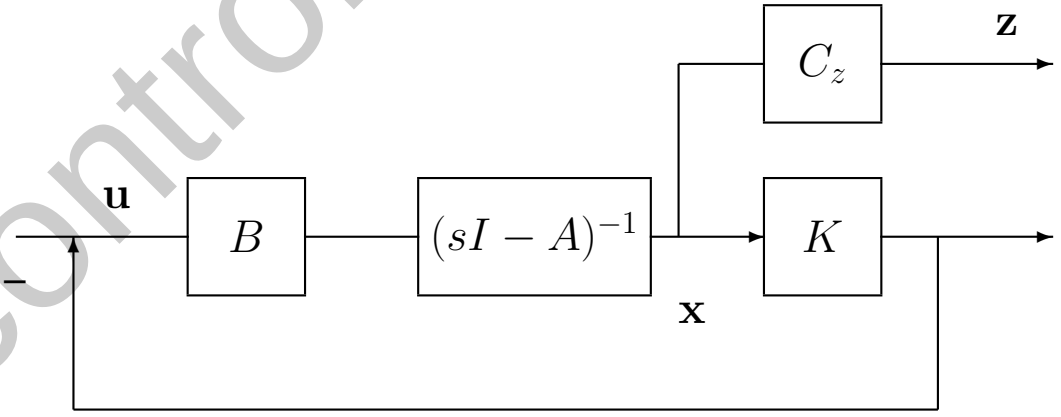
- Clearly, choosing a different C_z impacts the SRL because it completely changes the zero-structure for the system.

LQR Stability Margins

- LQR/SRL approach selects closed-loop poles that **balance** between system errors and the control effort.
  - Easy design iteration using r – poles move along the SRL.
  - Sometimes difficult to relate the desired transient response to the LQR cost function.
- Particularly nice thing about the LQR approach is that the designer is focused on system performance issues

- Turns out that the news is even better than that, because LQR exhibits very good stability margins
  - Consider the LQR stability robustness.

$$J = \int_0^\infty \mathbf{z}^T\mathbf{z} + \rho\mathbf{u}^T\mathbf{u}\, dt$$
$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$
$$\mathbf{z} = C_z\mathbf{x}, \quad R_{\text{xx}} = C_z^T C_z$$



- Study robustness in the frequency domain.
  - Loop transfer function $L(s) = K(sI - A)^{-1}B$
  - Cost transfer function $C(s) = C_z(sI - A)^{-1}B$

- Can develop a relationship between the open-loop cost $C(s)$ and the closed-loop return difference $I+L(s)$ called the **Kalman Frequency Domain Equality**

$$[I + L(-s)]^T [I + L(s)] = 1 + \frac{1}{\rho} C^T(-s)C(s)$$

- Sketch of Proof
  - Start with $\mathbf{u} = -K\mathbf{x}$, $K = \frac{1}{\rho}B^T P$, where

$$0 = -A^T P - PA - R_{xx} + \frac{1}{\rho} PBB^T P$$

  - Introduce Laplace variable s using $\pm sP$

$$0 = (-sI - A^T)P + P(sI - A) - R_{xx} + \frac{1}{\rho} PBB^T P$$

  - Pre-multiply by $B^T(-sI - A^T)^{-1}$, post-multiply by $(sI - A)^{-1}B$
  - Complete the square . . .

$$[I + L(-s)]^T [I + L(s)] = 1 + \frac{1}{\rho} C^T(-s)C(s)$$

- Can handle the MIMO case, but look at the SISO case to develop further insights ($s = \mathbf{j}\omega$)

$$\begin{aligned} [I + L(-s)]^T [I + L(s)] &= (I + L_r(\omega) - \mathrm{j}L_i(\omega))(I + L_r(\omega) + \mathrm{j}L_i(\omega)) \\ &\equiv |1 + L(\mathrm{j}\omega)|^2 \end{aligned}$$

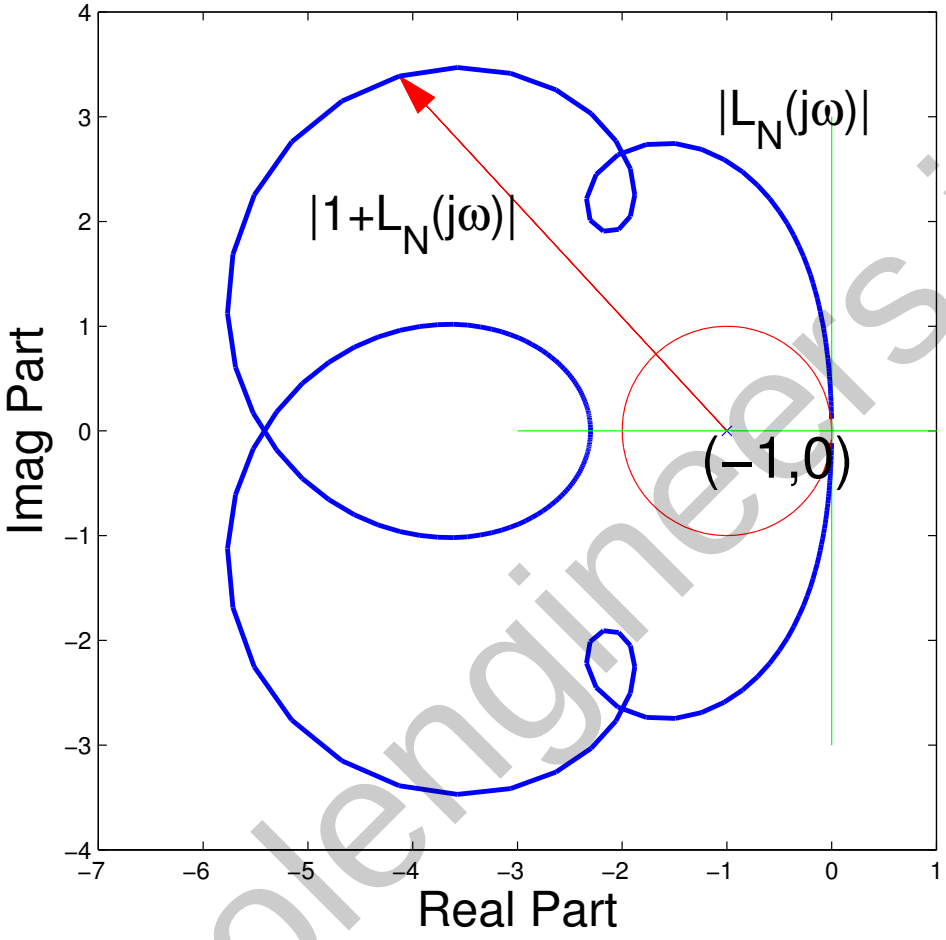
and

$$C^T(-\mathbf{j}\omega)C(\mathrm{j}\omega) = C_r^2 + C_i^2 = |C(\mathrm{j}\omega)|^2 \geq 0$$

- Thus the KFE becomes

$$|1 + L(\mathrm{j}\omega)|^2 = 1 + \frac{1}{\rho}|C(\mathrm{j}\omega)|^2 \geq 1$$

- **Implications:** The Nyquist plot of $L(j\omega)$ will always be outside the unit circle centered at (-1,0).



- Great, but why is this so significant? Recall the SISO form of the **Nyquist Stability Theorem:**

  If the loop transfer function $L(s)$ has P poles in the RHP s-plane (and $\lim_{s\to\infty} L(s)$ is a constant), then for closed-loop stability, the locus of $L(j\omega)$ for $\omega : (-\infty, \infty)$ must encircle the critical point (-1,0) P times in the **counterclockwise** direction (Ogata528)

- So we can directly prove stability from the Nyquist plot of $L(s)$.
  But what if the model is wrong and it turns out that the actual loop transfer function $L_A(s)$ is given by:

$$L_A(s) = L_N(s)[1 + \Delta(s)], \quad |\Delta(j\omega)| \leq 1, \quad \forall\omega$$

- We need to determine whether these perturbations to the loop TF will change the decision about closed-loop stability

  $\Rightarrow$ can do this directly by determining if it is possible to **change the number of encirclements of the critical point**

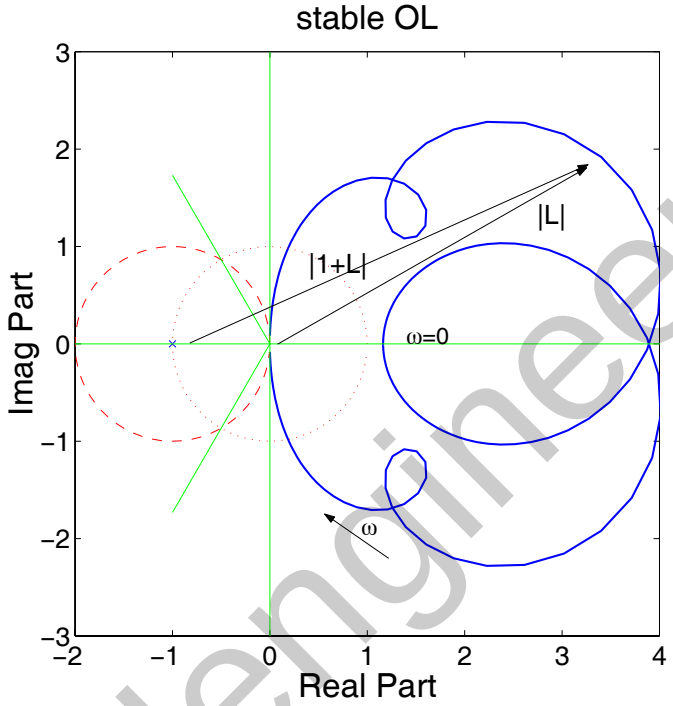
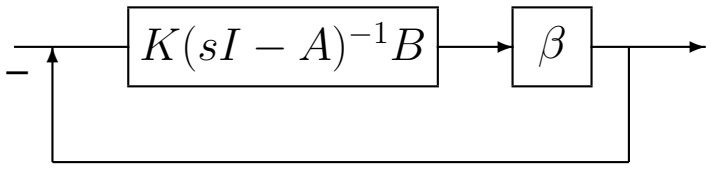


Figure 6.4: Example of LTF for an open-loop stable system

- Claim is that "since the LTF $L(\mathrm{j}\omega)$ is guaranteed to be far from the critical point for all frequencies, then LQR is VERY robust."
  - Can study this by introducing a modification to the system, where nominally $\beta = 1$, but we would like to consider:
    - ◇ The gain $\beta \in \mathbb{R}$
    - ◇ The phase $\beta \in e^{j\phi}$



- In fact, can be shown that:
  - If open-loop system is stable, then any $\beta \in (0, \infty)$ yields a stable closed-loop system. For an unstable system, any $\beta \in (1/2, \infty)$ yields a stable closed-loop system $\Rightarrow$ gain margins are $(1/2, \infty)$
  - Phase margins of at least $\pm 60^\circ$

  $\Rightarrow$ which are both huge.

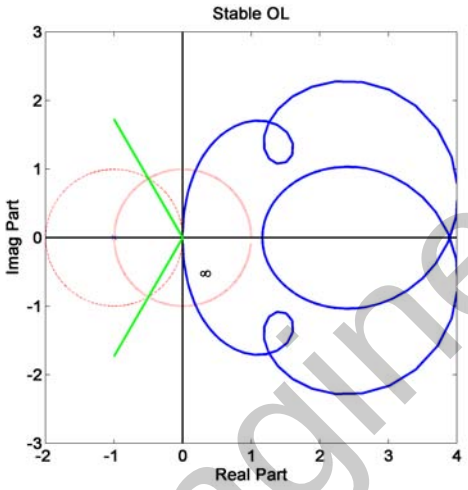


Figure 6.5: Example loop transfer functions for open-loop stable system.

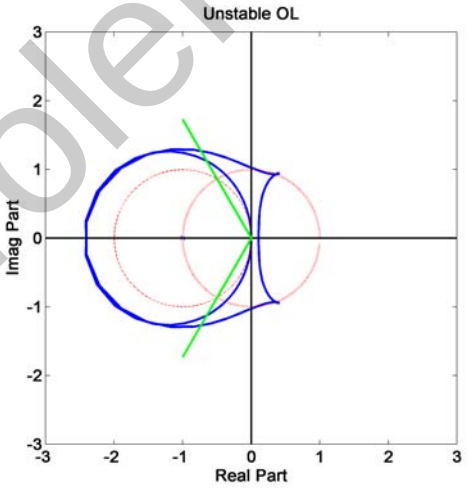


Figure 6.6: Example loop transfer functions for open-loop unstable system.

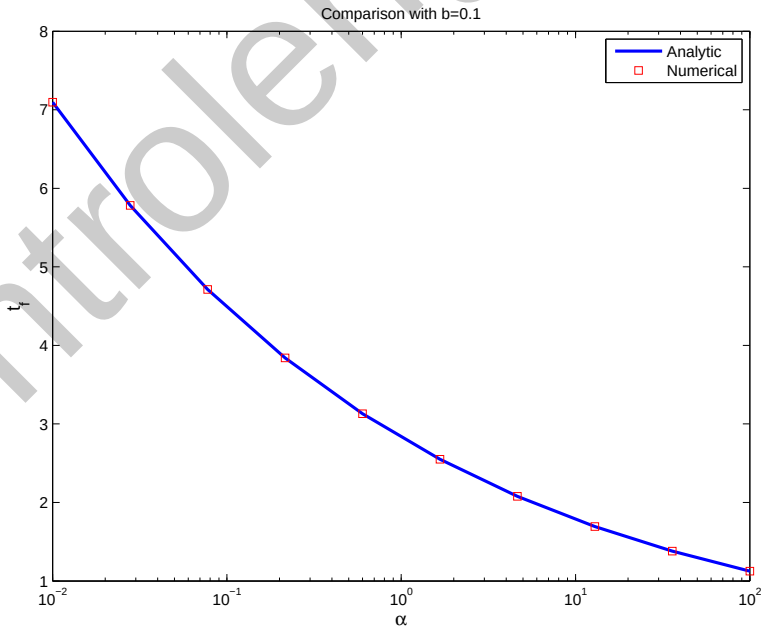
- While we have large margins, be careful because changes to some of the parameters in A or B can have a very large change to $L(s)$.
- Similar statements hold for the MIMO case, but it requires singular value analysis tools.

LTF for KDE

```
% Simple example showing LTF for KDE
% 16.323 Spring 2007
% Jonathan How
% rs2.m
%
clear all;close all;
set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontWeight','demi')
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontWeight','demi')

a=diag([-.75 -.75 -1 -1])+diag([-2 0 -4],1)+diag([2 0 4],-1);
b=[
       0.8180
    0.6602
    0.3420
    0.2897];
cz=[    0.3412    0.5341    0.7271    0.3093];
r=1e-2;
eig(a)
k=lqr(a,b,cz'*cz,r)
w=logspace(-2,2,200)';w2=-w(length(w):-1:1);
ww=[w2;0;w];
G=freqresp(a,b,k,0,1,sqrt(-1)*ww);

p=plot(G);
tt=[0:.1:2*pi]';Z=cos(tt)+sqrt(-1)*sin(tt);
hold on;plot(-1+Z,'r--');plot(Z,'r:','LineWidth',2);
plot(-1+1e-9*sqrt(-1),'x')
plot([0 0]',[-3 3]','k-','LineWidth',1.5)
plot([-3 6],[0 0]','k-','LineWidth',1.5)
plot([0 -2*cos(pi/3)],[0 -2*sin(pi/3)]','g-','LineWidth',2)
plot([0 -2*cos(pi/3)],[0 2*sin(pi/3)]','g-','LineWidth',2)
hold off
set(p,'LineWidth',2);
axis('square')
axis([-2 4 -3 3])

ylabel('Imag Part');xlabel('Real Part');title('Stable OL')
text(.25,-.5,'\infty')
print -dpng -r300 tf.png

%%%%%%%%%%%%%%%%%%%%%

a=diag([-.75 -.75 1 1])+diag([-2 0 -4],1)+diag([2 0 4],-1);
r=1e-1;
eig(a)
k=lqr(a,b,cz'*cz,r)
G=freqresp(a,b,k,0,1,sqrt(-1)*ww);

p=plot(G);
hold on;plot(-1+Z,'r--');plot(Z,'r:','LineWidth',2);
plot(-1+1e-9*sqrt(-1),'x')
plot([0 0]',[-3 3]','k-','LineWidth',1.5)
plot([-3 6],[0 0]','k-','LineWidth',1.5)
plot([0 -2*cos(pi/3)],[0 -2*sin(pi/3)]','g-','LineWidth',2)
plot([0 -2*cos(pi/3)],[0 2*sin(pi/3)]','g-','LineWidth',2)
hold off
set(p,'LineWidth',2)
axis('square')
axis([-3 3 -3 3])

ylabel('Imag Part');xlabel('Real Part');title('Unstable OL')
print -dpng -r300 tf1.png
```

16.323 Lecture 7

Numerical Solution in Matlab



Simple Problem

- Performance
$$J = \int_{t_0}^{t_f} (u - x)^2 dt$$
with dynamics $\dot{x} = u$ and BC $t_0 = 0$, $x_0 = 1$, $t_f = 1$.
  - So this is a fixed final time, free final state problem.
- Form Hamiltonian
$$H = (u - x)^2 + pu$$
- Necessary conditions become:
$$\dot{x} = u \tag{7.25}$$
$$\dot{p} = -2(u - x)(-1) \tag{7.26}$$
$$0 = 2(u - x) + p \tag{7.27}$$
with BC that $p(t_f) = 0$.
- Rearrange to get
$$\dot{p} = -p \tag{7.28}$$
$$\Rightarrow p(t) = c_1 e^{-t} \tag{7.29}$$
But now impose BC to get
$$p(t) = 0 \tag{7.30}$$
- This implies that $u = x$ is the optimal solution, and the closed-loop dynamics are
$$\dot{x} = x$$
with solution $x(t) = e^t$.
  - Clearly this would be an unstable response on a longer timescale, but given the cost and the short time horizon, this control is the best you can do.

Simple Zermelo's Problem

- Consider ship that has to travel through a region of strong currents. The ship is assumed to have constant speed V but its heading θ can be varied. The current is assumed to be in the y direction with speed of w.

- The motion of the boat is then given by the dynamics

$$\dot{x} = V\cos\theta \tag{7.31}$$
$$\dot{y} = V\sin\theta + w \tag{7.32}$$

- The goal is to minimize time, the performance index is

$$J = \int_0^{t_f} 1dt = t_f$$

  – with BC $x_0 = y_0 = 0$, $x_f = 1$, $y_f = 0$

  – Final time is unspecified.

- Define costate $\mathbf{p} = [p_1\ p_2]^T$, and in this case the Hamiltonian is

$$H = 1 + p_1(V\cos\theta) + p_2(V\sin\theta + w)$$

- Now use the necessary conditions to get $(\dot{\mathbf{p}} = -H_{\mathbf{x}}^T)$

$$\dot{p}_1 = -\frac{\partial H}{\partial x} = 0 \qquad \rightarrow p_1 = c_1 \tag{7.33}$$
$$\dot{p}_2 = -\frac{\partial H}{\partial y} = 0 \qquad \rightarrow p_2 = c_2 \tag{7.34}$$

- Control input $\theta(t)$ is unconstrained, so have $(H_{\mathbf{u}} = 0)$

$$\frac{\partial H}{\partial u} = -p_1 V \sin\theta + p_2 V \cos\theta = 0 \tag{7.35}$$

which gives the control law

$$\tan\theta = \frac{p_2}{p_1} = \frac{-p_2}{-p_1} \tag{7.36}$$

– Since p_1 and p_2 are constants, then $\theta(t)$ is also a constant.

- Optimal control is constant, so can integrate the state equations:

$$x = Vt\cos\theta \tag{7.37}$$

$$y = Vt(\sin\theta + w) \tag{7.38}$$

– Now impose the BC to get $x(t_f) = 1$, $y(t_f) = 0$ to get

$$t_f = \frac{1}{V\cos\theta} \qquad \sin\theta = -\frac{w}{V}$$

- Rearrange to get

$$\cos\theta = \frac{\sqrt{V^2 - w^2}}{V}$$

which gives that

$$t_f = \frac{1}{\sqrt{V^2 - w^2}} \qquad \theta = -\arcsin\frac{w}{V}$$

– Does this make sense?

Numerical Solutions

- Most of the problems considered so far have been simple. Things get more complicated by the need to solve a two-point boundary value problem when the dynamics are nonlinear.
- Numerous solution techniques exist, including `shooting methods`[13] and `collocation`
  - Will discuss the details on these later, but for now, let us look at how to solve these use existing codes
- Matlab code called `BVP4C` exists that is part of the standard package [14]
  - Solves problems of a "standard form":

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t, \mathbf{p}) \qquad a \leq t \leq b$$

    where $\mathbf{y}$ are the variables of interest, and $\mathbf{p}$ are extra variables in the problem that can also be optimized
  - Where the system is subject to the boundary conditions:

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = 0$$

- The solution is an approximation $\mathbf{S}(t)$ which is a continuous function that is a cubic polynomial on sub-intervals $[t_n, t_{n+1}]$ of a mesh

$$a = t_0 < t_1 < \ldots < t_{n-1} < t_n = b$$

  - This approximation satisfies the boundary conditions, so that:

$$\mathbf{g}(\mathbf{S}(a), \mathbf{S}(b)) = 0$$

  - And it satisfies the differential equations (collocates) at both ends and the mid-point of each subinterval:

$$\begin{aligned}
\dot{\mathbf{S}}(t_n) &= \mathbf{f}(\mathbf{S}(t_n), t_n) \\
\dot{\mathbf{S}}((t_n + t_{n+1})/2) &= \mathbf{f}(\mathbf{S}((t_n + t_{n+1})/2), (t_n + t_{n+1})/2) \\
\dot{\mathbf{S}}(t_{n+1}) &= \mathbf{f}(\mathbf{S}(t_{n+1}), t_{n+1})
\end{aligned}$$

[13] Online reference

[14] Matlab help and BVP4C Tutorial

- Now constrain continuity in the solution at the mesh points ⇒ converts problem to a series of nonlinear algebraic equations in the unknowns
  - Becomes a "root finding problem" that can be solved iteratively (Simpson's method).

- Inputs to BVP4C are functions that evaluate the differential equation $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$ and the residual of the boundary condition (e.g. $y_1(a) = 1$, $y_2(a) = y_1(b)$, and $y_3(b) = 0$):

$$\begin{array}{rl} \texttt{function} & res = \texttt{bvpbc}(ya, yb) \\ res = [& ya(1) - 1 \\ & ya(2) - yb(1) \\ & yb(3)]; \end{array}$$

- Redo example on page 4–15 using numerical techniques
  - Finite time LQR problem with $t_f = 10$

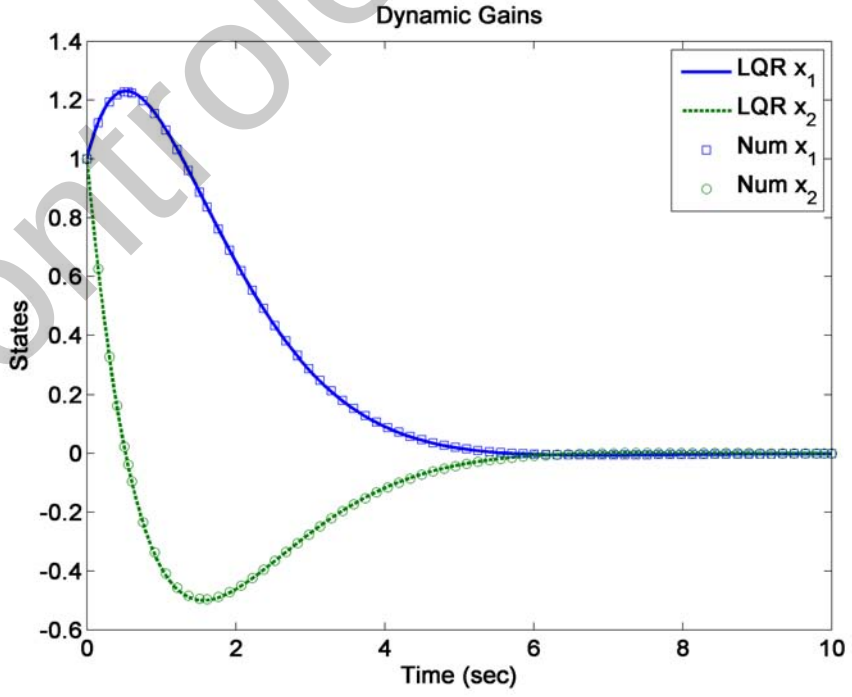


Figure 7.1: Results suggest a good comparison with the dynamic LQR result

TPBVP for LQR

```
function m = TPBVPlqr(p1,p2,p3)
global A B x0 Rxx Ruu Ptf
t_f=10;x0=[1 1]';
Rxx=p1;Ruu=p2;Ptf=p3;
solinit = bvpinit(linspace(0,t_f),@TPBVPlqrinit);
sol = bvp4c(@TPBVPlqrode,@TPBVPlqrbc,solinit);
time = sol.x;
state = sol.y([1 2],:);
adjoint = sol.y([3 4],:);
control = -inv(Ruu)*B'*sol.y([3 4],:);
m(1,:) = time;m([2 3],:) = state;m([4 5],:) = adjoint;m(6,:) = control;
%---------------------------------------------------------------------------
function dydt=TPBVPlqrode(t,y)
global A B x0 Rxx Ruu Ptf
dydt=[ A -B/Ruu*B'; -Rxx -A']*y;
%---------------------------------------------------------------------------
function res=TPBVPlqrbc(ya,yb)
global A B x0 Rxx Ruu Ptf
res=[ya(1) - x0(1);ya(2)-x0(2);yb(3:4)-Ptf*yb(1:2)];
%---------------------------------------------------------------------------
function v=TPBVPlqrinit(t)
global A B x0 b alp
v=[x0;1;0];
return

% 16.323 Spring 2007
% Jonathan How
% redo LQR example on page 4-15 using numerical approaches
clear all;close all;
set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontWeight','demi')
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontWeight','demi')
%
global A B
Ptf=[0 0;0 4];Rxx=[1 0;0 0];Ruu=1;A=[0 1;0 -1];B=[0 1]';
tf=10;dt=.01;time=[0:dt:tf];
m=TPBVPlqr(Rxx,Ruu,Ptf); % numerical result

% integrate the P backwards for LQR result
P=zeros(2,2,length(time));K=zeros(1,2,length(time));
Pcurr=Ptf;
for kk=0:length(time)-1
  P(:,:,length(time)-kk)=Pcurr;
  K(:,:,length(time)-kk)=inv(Ruu)*B'*Pcurr;
  Pdot=-Pcurr*A-A'*Pcurr-Rxx+Pcurr*B*inv(Ruu)*B'*Pcurr;
  Pcurr=Pcurr-dt*Pdot;
end

% simulate the state
x1=zeros(2,1,length(time));xcurr1=[1 1]';
for kk=1:length(time)-1
  x1(:,:,kk)=xcurr1;
  xdot1=(A-B*K(:,:,kk))*x1(:,:,kk);
  xcurr1=xcurr1+xdot1*dt;
end

figure(3);clf
plot(time,squeeze(x1(1,1,:)),time,squeeze(x1(2,1,:)),'--','LineWidth',2),
xlabel('Time (sec)');ylabel('States');title('Dynamic Gains')
hold on;plot(m(1,:),m([2],:),'s',m(1,:),m([3],:),'o');hold off
legend('LQR x_1','LQR x_2','Num x_1','Num x_2')
print -dpng -r300 numreg2.png
```

Conversion

- BVP4C sounds good, but this standard form doesn't match many of the problems that we care about
  - In particular, free end time problems are excluded, because the time period is defined to be fixed $t \in [a, b]$

- Can convert our problems of interest into this standard form though using some pretty handy tricks.
  - U. Ascher and R. D. Russell, "Reformulation of Boundary Value Problems into "Standard" Form," *SIAM Review*, Vol. 23, No. 2, 238-254. Apr., 1981.

- Key step is to re-scale time so that $\tau = t/t_f$, then $\tau \in [0, 1]$.
  - Implications of this scaling are that the derivatives must be changed since $d\tau = dt/t_f$

$$\frac{d}{d\tau} = t_f \frac{d}{dt}$$

- Final step is to introduce a dummy state r that corresponds to t_f with the trivial dynamics $\dot{r} = 0$.
  - Now replace all instances of t_f in the necessary/boundary conditions for state r.
  - Optimizer will then just pick an appropriate constant for $r = t_f$

- Recall that our basic set of necessary conditions are, for $t \in [t_0, t_f]$

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \\ \dot{\mathbf{p}} &= -H_{\mathbf{x}}^T \\ H_{\mathbf{u}} &= 0 \end{aligned}$$

- And we considered various boundary conditions $\mathbf{x}(t_0) = \mathbf{x}_0$, and:
  - If t_f is free: $h_t + g + \mathbf{p}^T\mathbf{a} = h_t + H(t_f) = 0$
  - If $\mathbf{x}_i(t_f)$ is fixed, then $\mathbf{x}_i(t_f) = x_{i_f}$
  - If $\mathbf{x}_i(t_f)$ is free, then $\mathbf{p}_i(t_f) = \dfrac{\partial h}{\partial x_i}(t_f)$

- Then

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \quad \Rightarrow \quad \mathbf{x}' = t_f \mathbf{a}(\mathbf{x}, \mathbf{u}, \tau)$$

and

$$\dot{\mathbf{p}} = -H_{\mathbf{x}}^T \quad \Rightarrow \quad \mathbf{p}' = -t_f H_{\mathbf{x}}^T$$

Example: 7–1

- Revisit example on page 6–6
- Linear system with performance/time weighting and free end time
  – Necessary conditions are:

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + Bu \\ \dot{\mathbf{p}} &= -A^T\mathbf{p} \\ 0 &= bu + \begin{bmatrix} 0 & 1 \end{bmatrix}\mathbf{p}\end{aligned}$$

with state conditions

$$\begin{aligned}\mathbf{x}_1(0) &= 10 \\ \mathbf{x}_2(0) &= 0 \\ \mathbf{x}_1(t_f) &= 0 \\ \mathbf{x}_2(t_f) &= 0 \\ -0.5bu^2(t_f) + \alpha t_f &= 0\end{aligned}$$

- Define the state of interest $\mathbf{z} = [\mathbf{x}^T \quad \mathbf{p}^T \quad r]^T$ and note that

$$\begin{aligned}\frac{d\mathbf{z}}{d\tau} &= t_f\frac{d\mathbf{z}}{dt} \\ &= \mathbf{z}_5\begin{bmatrix} A & -B\begin{bmatrix} 0 & 1 \end{bmatrix}/b & 0 \\ 0 & -A^T & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{z} \\ \Rightarrow \mathbf{z}' &= f(\mathbf{z}) \quad \text{which is nonlinear}\end{aligned}$$

with BC:

$$\begin{aligned}\mathbf{z}_1(0) &= 10 \\ \mathbf{z}_2(0) &= 0 \\ \mathbf{z}_1(1) &= 0 \\ \mathbf{z}_2(1) &= 0 \\ \frac{-0.5}{b}\mathbf{z}_4^2(1) + \alpha\mathbf{z}_5(1) &= 0\end{aligned}$$

- Code given on following pages
  - Note – it is not particularly complicated
  - Solution time/iteration count is a strong function of the initial solution – not a particularly good choice for $\mathbf{p}$ is used here

- Analytic solution gave $t_f = (1800b/\alpha)^{1/5}$
  - Numerical result give close agreement in prediction of the final time

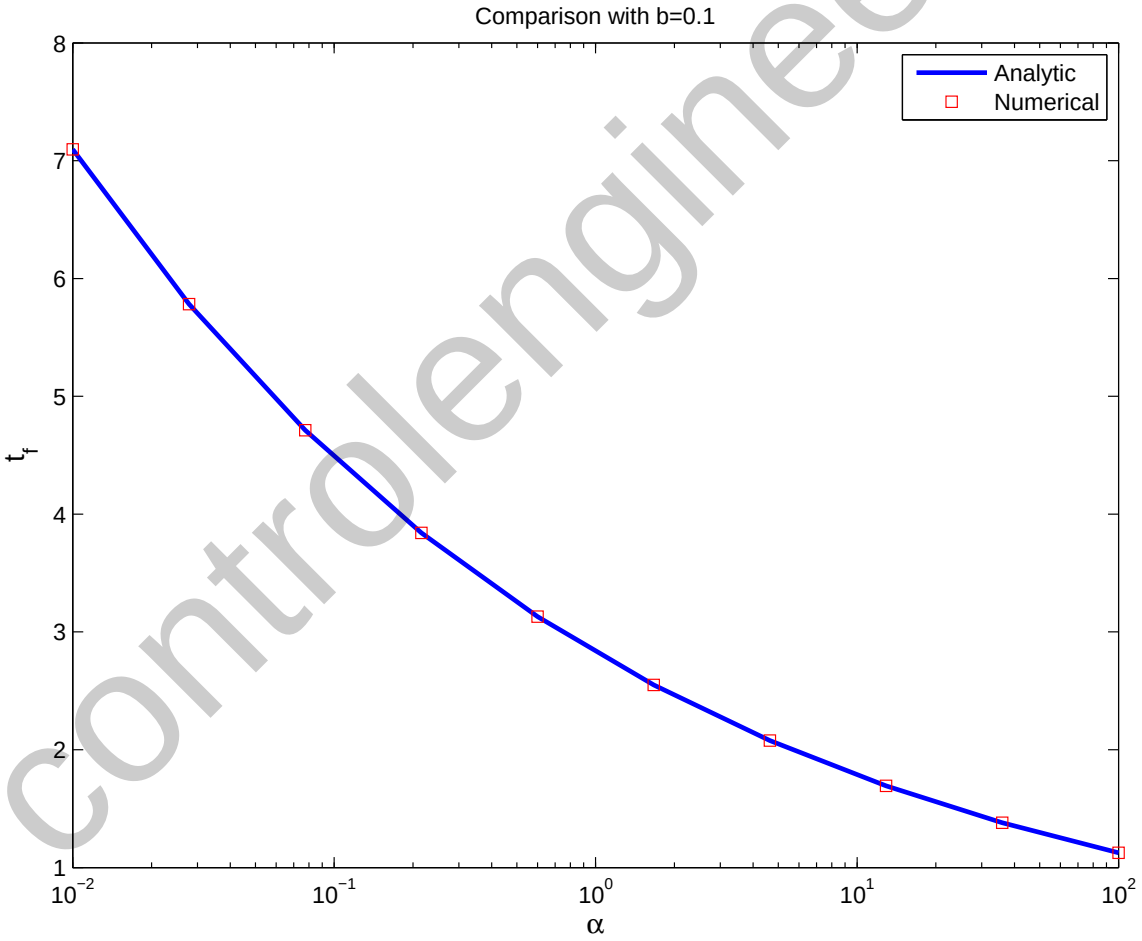


Figure 7.2: Comparison of the predicted completion times for the maneuver

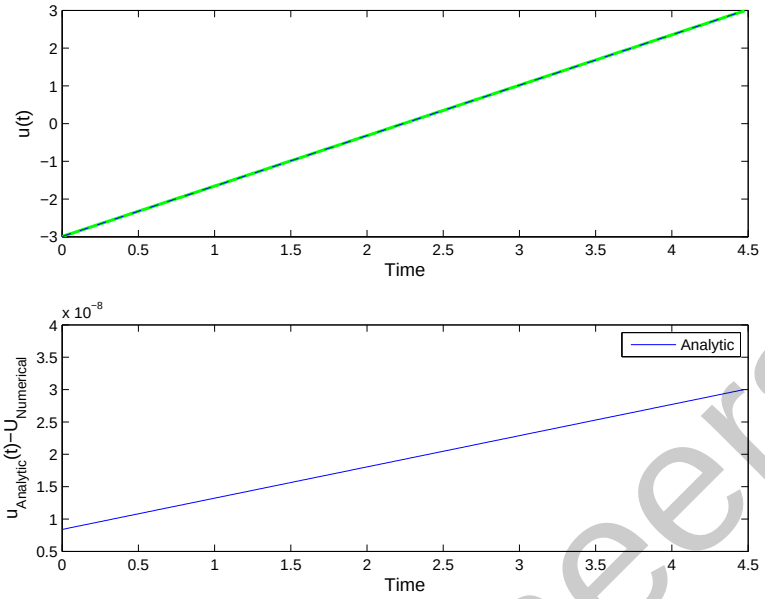


Figure 7.3: Control Inputs

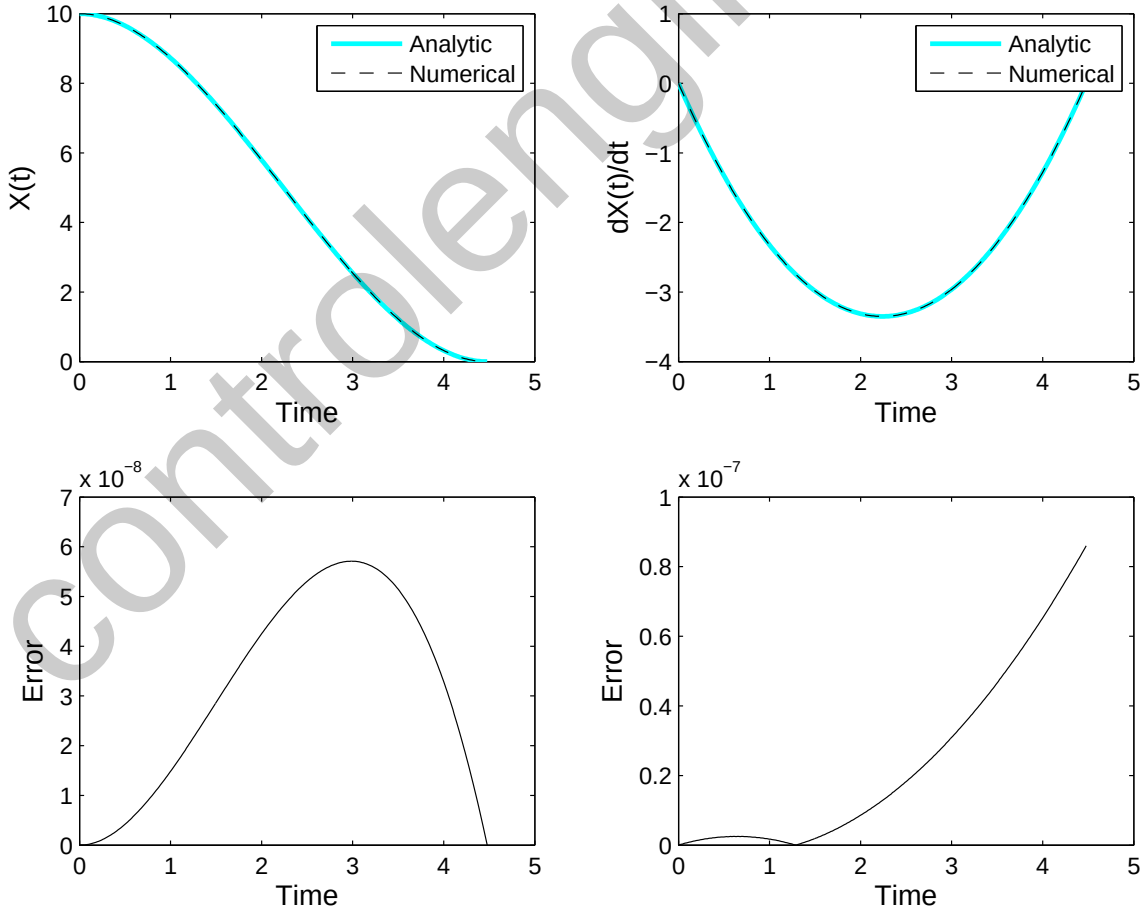


Figure 7.4: State response

TPBVP

```
function m = TPBVP(p1,p2)
% 16.323 Spring 2007
% Jonathan How
%
global A B x0 b alp

A=[0 1;0 0];
B=[0 1]';
x0=[10 0]';
b=p1;
alp=p2;

solinit = bvpinit(linspace(0,1),@TPBVPinit);
sol = bvp4c(@TPBVPode,@TPBVPbc,solinit);

time = sol.y(5)*sol.x;
state = sol.y([1 2],:);
adjoint = sol.y([3 4],:);
control = -(1/b)*sol.y(4,:);
m(1,:) = time;
m([2 3],:) = state;
m([4 5],:) = adjoint;
m(6,:) = control;

%-------------------------------------------------------------------------
function dydt=TPBVPode(t,y)
global A B x0 b alp
dydt=y(5)*[ A -B*[0 1]/b zeros(2,1); zeros(2,2) -A' zeros(2,1);zeros(1,5)]*y;

%-------------------------------------------------------------------------
function res=TPBVPbc(ya,yb)
global A B x0 b alp
res=[ya(1) - x0(1);ya(2)-x0(2);yb(1);yb(2);-0.5*yb(4)^2/b+ alp*yb(5)];

%-------------------------------------------------------------------------
function v=TPBVPinit(t)
global A B x0 b alp
v=[x0;1;0;1];

return

```

TPBVP Main

```
% 16.323 Spring 2007
% Jonathan How
% TPmain.m
%
b=0.1;
%alp=[.05 .1 1 10 20];
alp=logspace(-2,2,10);
t=[];
for alpha=alp
    m=TPBVP(b,alpha);
    t=[t;m(1,end)];
end

figure(1);clf
semilogx(alp,(1800*b./alp).^0.2,'-','Linewidth',2)
hold on;semilogx(alp,t,'rs');hold off
xlabel('\alpha','FontSize',12);ylabel('t_f','FontSize',12)
legend('Analytic','Numerical')
title('Comparison with b=0.1')
print -depsc -f1 TPBVP1.eps;jpdf('TPBVP1')

% code from opt1.m on the analytic solution
b=0.1;alpha=0.1;
m=TPBVP(b,alpha);
tf=(1800*b/alpha)^0.2;
c1=120*b/tf^3;
c2=60*b/tf^2;
u=(-c2+c1*m(1,:))/b;
A=[0 1;0 0];B=[0 1]';C=eye(2);D=zeros(2,1);G=ss(A,B,C,D);X0=[10 0]';
[y3,t3]=lsim(G,u,m(1,:),X0);

figure(2);clf
subplot(211)
plot(m(1,:),u,'g-','LineWidth',2);
xlabel('Time','FontSize',12);ylabel('u(t)','FontSize',12)
hold on;plot(m(1,:),m(6,:),'--');hold off
subplot(212)
plot(m(1,:),abs(u-m(6,:)),'-')
xlabel('Time','FontSize',12)
ylabel('u_{Analytic}(t)-U_{Numerical}','FontSize',12)
legend('Analytic','Numerical')
print -depsc -f2 TPBVP2.eps;jpdf('TPBVP2')

figure(3);clf
subplot(221)
plot(m(1,:),y3(:,1),'c-','LineWidth',2);
xlabel('Time','FontSize',12);ylabel('X(t)','FontSize',12)
hold on;plot(m(1,:),m([2],:),'k--');hold off
legend('Analytic','Numerical')
subplot(222)
plot(m(1,:),y3(:,2),'c-','LineWidth',2);
xlabel('Time','FontSize',12);ylabel('dX(t)/dt','FontSize',12)
hold on;plot(m(1,:),m([3],:),'k--');hold off
legend('Analytic','Numerical')
subplot(223)
plot(m(1,:),abs(y3(:,1)-m(2,:)'),'k-')
xlabel('Time','FontSize',12);ylabel('Error','FontSize',12)
subplot(224)
plot(m(1,:),abs(y3(:,2)-m(3,:)'),'k-')
xlabel('Time','FontSize',12);ylabel('Error','FontSize',12)
print -depsc -f3 TPBVP3.eps;jpdf('TPBVP3')
```

Zermelo's Problem

- Simplified dynamics of a UAV flying in a horizontal plane can be modeled as:

$$\begin{aligned} \dot{x}(t) &= V\cos\theta(t) \\ \dot{y}(t) &= V\sin\theta(t) + w \end{aligned}$$

where $\theta(t)$ is the heading angle (control input) with respect to the x axis, V is the speed.

- Objective: fly from point A to B in minimum time:

$$\min J = \int_0^{t_f} (1)dt$$

where t_f is free.

– Initial conditions are:

$$x(0) = x_0 \qquad\qquad y(0) = y_0$$

– Final conditions are:

$$x(t_f) = x_1 \qquad\qquad y(t_f) = y_1$$

- Apply the standard necessary conditions with

$$H = 1 + p_1 V(\cos\theta(t)) + p_2(V\sin\theta(t) + w)$$

$$\boxed{\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x},\mathbf{u},t) \\ \dot{\mathbf{p}} &= -H_{\mathbf{x}}^T \\ H_{\mathbf{u}} &= 0 \end{aligned}}$$

$$\begin{aligned} \dot{x}(t) &= V\cos\theta(t) \\ \dot{y}(t) &= V\sin\theta(t) + w \end{aligned}$$

$$\begin{aligned} \dot{p}_1(t) &= 0 \\ \dot{p}_2(t) &= 0 \\ 0 &= -p_1\sin\theta(t) + p_2\cos\theta(t) \end{aligned}$$

  - Then add extra state for the time.
- Since t_f is free, must add terminal condition that $H(t_f) = 0$, which gives a total of 5 conditions (2 initial, 3 terminal).

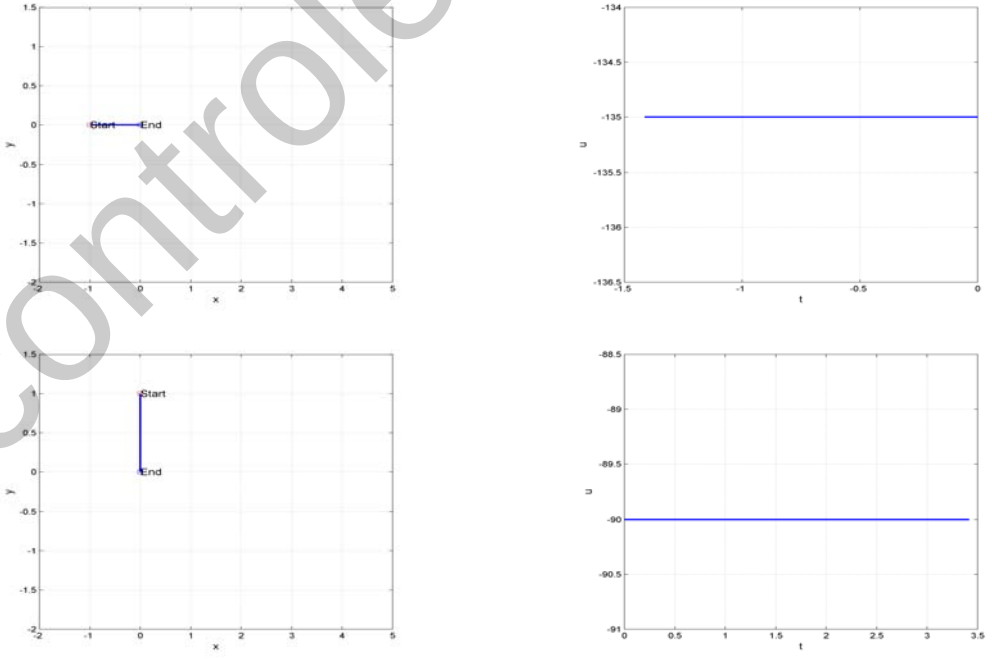


Figure 7.5: Zermelo examples

TPBVPZermelo

```
function m = TPBVPzermelo(p1,p2)
global x0 x1 V w

solinit = bvpinit(linspace(0,1),@TPBVPinit);
sol = bvp6c(@TPBVPode,@TPBVPbc,solinit);

time = sol.y(5)*sol.x;
state = sol.y([1 2],:);
adjoint = sol.y([3 4],:);
control = atan2(-sol.y([4],:),-sol.y([3],:));

m(1,:) = time;
m([2 3],:) = state;
m([4 5],:) = adjoint;
m(6,:) = control;
return

%-----------------------------------------------------------------------
function dydt=TPBVPode(t,y)
global x0 x1 V w

% x y p1 p2 t
% minimizing form
sinth=-y(4)/sqrt(y(3)^2+y(4)^2);
costh=-y(3)/sqrt(y(3)^2+y(4)^2);

dydt=y(5)*[V*costh ; V*sinth+w;0;0;0];
%-----------------------------------------------------------------------
function res=TPBVPbc(ya,yb)
global x0 x1 V w
% x y p1 p2 t
% minimizing form
costhb=-yb(3)/sqrt(yb(3)^2+yb(4)^2);
sinthb=-yb(4)/sqrt(yb(3)^2+yb(4)^2);

res=[ya(1) - x0(1);ya(2)-x0(2);
     yb(1) - x1(1);yb(2)-x1(2);
    1+V*costhb*yb(3)+V*(sinthb+w)*yb(4)];

%-----------------------------------------------------------------------
function v=TPBVPinit(t)
global x0 x1 V w
%v=[x0;-1;-1;norm(x1-x0)/(V-w)];
v=[x0;1;1;norm(x1-x0)/(V-w)];
return

clear all
global x0 x1 V w
w=1/sqrt(2);
x0=[-1 0]';x1=[0 0]';V = 1;
mm=TPBVPzermelo;

figure(1);clf
plot(mm(2,:),mm([3],:),'LineWidth',2);axis('square');grid on
axis([-2 5 -2 1.5 ])
xlabel('x','FontSize',12);ylabel('y','FontSize',12);
hold on;
plot(x0(1),x0(2),'rs');plot(x1(1),x1(2),'bs');
text(x0(1),x0(2),'Start','FontSize',12)
text(x1(1),x1(2),'End','FontSize',12)
hold off

figure(2);clf
plot(mm(1,:),180/pi*mm([6],:),'LineWidth',2);grid on;axis('square')
xlabel('t','FontSize',12);ylabel('u','FontSize',12);

print -dpng -r300 -f1 BVP_zermelo.png;
```

```
68  print -dpng -r300 -f2 BVP_zermelo2.png;
69
70  clear all
71  global x0 x1 V w
72  w=1/sqrt(2);
73  x0=[0 1]';x1=[0 0]';V = 1;
74  mm=TPBVPzermelo;
75
76  figure(1);clf
77  plot(mm(2,:),mm([3],:),'LineWidth',2);axis('square');grid on
78  axis([-2 5 -2 1.5 ])
79  xlabel('x','FontSize',12);ylabel('y','FontSize',12);
80  hold on;
81  plot(x0(1),x0(2),'rs');plot(x1(1),x1(2),'bs');
82  text(x0(1),x0(2),'Start','FontSize',12)
83  text(x1(1),x1(2),'End','FontSize',12)
84  hold off
85
86  figure(2);clf
87  plot(mm(1,:),180/pi*mm([6],:),'LineWidth',2);grid on;axis('square')
88  xlabel('t','FontSize',12);ylabel('u','FontSize',12);
89
90  print -dpng -r300 -f1 BVP_zermelo3.png;
91  print -dpng -r300 -f2 BVP_zermelo4.png;
92
```

Orbit Raising Example

- **Goal:** (Bryson page 66) determine the maximum radius orbit transfer in a given time t_f assuming a constant thrust rocket (thrust T).[15]
  - Must find the thrust direction angle $\phi(t)$
  - Assume a circular orbit for the initial and final times

- **Nomenclature:**
  - r – radial distance from attracting center, with gravitational constant μ
  - v, u tangential, radial components of the velocity
  - m mass of s/c, and $\dot{m}$ is the fuel consumption rate (constant)

- **Problem:** find $\phi(t)$ to maximize $r(t_f)$ subject to:

$$\begin{aligned} \text{Dynamics}: \quad \dot{r} &= u \\ \dot{u} &= \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T\sin\phi}{m_0 - |\dot{m}|t} \\ \dot{v} &= -\frac{uv}{r} + \frac{T\cos\phi}{m_0 - |\dot{m}|t} \end{aligned}$$

with initial conditions

$$r(0) = r_0 \qquad u(0) = 0 \qquad v(0) = \sqrt{\frac{\mu}{r_0}}$$

and terminal conditions

$$u(t_f) = 0 \qquad v(t_f) - \sqrt{\frac{\mu}{r(t_f)}} = 0$$

- With $\mathbf{p}^T = [p_1 \;\; p_2 \;\; p_3]$ this gives the Hamiltonian (since $g = 0$)

$$H = \mathbf{p}^T \begin{bmatrix} u \\ \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T\sin\phi}{m_0 - |\dot{m}|t} \\ -\frac{uv}{r} + \frac{T\cos\phi}{m_0 - |\dot{m}|t} \end{bmatrix}$$

[15] Thanks to Geoff Huntington

– Then $H_{\mathbf{u}} = 0$ with $\mathbf{u}(t) = \phi(t)$ gives

$$p_2 \left(\frac{T \cos\phi}{m_0 - |\dot{m}|t} \right) + p_3 \left(\frac{-T \sin\phi}{m_0 - |\dot{m}|t} \right) = 0$$

which gives that

$$\tan\phi = \frac{p_2(t)}{p_3(t)}$$

that can be solved for the control input given the costates.

- Note that this is a problem of the form on 6–6, with

$$\mathbf{m} = \begin{bmatrix} u(t_f) \\ v(t_f) - \sqrt{\frac{\mu}{r(t_f)}} \end{bmatrix} = 0$$

which gives

$$w = -r + \nu_1 u(t_f) + \nu_2 \left(v(t_f) - \sqrt{\frac{\mu}{r(t_f)}} \right)$$

- Since the first state r is not specified at the final time, must have that

$$p_1(t_f) = \frac{\partial w}{\partial r}(t_f) = -1 + \frac{\nu_2}{2} \sqrt{\frac{\mu}{r(t_f)^3}}$$

– And note that

$$p_3(t_f) = \frac{\partial w}{\partial v}(t_f) = \nu_2$$

which gives ν_2 in terms of the costate.

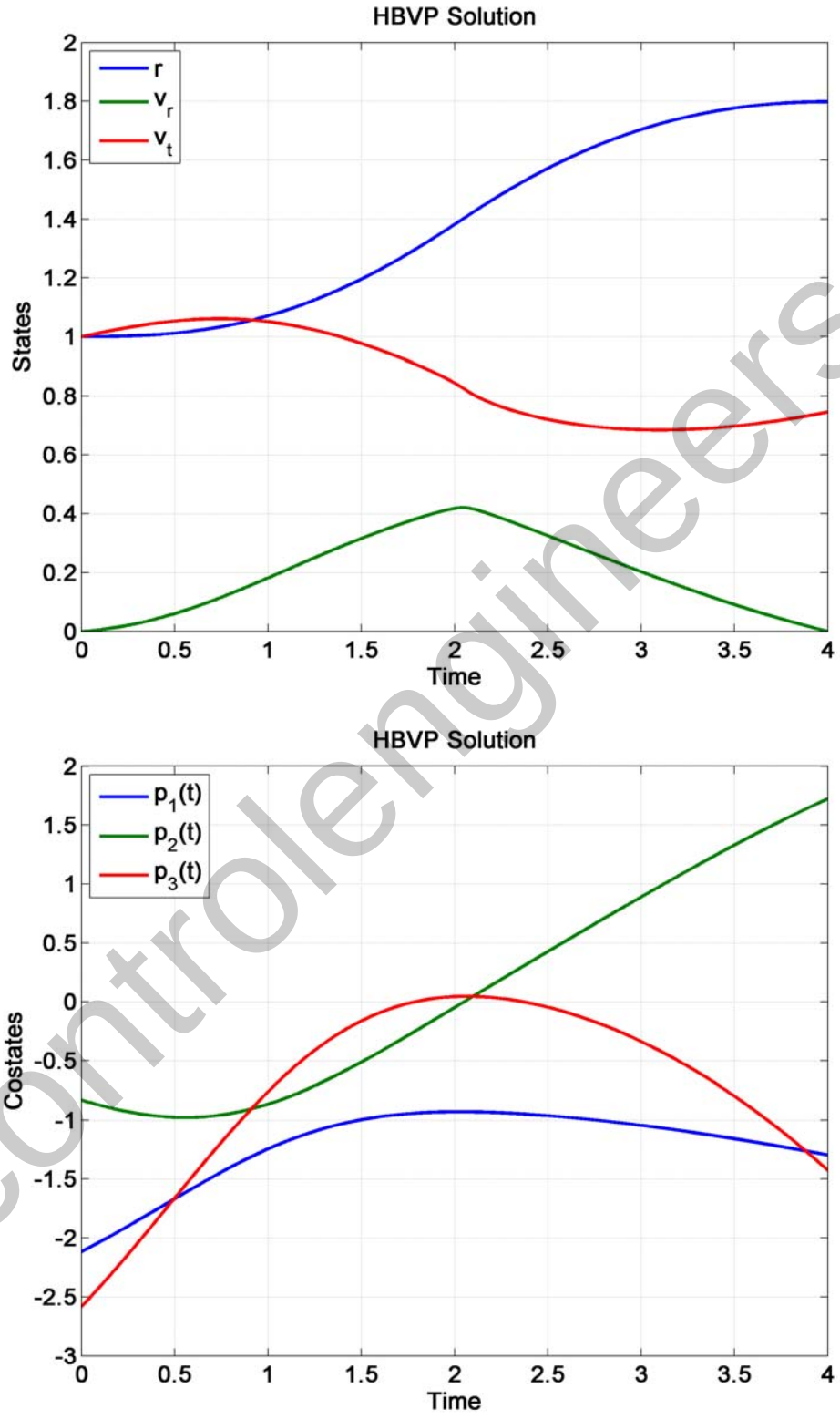


Figure 7.6: Orbit raising examples

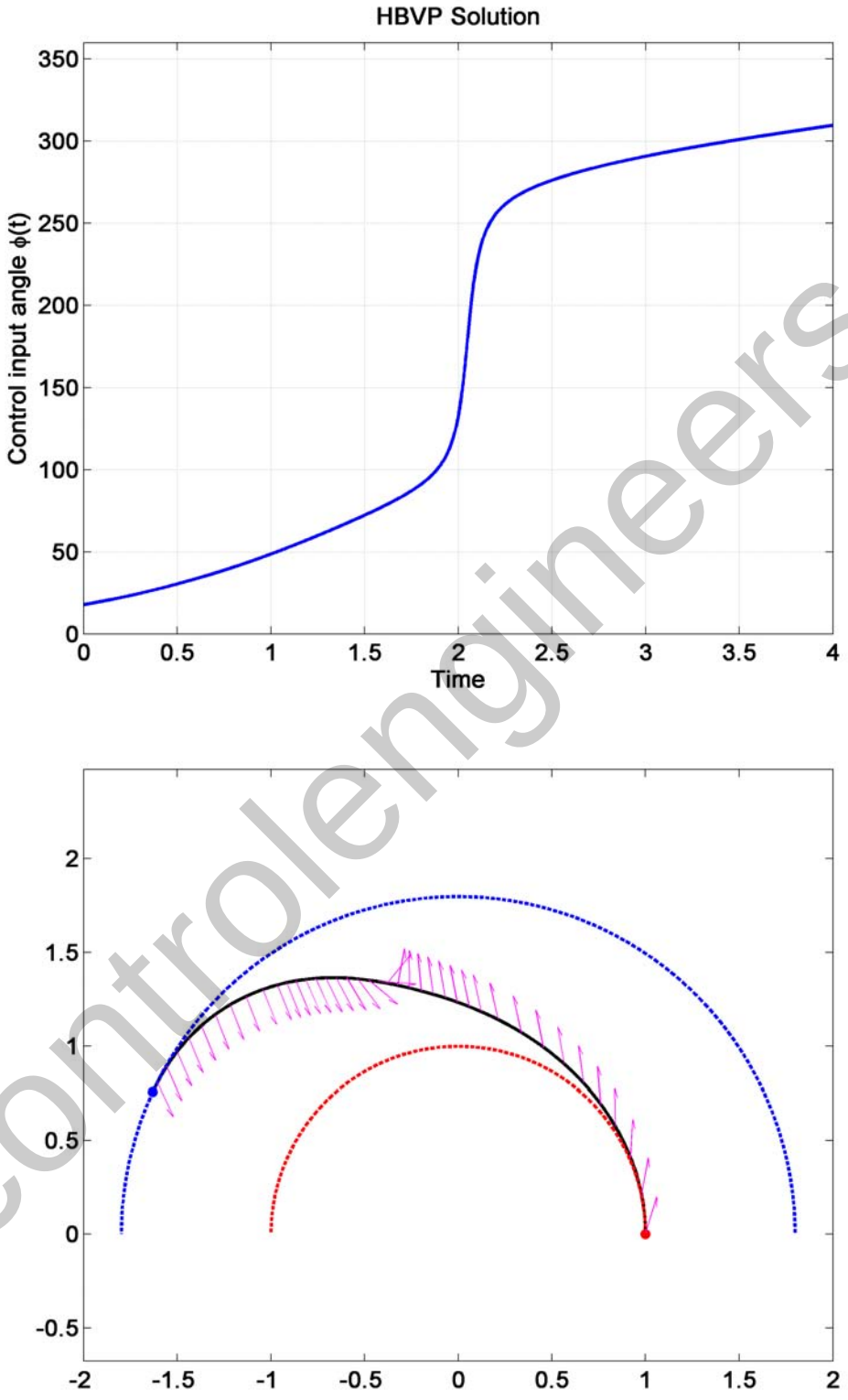


Figure 7.7: Orbit raising examples

Orbit Raising

```
%orbit_bvp_how created by Geoff Huntington 2/21/07
%Solves the Hamiltonian Boundary Value Problem for the orbit-raising optimal
%control problem (p.66 Bryson & Ho). Computes the solution using BVP4C
%Invokes subroutines orbit_ivp and orbit_bound
clear all;%close all;
set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontWeight','demi')
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontWeight','demi')

%Fixed final time %Tf = 3.3155;
Tf = 4;
four=0; % not four means use bvp6c

%Constants
global mu m0 m1 T
mu=1;  m0=1; m1=-0.07485; T= 0.1405;
%mu=1;  m0=1; m1=-.2; T= 0.1405;

%Create initial Guess
n=100;
y = [ones(1,n);  %r
     zeros(1,n); %vr
     ones(1,n);  %vt
    -ones(1,n);  %lambda_r
    -ones(1,n);  %lambda_vr
    -ones(1,n)]; %lambda_vt
x = linspace(0,Tf,n); %time
solinit.x = x;solinit.y = y;
%Set optimizer options
tol = 1E-10;
options = bvpset('RelTol',tol,'AbsTol',[tol tol tol tol tol tol],'Nmax', 2000);

%Solve
if four
    sol = bvp4c(@orbit_ivp,@orbit_bound,solinit,options);
    Nstep=40;
else
    sol = bvp6c(@orbit_ivp,@orbit_bound,solinit,options);
    Nstep=30;
end

%Plot results
figure(1);clf
plot(sol.x,sol.y(1:3,:),'LineWidth',2)
legend('r','v_r','v_t','Location','NorthWest')
grid on;
axis([0 4 0 2])
title('HBVP Solution')
xlabel('Time');ylabel('States')

figure(2);clf
plot(sol.x,sol.y(4:6,:),'LineWidth',2)
legend('p_1(t)','p_2(t)','p_3(t)','Location','NorthWest')
grid on;
axis([0 4 -3 2])
title('HBVP Solution')
xlabel('Time');ylabel('Costates')

ang2=atan2(sol.y([5],:),sol.y([6],:))+pi;
figure(3);clf
plot(sol.x,180/pi*ang2','LineWidth',2)
grid on;
axis([0 4 0 360])
title('HBVP Solution')
xlabel('Time');ylabel('Control input angle \phi(t)')
norm([tan(ang2')-(sol.y([5],:)./sol.y([6],:))'])

print -f1 -dpng -r300 orbit1.png
print -f2 -dpng -r300 orbit2.png
print -f3 -dpng -r300 orbit3.png

% Code below adapted inpart from Bryson "Dynamic Optimization"
```

```
72
73  dt=diff(sol.x);
74  dth=(sol.y(3,1:end-1)./sol.y(1,1:end-1)).*dt; % \dot \theta = v_t/r
75  th=0+cumsum(dth');
76  pathloc=[sol.y(1,1:end-1)'.*cos(th) sol.y(1,1:end-1)'.*sin(th)];
77
78  figure(4);clf
79  plot(pathloc(:,1),pathloc(:,2),'k-','LineWidth',2)
80  hold on
81  zz=exp(sqrt(-1)*[0:.01:pi]');
82  r0=sol.y(1,1);rf=sol.y(1,end);
83  plot(r0*real(zz),r0*imag(zz),'r--','LineWidth',2)
84  plot(rf*real(zz),rf*imag(zz),'b--','LineWidth',2)
85  plot(r0,0,'ro','MarkerFace','r')
86  plot(rf*cos(th(end)),rf*sin(th(end)),'bo','MarkerFace','b')
87  fact=0.2;ep=ones(size(th,1),1)*pi/2+th-ang2(1:end-1)';
88  xt=pathloc(:,1)+fact*cos(ep); yt=pathloc(:,2)+fact*sin(ep);
89  for i=1:Nstep:size(th,1),
90   pltarrow([pathloc(i,1);xt(i)],[pathloc(i,2);yt(i)],.05,'m','-');
91  end;
92  %axis([-1.6 1.6 -.1 1.8]);
93  axis([-2 2 -.1 1.8]);
94  axis('equal')
95  hold off
96
97  print -f4 -dpng -r300 orbit4.png;
```

```
1   function [dx] = orbit_ivp(t,x)
2   global mu m0 m1 T
3
4   %State
5   r = x(1);u = x(2);v = x(3);
6   lamr = x(4);lamu = x(5);lamv = x(6);
7
8   %Substitution for control
9   sinphi = -lamu./sqrt(lamu.^2+lamv.^2);
10  cosphi = -lamv./sqrt(lamu.^2+lamv.^2);
11
12  %Dynamic Equations
13  dr = u;
14  du = v^2/r - mu/r^2 + T*sinphi/(m0 + m1*t);
15  dv = -u*v/r + T*cosphi/(m0 + m1*t);
16
17  dlamr = -lamu*(-v^2/r^2 + 2*mu/r^3) - lamv*(u*v/r^2);
18  dlamu = -lamr + lamv*v/r;
19  dlamv = -lamu*2*v/r + lamv*u/r;
20
21  dx = [dr; du; dv; dlamr; dlamu; dlamv];
```

```
1   function [res] = orbit_bound(x,x2)
2   global mu m0 m1 T
3
4   %Initial State
5   r = x(1);u = x(2);v = x(3);
6   lamr = x(4);lamu = x(5);lamv = x(6);
7
8   %Final State
9   r2 = x2(1);u2 = x2(2);v2 = x2(3);
10  lamr2 = x2(4);lamu2 = x2(5);lamv2 = x2(6);
11
12  %Boundary Constraints
13  b1 = r - 1;
14  b2 = u;
15  b3 = v - sqrt(mu/r);
16  b4 = u2;
17  b5 = v2 - sqrt(mu/r2);
18  b6 = lamr2 + 1 - lamv2*sqrt(mu)/2/r2^(3/2);
19
20  %Residual
21  res = [b1;b2;b3;b4;b5;b6];
```

16.323 Lecture 8

Properties of Optimal Control Solution

Bryson and Ho – Section 3.5 and Kirk – Section 4.4

Properties of Optimal Control

- If $\mathbf{g} = \mathbf{g}(\mathbf{x}, \mathbf{u})$ and $\mathbf{a} = \mathbf{a}(\mathbf{x}, \mathbf{u})$ do not explicitly depend on time t, then the Hamiltonian H is at least piecewise constant.

$$H = g(\mathbf{x}, \mathbf{u}) + \mathbf{p}^T \mathbf{a}(\mathbf{x}, \mathbf{u}) \tag{8.1}$$

then

$$\frac{dH}{dt} = \cancelto{0}{\frac{\partial H}{\partial t}} + \left(\frac{\partial H}{\partial \mathbf{x}}\right)\frac{d\mathbf{x}}{dt} + \left(\frac{\partial H}{\partial \mathbf{u}}\right)\frac{d\mathbf{u}}{dt} + \left(\frac{\partial H}{\partial \mathbf{p}}\right)\frac{d\mathbf{p}}{dt} \tag{8.2}$$

$$= H_{\mathbf{x}}\mathbf{a} + H_{\mathbf{u}}\dot{\mathbf{u}} + H_{\mathbf{p}}\dot{\mathbf{p}} \tag{8.3}$$

Now use the necessary conditions:

$$\dot{\mathbf{x}} = \mathbf{a} = H_{\mathbf{p}}^T \tag{8.4}$$

$$\dot{\mathbf{p}} = -H_{\mathbf{x}}^T \tag{8.5}$$

to get that

$$\frac{dH}{dt} = -\dot{\mathbf{p}}^T\mathbf{a} + \mathbf{a}^T\dot{\mathbf{p}} + H_{\mathbf{u}}\dot{\mathbf{u}} = H_{\mathbf{u}}\dot{\mathbf{u}}$$

- Third necessary condition requires $H_{\mathbf{u}} = 0$, so clearly $\frac{dH}{dt} = 0$, which suggests H is a constant,
  - Note that it might be possible for the value of this constant to change at a discontinuity of $\mathbf{u}$, since then $\dot{\mathbf{u}}$ would be infinite, and $0 \cdot \infty$ is not defined.
  - Thus H is at least piecewise constant

- For free final time problems, transversality condition gives,

$$h_t + H(t_f) = 0.$$

  - If h is not a function of time, then $h_t = 0$ so $H(t_f) = 0$
  - With no jumps in $\mathbf{u}$, H is constant $\Rightarrow$ $H = 0$ for all time.

- If solution has a corner that is not induced by an intermediate state variable constraint, then H, $\mathbf{p}$, and $H_{\mathbf{u}}$ are all continuous across the corner.
- To see, this, write augmented cost functional on 6–1 in the form

$$J = \text{terminal terms} + \int_{t_0}^{t_f} \left(\mathbf{g} + \mathbf{p}^T(\mathbf{a} - \dot{\mathbf{x}})\right) dt$$

and recall definition of Hamiltonian $H = \mathbf{g} + \mathbf{p}^T\mathbf{a}$, so that

$$J = \text{terminal terms} + \int_{t_0}^{t_f} \left(H - \mathbf{p}^T\dot{\mathbf{x}}\right) dt$$

- Looks similar to the classical form analyzed on 5–16

$$\tilde{J} = \int_{t_0}^{t_f} g\left(x, \dot{x}, t\right) dt$$

which led to two Weierstrass-Erdmann corner conditions

$$g_{\dot{\mathbf{x}}}(t_1^-) = g_{\dot{\mathbf{x}}}(t_1^+) \tag{8.6}$$

$$g(t_1^-) - g_{\dot{\mathbf{x}}}(t_1^-)\dot{\mathbf{x}}(t_1^-) = g(t_1^+) - g_{\dot{\mathbf{x}}}(t_1^+)\dot{\mathbf{x}}(t_1^+) \tag{8.7}$$

- With $g\left(x, \dot{x}, t\right) \Rightarrow H - \mathbf{p}^T\dot{\mathbf{x}}$, equivalent continuity conditions are:

$$\frac{\partial(H - \mathbf{p}^T\dot{\mathbf{x}})}{\partial\dot{\mathbf{x}}} = -\mathbf{p}^T \qquad \text{must be cts at corner}$$

and

$$\begin{aligned}(H - \mathbf{p}^T\dot{\mathbf{x}}) - \frac{\partial(H - \mathbf{p}^T\dot{\mathbf{x}})}{\partial\dot{\mathbf{x}}}\dot{\mathbf{x}} \\ = (H - \mathbf{p}^T\dot{\mathbf{x}}) + \mathbf{p}^T\dot{\mathbf{x}} \\ = H \qquad \text{must be cts at corner}\end{aligned} \tag{8.8}$$

- So both $\mathbf{p}(t)$ and H must be continuous across a corner that is not induced by a state variable equality/inequality constraint.

Point State Constraint

- Consider what happens with an interior point state constraint (Bryson, section 3.5) of the form that

$$\mathbf{N}(\mathbf{x}(t_1), t_1) = 0$$

where $t_0 < t_1 < t_f$ and $\mathbf{N}$ is a vector of $q < n$ constraints.
– Assume that $\mathbf{x}(t_0)$, $\mathbf{x}(t_f)$, t_0, and t_f all specified.

- Augment constraint to cost (6–1) using multiplier $\boldsymbol{\pi}$

$$J_a = h(\mathbf{x}(t_f), t_f) + \boldsymbol{\pi}^T \mathbf{N} + \int_{t_0}^{t_f} \left(H - \mathbf{p}^T \dot{\mathbf{x}}\right) dt$$

- Proceed as before with the corner conditions (5–15), and split cost integral into 2 parts

$$\int_{t_0}^{t_f} \Rrightarrow \int_{t_0}^{t_1} + \int_{t_1}^{t_f}$$

and form the variation (drop terms associated with t_0 and t_f):

$$\begin{aligned}
\delta J_a = {} & \mathbf{N}^T(t_1)\delta\boldsymbol{\pi} + \boldsymbol{\pi}^T(\mathbf{N}_{\mathbf{x}}(t_1)\delta\mathbf{x}_1 + \mathbf{N}_t(t_1)\delta t_1) \\
& + \int_{t_0}^{t_1} \left(H_{\mathbf{x}}\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (H_{\mathbf{p}} - \dot{\mathbf{x}}^T)\delta\mathbf{p} - \mathbf{p}^T\delta\dot{\mathbf{x}}\right) dt \\
& + \int_{t_1}^{t_f} \left(H_{\mathbf{x}}\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (H_{\mathbf{p}} - \dot{\mathbf{x}}^T)\delta\mathbf{p} - \mathbf{p}^T\delta\dot{\mathbf{x}}\right) dt \\
& + \left(H - \mathbf{p}^T\dot{\mathbf{x}}\right)\Big|^{t_1^-} \delta t_1 + \left(H - \mathbf{p}^T\dot{\mathbf{x}}\right)\Big|_{t_1^+} \delta t_1
\end{aligned} \tag{8.9}$$

Collect:

$$\begin{aligned}
= {} & \mathbf{N}^T(t_1)\delta\boldsymbol{\pi} + \boldsymbol{\pi}^T(\mathbf{N}_{\mathbf{x}}(t_1)\delta\mathbf{x}_1 + \mathbf{N}_t(t_1)\delta t_1) \\
& + \int_{t_0}^{t_1} \left(H_{\mathbf{x}}\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (H_{\mathbf{p}} - \dot{\mathbf{x}}^T)\delta\mathbf{p} - \mathbf{p}^T\delta\dot{\mathbf{x}}\right) dt \\
& + \int_{t_1}^{t_f} \left(H_{\mathbf{x}}\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (H_{\mathbf{p}} - \dot{\mathbf{x}}^T)\delta\mathbf{p} - \mathbf{p}^T\delta\dot{\mathbf{x}}\right) dt \\
& + \left(H - \mathbf{p}^T\dot{\mathbf{x}}\right)(t_1^-)\delta t_1 - \left(H - \mathbf{p}^T\dot{\mathbf{x}}\right)(t_1^+)\delta t_1
\end{aligned} \tag{8.10}$$

- On 6–2 showed that the IBP will give:

$$-\int_{t_0}^{t_1} \mathbf{p}^T \delta\dot{\mathbf{x}} dt = -\mathbf{p}^T(t_1^-)\left(\delta\mathbf{x}_1 - \dot{\mathbf{x}}(t_1^-)\delta t_1\right) + \int_{t_0}^{t_1} \dot{\mathbf{p}}^T \delta\mathbf{x} dt$$
$$-\int_{t_1}^{t_f} \mathbf{p}^T \delta\dot{\mathbf{x}} dt = \mathbf{p}^T(t_1^+)\left(\delta\mathbf{x}_1 - \dot{\mathbf{x}}(t_1^+)\delta t_1\right) + \int_{t_1}^{t_f} \dot{\mathbf{p}}^T \delta\mathbf{x} dt$$

- Substitute into (13) to get

$$\begin{aligned}\delta J_a = {} & \mathbf{N}^T(t_1)\delta\boldsymbol{\pi} + \boldsymbol{\pi}^T(\mathbf{N}_\mathbf{x}(t_1)\delta\mathbf{x}_1 + \mathbf{N}_t(t_1)\delta t_1) \\ & + \int_{t_0}^{t_f} \left((H_\mathbf{x} + \dot{\mathbf{p}}^T)\delta\mathbf{x} + H_\mathbf{u}\delta\mathbf{u} + (H_\mathbf{p} - \dot{\mathbf{x}}^T)\delta\mathbf{p}\right) dt \\ & + \left(H - \mathbf{p}^T\dot{\mathbf{x}}\right)(t_1^-)\delta t_1 - \left(H - \mathbf{p}^T\dot{\mathbf{x}}\right)(t_1^+)\delta t_1 \\ & - \mathbf{p}^T(t_1^-)\left(\delta\mathbf{x}_1 - \dot{\mathbf{x}}(t_1^-)\delta t_1\right) + \mathbf{p}^T(t_1^+)\left(\delta\mathbf{x}_1 - \dot{\mathbf{x}}(t_1^+)\delta t_1\right)\end{aligned} \tag{8.11}$$

- Rearrange and cancel terms

$$\begin{aligned}\delta J_a = {} & \mathbf{N}^T(t_1)\delta\boldsymbol{\pi} + \int_{t_0}^{t_f} \left((H_\mathbf{x} + \dot{\mathbf{p}}^T)\delta\mathbf{x} + H_\mathbf{u}\delta\mathbf{u} + (H_\mathbf{p} - \dot{\mathbf{x}}^T)\delta\mathbf{p}\right) dt \\ & + \left[\mathbf{p}^T(t_1^+) - \mathbf{p}^T(t_1^-) + \boldsymbol{\pi}^T\mathbf{N}_\mathbf{x}(t_1)\right]\delta\mathbf{x}_1 \\ & + \left[H(t_1^-) - H(t_1^+) + \boldsymbol{\pi}^T\mathbf{N}_t(t_1)\right]\delta t_1\end{aligned} \tag{8.12}$$

- So choose $H(t_1^-)$ & $H(t_1^+)$ and $\mathbf{p}^T(t_1^-)$ & $\mathbf{p}^T(t_1^+)$ to ensure that the coefficients of $\delta\mathbf{x}_1$ and t_1 vanish in (15), giving:

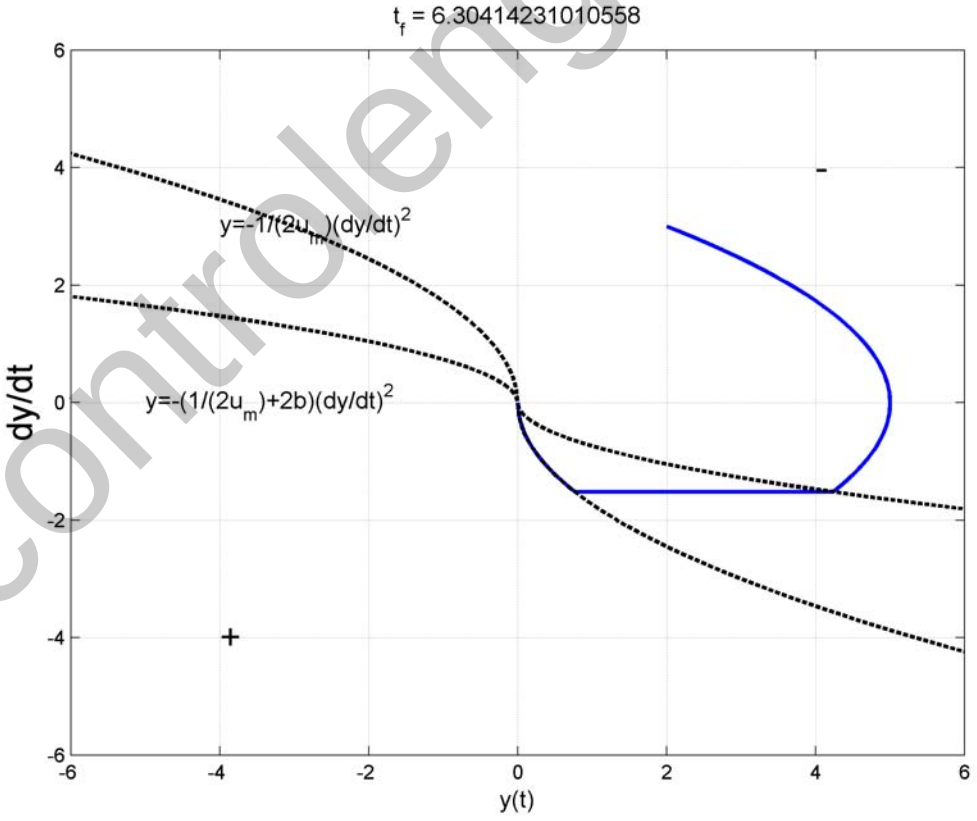
$$\begin{aligned}\mathbf{p}^T(t_1^-) &= \mathbf{p}^T(t_1^+) + \boldsymbol{\pi}^T\mathbf{N}_\mathbf{x}(t_1) \\ H(t_1^-) &= H(t_1^+) - \boldsymbol{\pi}^T\mathbf{N}_t(t_1)\end{aligned}$$

- These explicitly show that $\mathbf{p}(t_1)$ and $H(t_1)$ are discontinuous across the state constraint induced corner, but $H_\mathbf{u}$ will be continuous.

16.323 Lecture 9

Constrained Optimal Control

Bryson and Ho – Section 3.x and Kirk – Section 5.3



Constrained Optimal Control

- First consider cases with constrained control inputs so that $\mathbf{u}(t) \in \mathcal{U}$ where $\mathcal{U}$ is some bounded set.
  - Example: inequality constraints of the form $\mathbf{C}(\mathbf{x}, \mathbf{u}, t) \leq 0$
  - Much of what we had on 6–3 remains the same, but algebraic condition that $H_{\mathbf{u}} = 0$ must be replaced
  - Note that $\mathbf{C}(\mathbf{x}, t) \leq 0$ is a much harder case

- Augment constraint to cost (along with differential equation constraints)

$$J_a = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \left[H - \mathbf{p}^T \dot{\mathbf{x}} + \boldsymbol{\nu}^T \mathbf{C} \right] dt$$

- Find the variation (assume t_0 and $x(t_0)$ fixed):

$$\begin{aligned} \delta J_a \;=\; & h_{\mathbf{x}} \delta \mathbf{x}_f + h_{t_f} \delta t_f + \int_{t_0}^{t_f} \Big[H_{\mathbf{x}} \delta \mathbf{x} + H_{\mathbf{u}} \delta \mathbf{u} + (H_{\mathbf{p}} - \dot{\mathbf{x}}^T) \delta \mathbf{p}(t) \\ & - \mathbf{p}^T(t) \delta \dot{\mathbf{x}} + \mathbf{C}^T \delta \boldsymbol{\nu} + \boldsymbol{\nu}^T \left\{ \mathbf{C}_{\mathbf{x}} \delta \mathbf{x} + \mathbf{C}_{\mathbf{u}} \delta \mathbf{u} \right\} \Big] dt \\ & + \left[H - \mathbf{p}^T \dot{\mathbf{x}} + \boldsymbol{\nu}^T \mathbf{C} \right] (t_f) \delta t_f \end{aligned}$$

- Now IBP

$$-\int_{t_0}^{t_f} \mathbf{p}^T(t) \delta \dot{\mathbf{x}} dt = -\mathbf{p}^T(t_f) \left(\delta \mathbf{x}_f - \dot{\mathbf{x}}(t_f) \delta t_f \right) + \int_{t_0}^{t_f} \dot{\mathbf{p}}^T(t) \delta \mathbf{x} dt$$

  then combine and drop terminal conditions for simplicity:

$$\begin{aligned} \delta J_a \;=\; & \int_{t_0}^{t_f} \Big\{ \left[H_{\mathbf{x}} + \dot{\mathbf{p}}^T + \boldsymbol{\nu}^T \mathbf{C}_{\mathbf{x}} \right] \delta \mathbf{x} + \left[H_{\mathbf{u}} + \boldsymbol{\nu}^T \mathbf{C}_{\mathbf{u}} \right] \delta \mathbf{u} \\ & + (H_{\mathbf{p}} - \dot{\mathbf{x}}^T) \delta \mathbf{p}(t) + \mathbf{C}^T \delta \boldsymbol{\nu} \Big\} dt \end{aligned}$$

- Clean up by defining augmented **Hamiltonian**

$$H_a(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) = g + \mathbf{p}^T(t)\mathbf{a} + \boldsymbol{\nu}^T(t)\mathbf{C}$$

  where (see 2–12)

$$\nu_i(t) \begin{cases} \geq 0 & \text{if} \quad C_i = 0 \quad \text{active} \\ = 0 & \text{if} \quad C_i < 0 \quad \text{inactive} \end{cases}$$

  – So that $\nu_i C_i = 0 \; \forall \; i$.

- So necessary conditions for $\delta J_a = 0$ are that for $t \in [t_0, t_f]$

$$\boxed{\begin{aligned} \dot{\mathbf{x}} &= \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \\ \dot{\mathbf{p}} &= -(H_a)_{\mathbf{x}}^T \\ (H_a)_{\mathbf{u}} &= 0 \end{aligned}}$$

  – With appropriate boundary conditions and $\nu_i C_i(\mathbf{x}, \mathbf{u}, t) = 0$

- Complexity here is that typically will have sub-arcs to the solution where the inequality constraints are active (so $C_i(\mathbf{x}, \mathbf{u}, t) = 0$) and then not (so $\nu_i = 0$).
  – Transitions between the sub-arcs must be treated as corners that are at unspecified times - need to impose the equivalent of the Erdmann-Weirstrass corner conditions for the control problem, as in Lecture 8.

Constrained Example

- Design the control inputs that minimize the cost functional

$$\min_u J = -x(4) + \int_0^4 u^2(t)dt$$

with $\dot{x} = x + u$, $x(0) = 0$, and $u(t) \leq 5$.

- Form augmented Hamiltonian:

$$H = u^2 + p(x + u) + \nu(u - 5)$$

- Note that, independent of whether the constraint is active or not, we have that

$$\dot{p} = -H_x = -p \quad \Rightarrow \quad p(t) = ce^{-t}$$

and from transversality BC, know that $p(4) = \partial h/\partial x = -1$, so have that $c = -e^4$ and thus $p(t) = -e^{4-t}$

- Now let us assume that the control constraint is initially active for some period of time, then $\nu \geq 0$, $u = 5$, and

$$H_u = 2u + p + \nu = 0$$

so we have that

$$\nu = -10 - p = -10 + e^{4-t}$$

– Question: for what values of t will $\nu \geq 0$?

$$\begin{aligned} \nu &= -10 + e^{4-t} \geq 0 \\ &\rightarrow e^{4-t} \geq 10 \\ &\rightarrow 4 - t \geq \ln(10) \\ &\rightarrow 4 - \ln(10) \geq t \end{aligned}$$

– So provided $t \leq t_c = 4 - \ln(10)$ then $\nu \geq 0$ and the assumptions are consistent.

- Now consider the inactive constraint case:

$$H_u = 2u + p = 0 \Rightarrow u(t) = -\frac{1}{2}p(t)$$

- The control inputs then are

$$u(t) = \begin{cases} 5 & t \le t_c \\ \frac{1}{2}e^{4-t} & t \ge t_c \end{cases}$$

which is continuous at t_c.

- To finish the solution, find the state in the two arcs $x(t)$ and enforce continuity at t_c, which gives that:

$$x(t) = \begin{cases} 5e^t - 5 & t \le t_c \\ -\frac{1}{4}e^{4-t} + (5 - 25e^{-4})e^t) & t \ge t_c \end{cases}$$

- Note that since the corner condition was not specified by a state constraint, continuity of λ and H at the corner is required – but we did not need to use that in this solution, it will occur naturally.

Pontryagin's Minimum Principle

- For an alternate perspective, consider general control problem statement on 6–1 (free end time and state). Then on 6–2,

$$\delta J_a = \left(h_{\mathbf{x}} - \mathbf{p}^T(t_f)\right)\delta\mathbf{x}_f + \left[h_{t_f} + H\right](t_f)\delta t_f \qquad (9.13)$$
$$+ \int_{t_0}^{t_f} \left[\left(H_{\mathbf{x}} + \dot{\mathbf{p}}^T\right)\delta\mathbf{x} + H_{\mathbf{u}}\delta\mathbf{u} + (H_{\mathbf{p}} - \dot{\mathbf{x}}^T)\delta\mathbf{p}(t)\right] dt$$

  now assume we have a trajectory that satisfies all other differential equation and terminal constraints, then all remains is

$$\Rightarrow \delta J_a = \int_{t_0}^{t_f} \left[H_{\mathbf{u}}(t)\delta\mathbf{u}(t)\right] dt \qquad (9.14)$$

- For the control to be minimizing, need $\delta J_a \geq 0$ for all admissible variations in $\mathbf{u}$ (i.e., $\delta\mathbf{u}$ for which $C_{\mathbf{u}}\delta\mathbf{u} \leq 0$)
  - Equivalently, need $\delta H = H_{\mathbf{u}}(t)\delta\mathbf{u}(t) \geq 0$ for all time and for all admissible $\delta\mathbf{u}$
  - Gives condition that $H_{\mathbf{u}} = 0$ if control constraints not active
  - However, at the constraint boundary, could have $H_{\mathbf{u}} \neq 0$ and whether we need $H_{\mathbf{u}} > 0$ or $H_{\mathbf{u}} < 0$ depends on the direction (sign) of the admissible $\delta\mathbf{u}$.

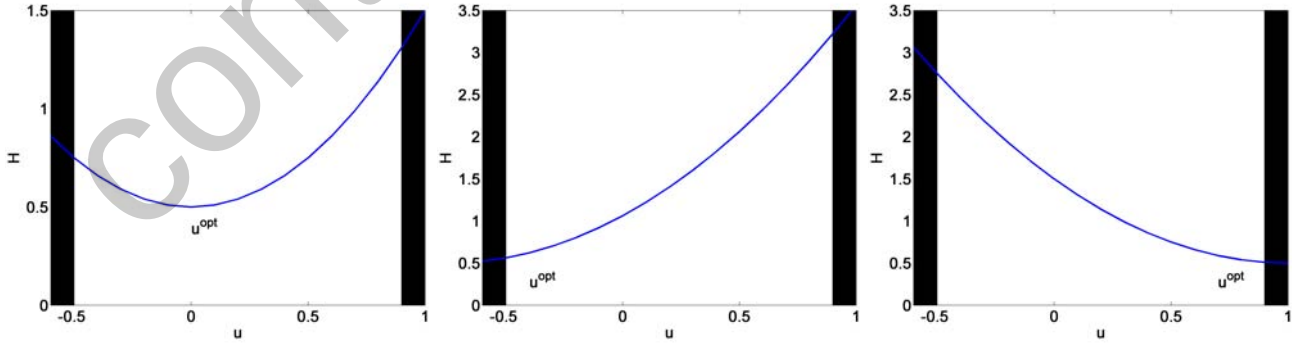


Figure 9.1: Examples of options for $\delta H = H_{\mathbf{u}}(t)\delta\mathbf{u}(t)$. Left: unconstrained min, so need $H_{\mathbf{u}} = 0$. Middle: constraint on left, so at min value, must have $\delta u \geq 0$ $\Rightarrow$ need $H_{\mathbf{u}} \geq 0$ so that $\delta H \geq 0$. Right: constraint on right, so at min value, must have $\delta u \leq 0$ $\Rightarrow$ need $H_{\mathbf{u}} \leq 0$ so that $\delta H \geq 0$.

- The requirement that $\delta H \geq 0$ says that δH must be non-improving to the cost (recall trying to minimize the cost) over the set of possible $\delta \mathbf{u}$.
  - Can actually state a stronger condition: H must be minimized over the set of all possible $\mathbf{u}$

- Thus for control constrained problems, third necessary condition

$$H_{\mathbf{u}} = 0$$

must be replaced with a more general necessary condition

$$\mathbf{u}^\star(t) = \arg \left\{ \min_{\mathbf{u}(t) \in \mathcal{U}} H(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) \right\}$$

  - So must look at H and explicitly find the minimizing control inputs given the constraints - not as simple as just solving $H_{\mathbf{u}} = 0$
  - Known as **Pontryagin's Minimum Principle**
  - Handles "edges" as well, where the admissible values of $\delta \mathbf{u}$ are "inwards"

- PMP is very general and applies to all constrained control problems – will now apply it to a special case in which the performance and the constraints are linear in the control variables.

PMP Example: Control Constraints

- Consider simple system $y = G(s)u$, $G(s) = 1/s^2$ with $|u(t)| \leq u_m$
  - Motion of a rigid body with limited control inputs – can be used to model many different things
- Want to solve the **minimum time-fuel problem**

$$\min J = \int_0^{t_f} (1 + b|u(t)|)dt$$

  - The goal is to drive the state to the origin with minimum cost.
  - Typical of many spacecraft problems – $\int |u(t)|dt$ sums up the fuel used, as opposed to $\int u^2(t)dt$ that sums up the power used.

- Define $x_1 = y$, $x_2 = \dot{y} \Rightarrow$ dynamics are $\dot{x}_1 = x_2$, $\dot{x}_2 = u$

- First consider the response if we apply ± 1 as the input. Note:
  - If $u = 1$, $x_2(t) = t + c_1$ and

$$x_1(t) = 0.5t^2 + c_1 t + c_2 = 0.5(t + c_1)^2 + c_3 = 0.5x_2(t)^2 + c_3$$

  - If $u = -1$, $x_2(t) = -t + c_4$ and

$$x_1(t) = -0.5t^2 + c_4 t + c_5 = -0.5(t + c_4)^2 + c_6 = -0.5x_2(t)^2 + c_6$$

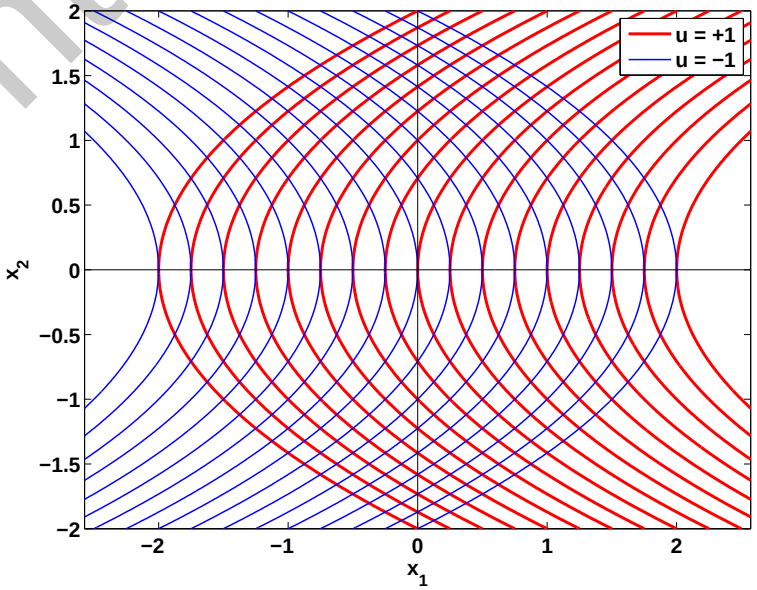


Figure 9.2: Possible response curves – what is the direction of motion?

- Hamiltonian for the system is:

$$\begin{aligned} H &= 1 + b|u| + \begin{bmatrix} p_1 & p_2 \end{bmatrix} \left\{ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \right\} \\ &= 1 + b|u| + p_1 x_2 + p_2 u \end{aligned}$$

- First find the equations for the co-state:

$$\dot{\mathbf{p}} = -H_{\mathbf{x}}^T \qquad \Rightarrow \qquad \begin{cases} \dot{p}_1 = -H_{x_1} = 0 & \rightarrow \quad p_1 = c_1 \\ \dot{p}_2 = -H_{x_2} = -p_1 & \rightarrow \quad p_2 = -c_1 t + c_2 \end{cases}$$

  - So p_2 is linear in time

- To find optimal control, look at the parts of H that depend on u:

$$\tilde{H} = b|u| + p_2 u$$

  - **Recall PMP:** given constraints, goal is to find u that minimizes H (or $\tilde{H}$)
  - Sum of two functions $|u|$ and u - sign of which depends on sign and relative size of p_2 compared to $b > 0$

- Three cases to consider (plots use $u_m = 1.5$):

  1. $p_2 > b > 0 \rightarrow$ choose $u^\star(t) = -u_m$

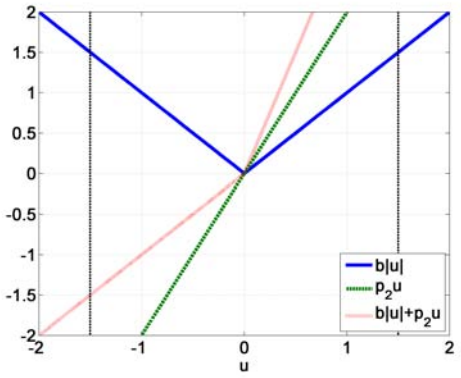


Figure 9.3: $b = 1, p_2 = 2$, so $p_2 > b > 0$
fopt1

  2. $p_2 < -b \rightarrow$ choose $u^\star(t) = u_m$

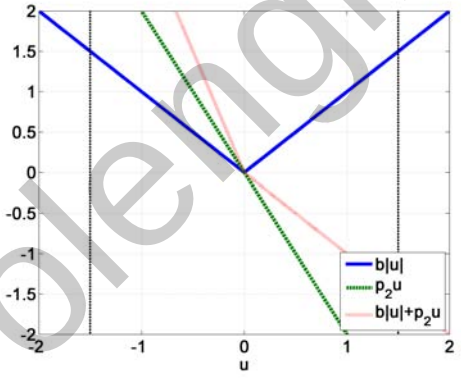


Figure 9.4: $b = 1, p_2 = -2$, so $p_2 < -b$

  3. $-b < p_2 < b \rightarrow$ choose $u^\star(t) = 0$

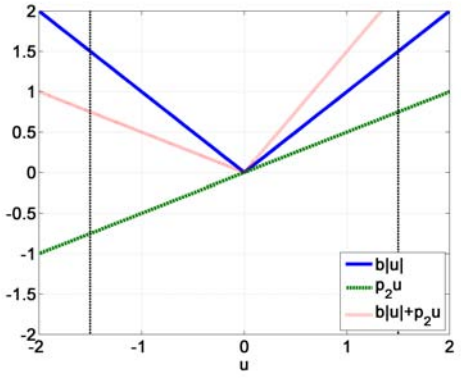


Figure 9.5: $b = 1, p_2 = 1$, so $-b < p_2 < b$

- The resulting control law is:

$$u(t) = \begin{cases} -u_m & b < p_2(t) \\ 0 & -b < p_2(t) < b \\ u_m & p_2(t) < -b \end{cases}$$

- So the control depends on $p_2(t)$ - but since it is a linear function of time, it is only possible to get at most 2 switches
  - Also, since $\dot{x}_2(t) = u$, and since we must stop at t_f, then must have that $u = \pm u_m$ at t_f

- To complete the solution, impose the boundary conditions (transversality condition), with $x_2(t_f) = 0$

$$H(t_f) + h_t(t_f) = 0 \quad \rightarrow \quad 1 + b|u(t_f)| + p_2(t_f)u(t_f) = 0$$

  - If $u = u_m$, then $1 + bu_m + p_2(t_f)u_m = 0$ implies that

$$p_2(t_f) = -(b + \frac{1}{u_m}) < -b$$

    which is consistent with the selection rules.
  - And if $u = -u_m$, then $1 + bu_m - p_2(t_f)u_m = 0$ implies that

$$p_2(t_f) = (b + \frac{1}{u_m}) > b$$

    which is also consistent.
  - So the terminal condition does not help us determine if $u = \pm u_m$, since it could be either

- So first look at the case where $u(t_f) = u_m$. Know that

$$p_2(t) = c_2 - c_1 t$$

and $p_2(t_f) = -(b + \frac{1}{u_m}) < -b$.

– Assume that $c_1 > 0$ so that we get some switching.

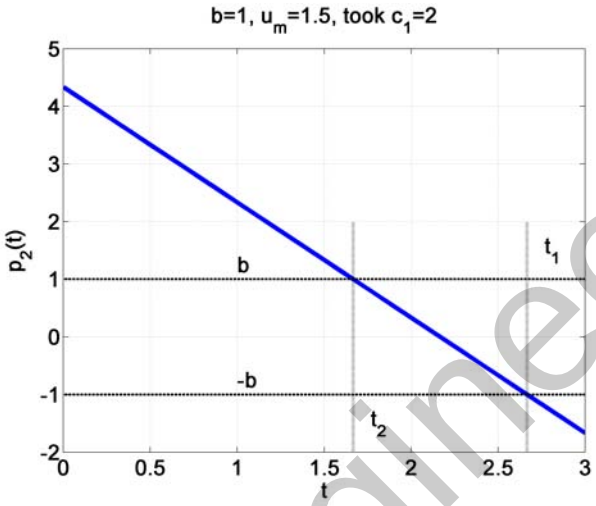


Figure 9.6: Possible switching case, but both t_f and c_1 are unknown at this point.

– Then set $p_2(t_1) = -b$ to get that $t_1 = t_f - 1/(u_m c_1)$

– And $p_2(t_2) = b$ gives $t_2 = t_f - (2b + 1/u_m)/c_1$

- Now look at the state response:

– Starting at the end: $\ddot{y} = u_m$, gives $y(t) = u_m/2t^2 + c_3 t + c_4$, where $\dot{y} = y = 0$ at t_f gives us that $c_3 = -u_m t_f$ and $c_4 = u_m/2t_f^2$, so

$$y(t) = \frac{u_m}{2}t^2 - u_m t_f t + \frac{u_m}{2}t_f^2 = \frac{u_m}{2}(t - t_f)^2$$

– But since $\dot{y}(t) = u_m t + c_3 = u_m(t - t_f)$, then $\boxed{y(t) = \dfrac{\dot{y}(t)^2}{2u_m}}$

– State response associated with $u = u_m$ is in lower right quadrant of the $y/\dot{y}$ phase plot

- Between times t_2–t_1, control input is zero $\Rightarrow$ **coasting phase**.
  - Terminal condition for coast same as the start of the next one:

$$y(t_1) = \frac{u_m}{2}(t_1 - t_f)^2 = \frac{1}{2u_m c_1^2}$$

    and $\dot{y}(t_1) = -1/c_1$
  - On a coasting arc, $\dot{y}$ is a constant (so $\dot{y}(t_2) = -1/c_1$), and thus

$$y(t_2) - \frac{(t_1 - t_2)}{c_1} = \frac{1}{2u_m c_1^2}$$

    which gives that

$$\begin{aligned} y(t_2) &= \frac{1}{2u_m c_1^2} + \frac{1}{c_1}\left(t_f - \frac{1}{u_m c_1} - (t_f - (\frac{2b}{c_1} + \frac{1}{u_m c_1}))\right) \\ &= (2b + \frac{1}{2u_m})\frac{1}{c_1^2} = (2b + \frac{1}{2u_m})\dot{y}(t_2)^2 \end{aligned}$$

- So the first transition occurs along the curve $\boxed{y(t) = (2b + \frac{1}{2u_m})\dot{y}(t)^2}$
- For the first arc, things get a bit more complicated.
  Clearly $u(t) = -u_m$, with IC $y_0,\ \dot{y}_0$ so

$$\begin{aligned} \dot{y}(t) &= -u_m t + c_5 = -u_m t + \dot{y}_0 \\ y(t) &= -\frac{u_m}{2}t^2 + c_5 t + c_6 = -\frac{u_m}{2}t^2 + \dot{y}_0 t + y_0 \end{aligned}$$

  - Now project forward to t_2

$$\begin{aligned} \dot{y}(t_2) &= -u_m t_2 + \dot{y}_0 = \dot{y}(t_1) = -\frac{1}{c_1} \ \rightarrow\ c_1 = \frac{2(b + 1/u_m)}{t_f - \dot{y}_0/u_m} \\ y(t_2) &= -\frac{u_m}{2}t_2^2 + \dot{y}_0 t_2 + y_0 \end{aligned}$$

    and use these expressions in the quadratic for the switching curve to solve for c_1, t_1, t_2

- The solutions have a very distinctive **Bang–Off–Bang** pattern
  - Two parabolic curves define switching from $+u_m$ to 0 to $-u_m$

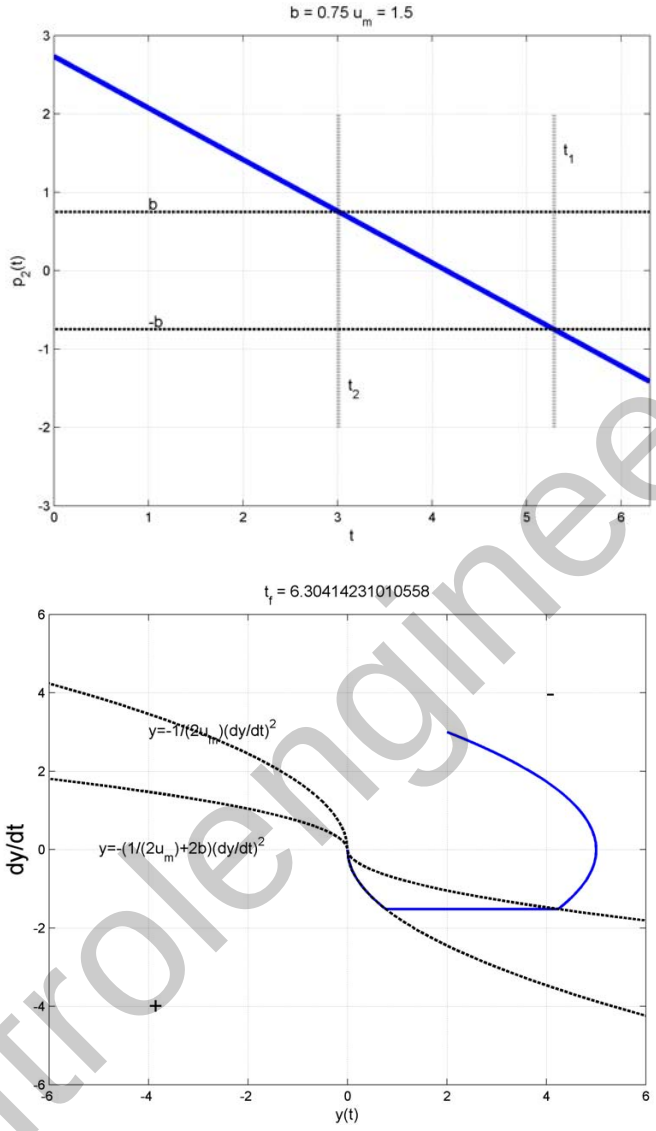


Figure 9.7: $y_0 = 2$ $\dot{y}_0 = 3$ $b = 0.75$ $u_m = 1.5$

- Switching control was derived using a detailed evaluation of the state and costate
  - But final result is a switching law that can be written wholly in terms of the system states.

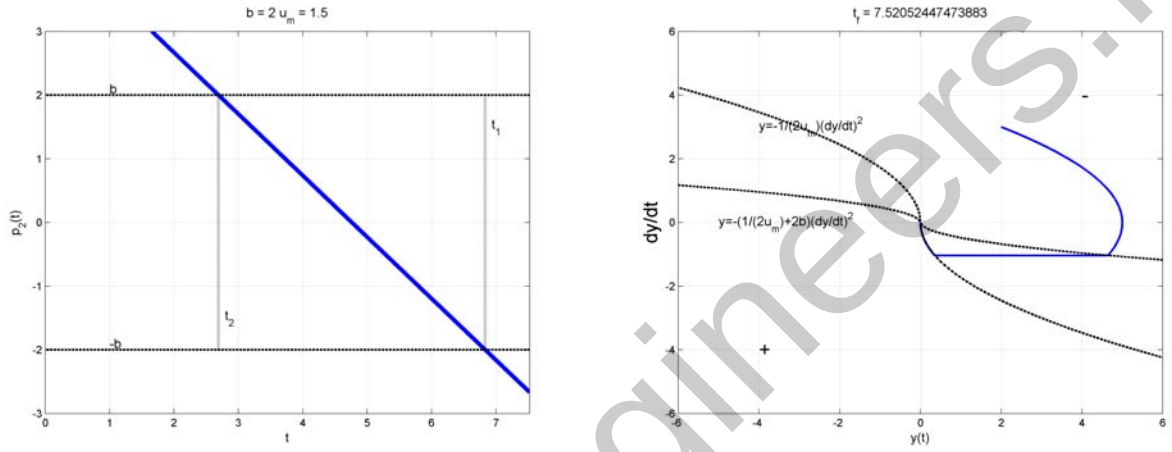


Figure 9.8: $y_0 = 2\ \dot{y}_0 = 3\ b = 2\ u_m = 1.5$

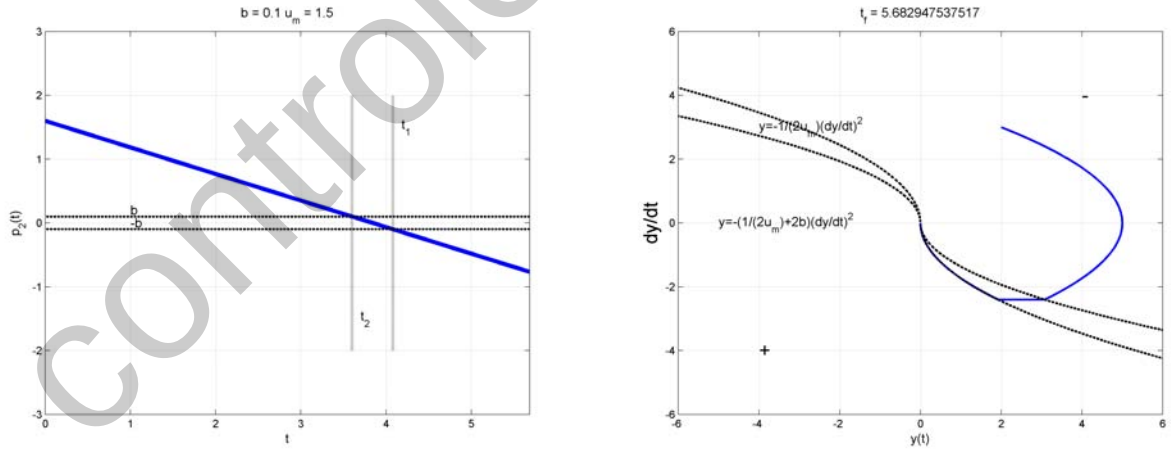


Figure 9.9: $y_0 = 2\ \dot{y}_0 = 3\ b = 0.1\ u_m = 1.5$

- Clearly get a special result as $b \to 0$, which is the solution to the **minimum time problem**
  - Control inputs are now just **Bang–Bang**
  - One parabolic curve defines switching from $+u_m$ to $-u_m$

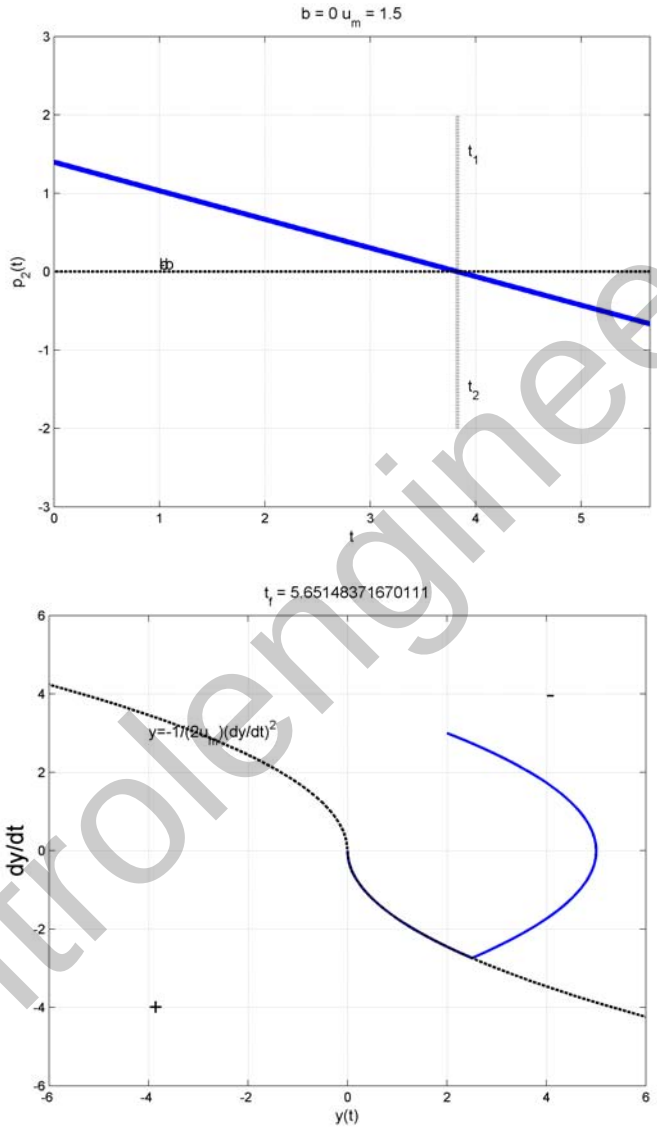


Figure 9.10: Min time: $y_0 = 2$ $\dot{y}_0 = 3$ $b = 0$ $u_m = 1.5$

- Can show that the switching and final times are given by

$$t_1 = \dot{y}(0) + \sqrt{y(0) + 0.5\dot{y}^2(0)} \qquad t_f = \dot{y}(0) + 2\sqrt{y(0) + 0.5\dot{y}^2(0)}$$

- **Trade-off:** coasting is fuel efficient, but it takes a long time.

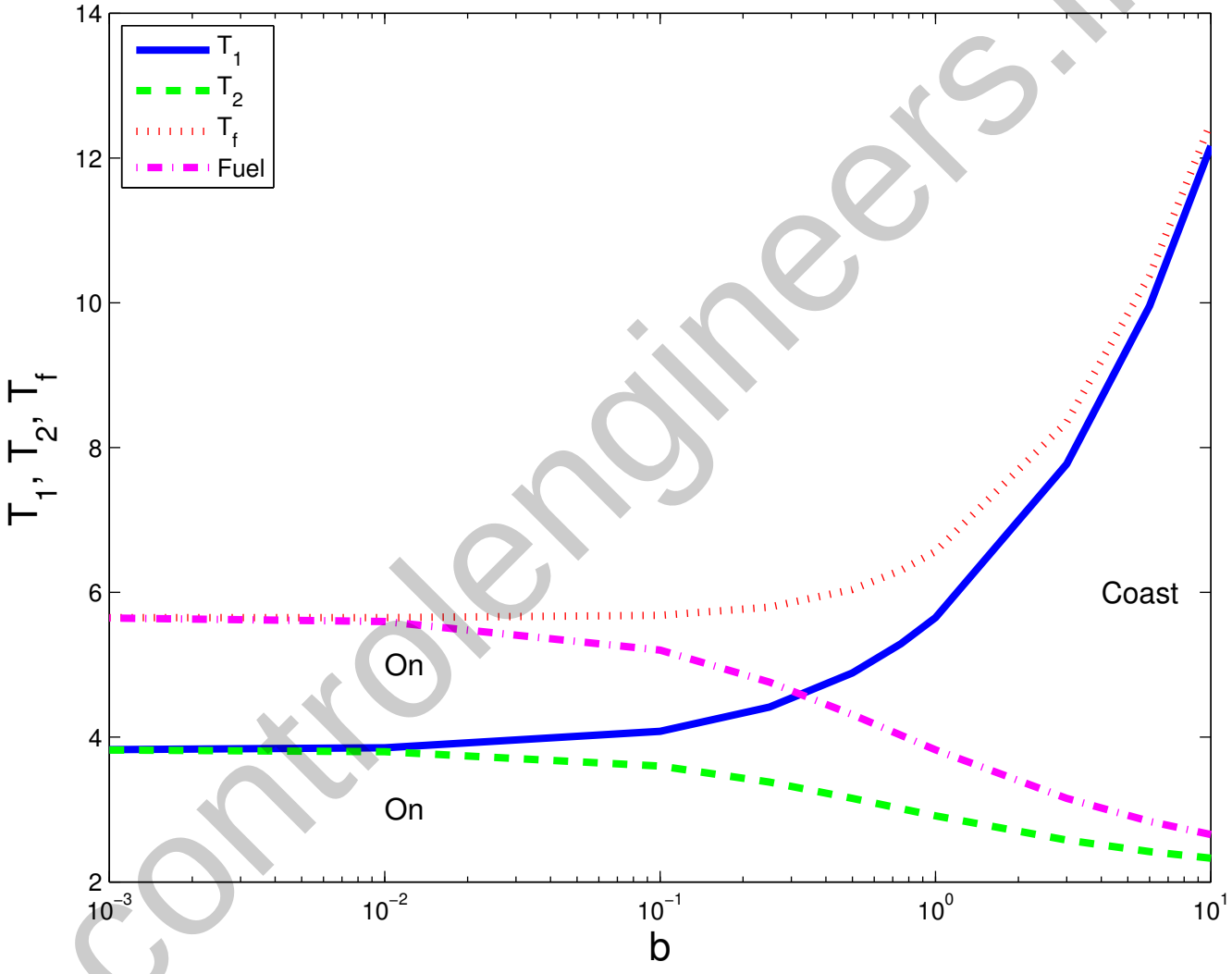


Figure 9.11: Summary of switching times for various fuel weights

Min time fuel

```
% Min time fuel for double integrator
% 16.323 Spring 2008
% Jonathan How
figure(1);clf;%
if jcase==1;y0=2;yd0=3; b=.75;u_m=1.5;% baseline
elseif jcase==2;y0=2;yd0=3; b=2;u_m=1.5;% fuel exp
elseif jcase==3;y0=2;yd0=3; b=.1;u_m=1.5;% fuel cheap
elseif jcase==4;y0=2;yd0=3; b=0;u_m=1.5;% min time
elseif jcase==5;y0=-4;yd0=4; b=1;u_m=1.5;% min time
end

% Tf is unknown - put together the equations to solve for it
alp=(1/2/u_m+2*b) % switching line
% middle of 8--6: t_2 as a ftn of t_f
T2=[1/u_m (2*b+1/u_m)*yd0/u_m]/(2*b+2/u_m);%
% bottom of 8--7: quadratic for y(t_2) in terms of t_2
% converted into quad in t_f
T_f=roots(-u_m/2*conv(T2,T2)+yd0*[0 T2]+[0 0 y0]  - ...
alp*conv(-u_m*T2+[0 yd0],-u_m*T2+[0 yd0]));%
t_f=max(T_f);t=[0:.01:t_f]'; %

c_1=(2*b+2/u_m)/(t_f-yd0/u_m);% key parameters for p(t)
c_2=c_1*t_f-(b+1/u_m);% key parameters for p(t)
t_1=t_f-1/(u_m*c_1); t_2=t_f-(2*b+1/u_m)/c_1;%switching times

G=ss([0 1;0 0],[0 1]',eye(2),zeros(2,1));
arc1=[0:.001:t_2]'; arc2=[t_2:.001:t_1]';arc3=[t_1:.001:t_f]'; %
if jcase==4;arc2=[t_2 t_2+1e-6]';end
[Y1,T1,X1]=lsim(G,-u_m*ones(length(arc1),1),arc1,[y0 yd0]'); %
[Y2,T2,X2]=lsim(G,0*ones(length(arc2),1),arc2,Y1(end,:)'); %
[Y3,T3,X3]=lsim(G,u_m*ones(length(arc3),1),arc3,Y2(end,:)'); %
plot(Y1(:,1),Y1(:,2),'Linewidth',2); hold on%
plot(Y2(:,1),Y2(:,2),'Linewidth',2); plot(Y3(:,1),Y3(:,2),'Linewidth',2);%
ylabel('dy/dt','Fontsize',18); xlabel('y(t)','Fontsize',12);%
text(-4,3,'y=-1/(2u_m)(dy/dt)^2','Fontsize',12)%
if jcase ~= 4; text(-5,0,'y=-(1/(2u_m)+2b)(dy/dt)^2','Fontsize',12);end
text(4,4,'-','Fontsize',18);text(-4,-4,'+','Fontsize',18);grid;hold off
title(['t_f = ',mat2str(t_f)],'Fontsize',12)%

hold on;% plot the switching curves
if jcase ~= 4;kk=[0:.1:5]'; plot(-alp*kk.^2,kk,'k--','Linewidth',2);plot(alp*kk.^2,-kk,'k--','Linewidth',2);end
kk=[0:.1:5]';plot(-(1/(2*u_m))*kk.^2,kk,'k--','Linewidth',2);plot((1/(2*u_m))*kk.^2,-kk,'k--','Linewidth',2);%
hold off;axis([-4 4 -4 4]/4*6);

figure(2);p2=c_2-c_1*t;%
plot(t,p2,'Linewidth',4);%
hold on; plot([0 max(t)],[b b],'k--','Linewidth',2);hold off; %
hold on; plot([0 max(t)],-[b b],'k--','Linewidth',2);hold off; %
hold on; plot([t_1 t_1],[-2 2],'k:','Linewidth',3);hold off; %
text(t_1+.1,1.5,'t_1','Fontsize',12)%
hold on; plot([t_2 t_2],[-2 2],'k:','Linewidth',3);hold off; %
text(t_2+.1,-1.5,'t_2','Fontsize',12)%
title(['b = ',mat2str(b),' u_m = ',mat2str(u_m)],'Fontsize',12);%
ylabel('p_2(t)','Fontsize',12); xlabel('t','Fontsize',12);%
text(1,b+.1,'b','Fontsize',12);text(1,-b+.1,'-b','Fontsize',12)%
axis([0 t_f -3 3]);grid on; %
%
if jcase==1
print -f1 -dpng -r300 fopt5a.png;;print -f2 -dpng -r300 fopt5b.png;
elseif jcase==2
print -f1 -dpng -r300 fopt6a.png;print -f2 -dpng -r300 fopt6b.png;
elseif jcase==3
print -f1 -dpng -r300 fopt7a.png;print -f2 -dpng -r300 fopt7b.png;
elseif jcase==4
print -f1 -dpng -r300 fopt8a.png;print -f2 -dpng -r300 fopt8b.png;
end
```

Minimum Time Problems

- Can repeat this analysis for minimum time and energy problems using the PMP
  - Issue is that the process of a developing a solution by analytic construction is laborious and very hard to extend to anything nonlinear and/or linear with more than 2 states
- Need to revisit the problem statement and develop a new approach.
- **Goal:** develop the control input sequence

$$M_i^- \leq u_i(t) \leq M_i^+$$

that drives the system (nonlinear, but linear control inputs)

$$\dot{\mathbf{x}} = A(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u}$$

from an arbitrary state $\mathbf{x}_0$ to the origin to minimize maneuver time

$$\min J = \int_{t_0}^{t_f} dt$$

- **Solution:** form the Hamiltonian

$$\begin{aligned} H &= 1 + \mathbf{p}^T(t)\{A(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u}\} \\ &= 1 + \mathbf{p}^T(t)\{A(\mathbf{x},t) + \left[\begin{array}{cccc} \mathbf{b}_1(\mathbf{x},t) & \mathbf{b}_2(\mathbf{x},t) & \cdots & \mathbf{b}_m(\mathbf{x},t) \end{array}\right]\mathbf{u}\} \\ &= 1 + \mathbf{p}^T(t)A(\mathbf{x},t) + \sum_{i=1}^{m} \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x},t)u_i(t) \end{aligned}$$

- Now use the PMP: select $u_i(t)$ to minimize H, which gives

$$u_i(t) = \begin{cases} M_i^+ & \texttt{if} \quad \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x},t) < 0 \\ \\ M_i^- & \texttt{if} \quad \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x},t) > 0 \end{cases}$$

which gives us the expected **Bang-Bang** control

- Then solve for the costate

$$\dot{\mathbf{p}} = -H_{\mathbf{x}}^T = -\left(\frac{\partial A}{\partial \mathbf{x}} + \frac{\partial B}{\partial \mathbf{x}}u\right)^T \mathbf{p}$$

  - Could be very complicated for a nonlinear system.

- Note: shown how to pick $u(t)$ given that $\mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x},t) \neq 0$
  - Not obvious what to do if $\mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x},t) = 0$ for some finite time interval.
  - In this case the coefficient of $u_i(t)$ is zero, and PMP provides no information on how to pick the control inputs.
  - Will analyze this **singular condition** in more detail later.

- To develop further insights, restrict the system model further to LTI, so that

$$A(\mathbf{x},t) \to A\mathbf{x} \qquad B(\mathbf{x},t) \to B$$

  - Assume that $[A, B]$ controllable
  - Set $M_i^+ = -M_i^- = u_{m_i}$

- Just showed that if a solution exists, it is Bang-Bang
  - **Existence:** if $\mathbb{R}(\lambda_i(A)) \leq 0$, then an optimal control exists that transfers any initial state $\mathbf{x}_0$ to the origin.
    - ◇ Must eliminate unstable plants from this statement because the control is bounded.
  - **Uniqueness:** If an extremal control exists (i.e. solves the necessary condition and satisfies the boundary conditions), then it is unique.
    - ◇ Satisfaction of the PMP is both necessary and sufficient for time-optimal control of a LTI system.

- If the eigenvalues of A are all real, and a unique optimal control exists, **then each control input can switch at most $n-1$ times.**
  - Still need to find the costates to determine the switching times – but much easier in the linear case.

Min Fuel Problems

- **Goal:** develop the control input sequence

$$M_i^- \leq u_i(t) \leq M_i^+$$

that drives the system

$$\dot{\mathbf{x}} = A(\mathbf{x}, t) + B(\mathbf{x}, t)\mathbf{u}$$

from an arbitrary state $\mathbf{x}_0$ to the origin in a fixed time t_f and optimizes the cost

$$\min J = \int_{t_0}^{t_f} \sum_{i=1}^{m} c_i |u_i(t)| dt$$

- **Solution:** form the Hamiltonian

$$\begin{aligned} H &= \sum_{i=1}^{m} c_i |u_i(t)| + \mathbf{p}^T(t)\{A(\mathbf{x}, t) + B(\mathbf{x}, t)\mathbf{u}\} \\ &= \sum_{i=1}^{m} c_i |u_i(t)| + \mathbf{p}^T(t)A(\mathbf{x}, t) + \sum_{i=1}^{m} \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x}, t)u_i(t) \\ &= \sum_{i=1}^{m} \left[c_i |u_i(t)| + \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x}, t)u_i(t)\right] + \mathbf{p}^T(t)A(\mathbf{x}, t) \end{aligned}$$

- Use the PMP, which requires that we select $u_i(t)$ to ensure that for all admissible $u_i(t)$

$$\sum_{i=1}^{m} \left[c_i |u_i^\star(t)| + \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x}, t)u_i^\star(t)\right] \leq \sum_{i=1}^{m} \left[c_i |u_i(t)| + \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x}, t)u_i(t)\right]$$

- If the components of $\mathbf{u}$ are independent, then can just look at

$$c_i |u_i^\star(t)| + \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x}, t)u_i^\star(t) \leq c_i |u_i(t)| + \mathbf{p}^T(t)\mathbf{b}_i(\mathbf{x}, t)u_i(t)$$

  - As before, this boils down to a comparison of c_i and $\mathbf{p}^T(t)\mathbf{b}_i$
  - Resulting control law is:

$$u_i^\star(t) = \begin{cases} M_i^- & \texttt{if} \quad c_i < \mathbf{p}^T(t)\mathbf{b}_i \\ 0 & \texttt{if} \quad -c_i < \mathbf{p}^T(t)\mathbf{b}_i < c_i \\ M_i^+ & \texttt{if} \quad \mathbf{p}^T(t)\mathbf{b}_i < -c_i \end{cases}$$

Example: 9–1

- Consider $G(s) = 1/s^2 \Rightarrow A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$$\min J = \int_{t_0}^{t_f} |u(t)| dt$$

  - Drive state to the origin with t_f fixed.

- Gives $H = |u| + p_1 x_2 + p_2 u$
  - Final control $u(t_f) = u_m \Rightarrow p_2(t_f) < -1$ $p_2(t) = c_2 - c_1 t$

- As before, integrate EOM forward from 0 to t_2 using $-u_m$, then from t_2 to t_1 using $u = 0$, and from t_1 to t_f using u_m
  - Apply terminal conditions and solve for c_1 and c_2

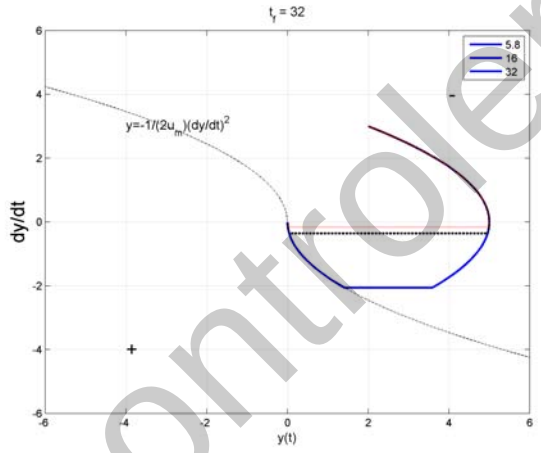


Figure 9.12: Min Fuel for varying final times

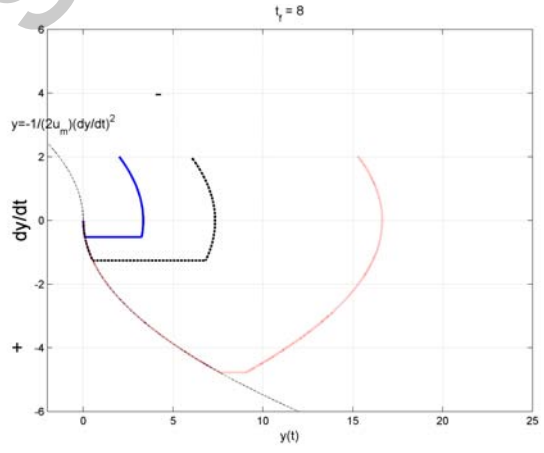


Figure 9.13: Min fuel for fixed final time, varying IC's

- First switch depends on IC and $t_f \Rightarrow$ no clean closed-form solution for switching curve
  - Larger t_f leads to longer coast.
  - For given t_f, there is a limit to the IC from which we can reach the origin.

- If specified completion time $t_f > T_{\min} = \dot{y}(0) + 2\sqrt{y(0) + 0.5\dot{y}^2(0)}$, then

$$t_2 = 0.5\left\{(\dot{y}(0) + t_f) - \sqrt{(\dot{y}(0) - t_f)^2 - (4y(0) + 2\dot{y}^2(0))}\right\}$$

$$t_1 = 0.5\left\{(\dot{y}(0) + t_f) + \sqrt{(\dot{y}(0) - t_f)^2 - (4y(0) + 2\dot{y}^2(0))}\right\}$$

Minimum Energy Problem

- Goal: for a fixed final time and terminal constraints

$$\min J = \frac{1}{2}\int_0^{t_f} \mathbf{u}^T R\mathbf{u}\, dt \qquad R > 0$$

- Again use special dynamics:

$$\begin{aligned}\dot{\mathbf{x}} &= A(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u}\\ H &= \frac{1}{2}\mathbf{u}^T R\mathbf{u} + \mathbf{p}^T\{\, A(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u}\}\end{aligned}$$

- Obviously with no constraints on $\mathbf{u}$, solve $H_{\mathbf{u}} = 0$, to get

$$\mathbf{u} = -R^{-1}B^T\mathbf{p}(t)$$

- But with bounded controls, must solve:

$$\mathbf{u}^\star(t) = \arg\min_{\mathbf{u}(t)\in\mathcal{U}}\left[\frac{1}{2}\mathbf{u}^T R\mathbf{u} + \mathbf{p}^T B(\mathbf{x},t)\mathbf{u}\right]$$

  which is a constrained quadratic program in general
  - However, for diagonal R, the effects of the controls are independent

$$\mathbf{u}^\star(t) = \arg\min_{\mathbf{u}(t)\in\mathcal{U}}\left[\sum_{i=1}^{m}\frac{1}{2}R_{ii}u_i^2 + \mathbf{p}^T\mathbf{b}_i u_i\right]$$

  - In the unconstrained case, each $u_i(t)$ can easily be determined by minimizing

$$\frac{1}{2}R_{ii}u_i^2 + \mathbf{p}^T\mathbf{b}_i u_i \qquad \rightarrow \qquad \tilde{u}_i = -R_{ii}^{-1}\mathbf{p}^T\mathbf{b}_i$$

- The resulting controller inputs are $u_i(t) = \mathtt{sat}(\tilde{u}_i(t))$

$$u_i(t) = \begin{cases} M_i^- & \mathtt{if} \quad \tilde{u}_i < M_i^- \\ \tilde{u}_i & \mathtt{if} \quad M_i^- < \tilde{u}_i < M_i^+ \\ M_i^+ & \mathtt{if} \quad M_i^+ < \tilde{u}_i \end{cases}$$

Min Fuel

```
% Min fuel for double integrator
% 16.323 Spring 2008
% Jonathan How
%
c=1;
t=[0:.01:t_f];
alp=(1/2/u_m) % switching line
T_2=roots([-u_m/2 yd0 y0] + conv([-u_m yd0],[-2 t_f+yd0/u_m])-alp*conv([-u_m yd0],[-u_m yd0]));%
t_2=min(T_2);
yd2=-u_m*t_2+yd0;yd1=yd2;
t_1=t_f+yd1/u_m;
c_1=2/(t_1-t_2);c_2=c_1*t_1-1;

G=ss([0 1;0 0],[0 1]',eye(2),zeros(2,1));
arc1=[0:.001:t_2]'; arc2=[t_2:.001:t_1]';arc3=[t_1:.001:t_f]'; %
[Y1,T1,X1]=lsim(G,-u_m*ones(length(arc1),1),arc1,[y0 yd0]'); %
[Y2,T2,X2]=lsim(G,0*ones(length(arc2),1),arc2,Y1(end,:)'); %
[Y3,T3,X3]=lsim(G,u_m*ones(length(arc3),1),arc3,Y2(end,:)'); %
plot(Y1(:,1),Y1(:,2),zzz,'Linewidth',2); hold on%
plot(Y2(:,1),Y2(:,2),zzz,'Linewidth',2); plot(Y3(:,1),Y3(:,2),zzz,'Linewidth',2);%
ylabel('dy/dt','Fontsize',18); xlabel('y(t)','Fontsize',12);%
text(-4,3,'y=-1/(2u_m)(dy/dt)^2','Fontsize',12)%
text(4,4,'-','Fontsize',18);text(-4,-4,'+','Fontsize',18);grid on;hold off
title(['t_f = ',mat2str(t_f)],'Fontsize',12)%

hold on;% plot the switching curves
kk=[0:.1:8]'; plot(-alp*kk.^2,kk,'k--');plot(alp*kk.^2,-kk,'k--');
hold off;axis([-4 4 -4 4]/4*6);

figure(2);%
p2=c_2-c_1*t;%
plot(t,p2,'Linewidth',4);%
hold on; plot([0 t_f],[c c],'k--','Linewidth',2);hold off; %
hold on; plot([0 t_f],-[c c],'k--','Linewidth',2);hold off; %
hold on; plot([t_1 t_1],[-2 2],'k:','Linewidth',3);hold off; %
text(t_1+.1,1.5,'t_1','Fontsize',12)%
hold on; plot([t_2 t_2],[-2 2],'k:','Linewidth',3);hold off; %
text(t_2+.1,-1.5,'t_2','Fontsize',12)%
title(['c = ',mat2str(c),' u_m = ',mat2str(u_m)],'Fontsize',12);%
ylabel('p_2(t)','Fontsize',12); xlabel('t','Fontsize',12);%
text(1,c+.1,'c','Fontsize',12);text(1,-c+.1,'-c','Fontsize',12)%
axis([0 t_f -3 3]);grid on; %

return

figure(1);clf
y0=2;yd0=3;t_f=5.8;u_m=1.5;zzz='-';minu;
figure(1);hold on
y0=2;yd0=3;t_f=16;u_m=1.5;zzz='k--';minu;
figure(1);hold on
y0=2;yd0=3;t_f=32;u_m=1.5;zzz='r:';minu;
figure(1);
axis([-6 6 -6 6])
legend('5.8','16','32')
print -f1 -dpng -r300 uopt1.png;


figure(1);clf
y0=2;yd0=2;t_f=8;u_m=1.5;zzz='-';minu
figure(1);hold on
y0=6;yd0=2;t_f=8;u_m=1.5;zzz='k--';minu
figure(1);hold on
y0=15.3;yd0=2;t_f=8;u_m=1.5;zzz='r:';minu
figure(1);axis([-2 25 -6 6])
print -f1 -dpng -r300 uopt2.png;
```

16.323 Lecture 10

Singular Arcs

- Bryson Chapter 8
- Kirk Section 5.6

Singular Problems

- There are occasions when the PMP

$$\mathbf{u}^\star(t) = \arg\left\{ \min_{\mathbf{u}(t)\in\mathcal{U}} H(\mathbf{x},\mathbf{u},\mathbf{p},t) \right\}$$

  fails to define $\mathbf{u}^\star(t)$ $\Rightarrow$ can an extremal control still exist?
  - Typically occurs when the Hamiltonian is linear in the control, and the coefficient of the **control term equals zero**.

- Example: on page 9-10 we wrote the control law:

$$u(t) = \begin{cases} -u_m & b < p_2(t) \\ 0 & -b < p_2(t) < b \\ u_m & p_2(t) < -b \end{cases}$$

  but we do not know what happens if $p_2 = b$ for an interval of time.
  - Called a **singular arc**.
  - Bottom line is that the straightforward solution approach does not work, and we need to investigate the PMP conditions in more detail.

- **Key point**: depending on the system and the cost, singular arcs might exist, and we must determine their existence to fully characterize the set of possible control solutions.

- Note: control on the singular arc is determined by the requirements that the coefficient of the linear control terms in $H_{\mathbf{u}}$ remain zero on the singular arc and so must the time derivatives of $H_{\mathbf{u}}$.
  - Necessary condition for scalar u can be stated as

$$(-1)^k \frac{\partial}{\partial u}\left[\left(\frac{d^{2k}}{dt^{2k}}\right) H_u\right] \geq 0 \quad k = 0, 1, 2 \ldots$$

Singular Arc Example 1

- With $\dot{x} = u$, $x(0) = 1$ and $0 \le u(t) \le 4$, consider objective

$$\min \int_0^2 (x(t) - t^2)^2 dt$$

- First form standard Hamiltonian

$$H = (x(t) - t^2)^2 + p(t)u(t)$$

which gives $H_u = p(t)$ and

$$\dot{p}(t) = -H_x = -2(x - t^2), \qquad \text{with } p(2) = 0 \qquad (10.15)$$

- Note that if $p(t) > 0$, then PMP indicates that we should take the minimum possible value of $u(t) = 0$.
  - Similarly, if $p(t) < 0$, we should take $u(t) = 4$.

- Question: can we get that $H_u \equiv 0$ for some interval of time?
  - Note: $H_u \equiv 0$ implies $p(t) \equiv 0$, which means $\dot{p}(t) \equiv 0$, and thus

$$\dot{p}(t) \equiv 0 \Rightarrow x(t) = t^2, \quad u(t) = \dot{x} = 2t$$

- Thus we get the control law that

$$u(t) = \begin{cases} 0 & p(t) > 0 \\ 2t & \text{when } p(t) = 0 \\ 4 & p(t) < 0 \end{cases}$$

- Can show by contradiction that optimal solution has $x(t) \geq t^2$ for $t \in [0, 2]$.
  - And thus we know that $\dot{p}(t) \leq 0$ for $t \in [0, 2]$
  - But $p(2) = 0$ and $\dot{p}(t) \leq 0$ imply that $p(t) \geq 0$ for $t \in [0, 2]$

- So there must be a point in time $k \in [0, 2]$ after which $p(t) = 0$ (some steps skipped here...)
  - Check options: $k = 0$? $\Rightarrow$ contradiction
  - Check options: $k = 2$? $\Rightarrow$ contradiction

- So must have $0 < k < 2$. How find it? Control law will be

$$u(t) = \begin{cases} 0 & \text{when } 0 \leq t < k \\ 2t & k \leq t < 2 \end{cases}$$

apply this control to the state equations and get:

$$x(t) = \begin{cases} 1 & \text{when } 0 \leq t \leq k \\ t^2 + (1 - k^2) & k \leq t \leq 2 \end{cases}$$

To find k, note that must have $p(t) \equiv 0$ for $t \in [k, 2]$, so in this time range

$$\dot{p}(t) \equiv 0 = -2(1 - k^2) \quad \Rightarrow \quad k = 1$$

  - So now both $u(t)$ and $x(t)$ are known, and the optimal solution is to "bang off" and then follow a singular arc.

Singular Arc Example 2

- LTI system, $x_1(0)$, $x_2(0)$, t_f given; $x_1(t_f) = x_2(t_f) = 0$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

  and $J = \frac{1}{2}\int_0^{t_f} x_1^2 dt$ (see Bryson and Ho, p. 248)

- So $H = \frac{1}{2}x_1(t)^2 + p_1(t)x_2(t) + p_1(t)u(t) - p_2(t)u(t)$

$$\Rightarrow \quad \dot{p}_1(t) = -x_1(t), \qquad \dot{p}_2(t) = -p_1(t)$$

- For a singular arc, we must have $H_u = 0$ for a finite time interval

$$H_u = p_1(t) - p_2(t) = 0?$$

- Thus, during that interval

$$\begin{aligned} \frac{d}{dt}H_u &= \dot{p}_1(t) - \dot{p}_2(t) \\ &= -x_1(t) + p_1(t) = 0 \end{aligned}$$

- Note that H is not an explicit function of time, so H is a **constant for all time**

$$H = \frac{1}{2}x_1(t)^2 + p_1(t)x_2(t) + [p_1(t) - p_2(t)]\,u(t) = C$$

  but can now substitute from above along the singular arc

$$\frac{1}{2}x_1(t)^2 + x_1(t)x_2(t) = C$$

  which gives a family of singular arcs in the state x_1, x_2

- To find the appropriate control to stay on the arc, use

$$\frac{d^2}{dt^2}(H_u) = -\dot{x}_1 + \dot{p}_1 = -(x_2(t) + u(t)) - x_1(t) = 0$$

  or that $u(t) = -(x_1(t) + x_2(t))$ which is a linear feedback law to use along the singular arc.

Details for LTI Systems

- Consider the min time-fuel problem for the general system

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

with $M^- \leq u_i \leq M^+$ and

$$J = \int_0^{t_f} (1 + \sum_{i=1}^{m} c_i |u_i|) dt$$

t_f is free and we want to drive the state to the origin

- We studied this case before, and showed that

$$H = 1 + \sum_{i=1}^{m} (c_i |u_i| + \mathbf{p}^T B_i u_i) + \mathbf{p}^T A\mathbf{x}$$

- On a singular arc, $\frac{d^k}{dt^k}(H_u) = 0 \Rightarrow$ coefficient of u in H is zero

$$\Rightarrow \mathbf{p}^T(t) B_i = \pm c_i$$

for non-zero period of time and

$$\frac{d^k}{dt^k}(\mathbf{p}^T(t) B_i) = \left(\frac{d^k \mathbf{p}(t)}{dt^k} \right)^T B_i = 0 \quad \forall\ k \geq 1$$

- Recall the necessary conditions $\dot{\mathbf{p}}^T = -H_{\mathbf{x}} = -\mathbf{p}^T A$, which imply

$$\begin{aligned} \ddot{\mathbf{p}}^T &= -\dot{\mathbf{p}}^T A = \mathbf{p}^T A^2 \\ \dddot{\mathbf{p}}^T &= -\ddot{\mathbf{p}}^T A = -\mathbf{p}^T A^3 \\ &\vdots \\ \left(\frac{d^k \mathbf{p}(t)}{dt^k} \right)^T &\equiv (-1)^k \mathbf{p}^T A^k \end{aligned}$$

and combining with the above gives

$$\left(\frac{d^k \mathbf{p}(t)}{dt^k} \right)^T B_i = (-1)^k \mathbf{p}^T A^k B_i = 0$$

- Rewriting these equations yields the conditions that

$$\mathbf{p}^T AB_i = 0, \qquad \mathbf{p}^T A^2 B_i = 0, \qquad \cdots$$
$$\Rightarrow \mathbf{p}^T A \left[\begin{array}{cccc} B_i & AB_i & \cdots & A^{n-1}B_i \end{array} \right] = 0$$

- There are three ways to get:

$$\mathbf{p}^T A \left[\begin{array}{cccc} B_i & AB_i & \cdots & A^{n-1}B_i \end{array} \right] = 0$$

- On a singular arc, we know that $\mathbf{p}(t) \neq 0$ so this does not cause the condition to be zero.

- What if A singular, and $\mathbf{p}(t)^T A = 0$ on the arc?
  - Then $\dot{\mathbf{p}}^T = -\mathbf{p}^T A = 0$. In this case, $\mathbf{p}(t)$ is constant over $[t_0, t_f]$
  - Indicates that if the problem is singular at any time, it is singular for all time.
  - This also indicates that $\mathbf{u}$ is a constant.
  - A possible case, but would be unusual since it is very restrictive set of control inputs.

- Third possibility is that $\left[\begin{array}{cccc} B_i & AB_i & \cdots & A^{n-1}B_i \end{array} \right]$ is singular, meaning that the system is not controllable by the individual control inputs.
  - Very likely scenario – most common cause of singularity conditions.
  - Lack of controllability by a control input does not necessarily mean that a singular arc has to exist, but it is a possibility.

- For **Min Time** problems, now $c_i = 0$, so things are a bit different

- In this case the switchings are at $\mathbf{p}^T B_i = 0$ and a similar analysis as before gives the condition that

$$\mathbf{p}^T \left[\begin{array}{cccc} B_i & AB_i & \cdots & A^{n-1}B_i \end{array} \right] = 0$$

- Now there are only 2 possibilities
  - $\mathbf{p} = 0$ is one, but in that case,

$$H = 1 + \mathbf{p}^T(A\mathbf{x} + B\mathbf{u}) = 1$$

    but we would expect that $H = 0$
  - Second condition is obviously the lack of controllability again.

- **Summary (Min time)**:
  - If the system is completely controllable by B_i, then u_i can have no singular intervals
  - Not shown, but if the system is not completely controllable by B_i, then u_i **must** have a singular interval.

- **Summary (Min time-fuel)**:
  - If the system is completely controllable by B_i and A is non-singular, then there can be no singular intervals

Nonlinear Systems

- Consider systems that are nonlinear in the state, but linear in the control

$$\dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t)) + \mathbf{b}(\mathbf{x}(t))\mathbf{u}(t)$$

with cost

$$J = \int_{t_0}^{t_f} \mathbf{g}(\mathbf{x}(t))dt$$

- For a singular arc, in general you will find that

$$\frac{d^k}{dt^k}\left(H_{u_i}\right) = 0 \qquad k = 0, \ldots, r-1$$

but these conditions provide no indication of the control required to keep the system on the singular arc

– i.e. the coefficient of the control terms is zero.

- But then for some r and i, $\frac{d^r}{dt^r}\left(H_{u_i}\right) = 0$ does retain u_i.

– So if $\mathbf{u}_j(\mathbf{x}, \mathbf{p})$ are the other control inputs, then

$$\frac{d^r}{dt^r}\left(H_{u_i}\right) = C(\mathbf{x}, \mathbf{p}, \mathbf{u}_j(\mathbf{x}, \mathbf{p})) + D(\mathbf{x}, \mathbf{p}, \mathbf{u}_j(\mathbf{x}, \mathbf{p}))u_i = 0$$

with $D \neq 0$, so the condition does depend on u_i.

- Then can define the appropriate control law to stay on the singular arc as

$$u_i = -\frac{C(\mathbf{x}, \mathbf{p}, \mathbf{u}_j(\mathbf{x}, \mathbf{p},))}{D(\mathbf{x}, \mathbf{p}, \mathbf{u}_j(\mathbf{x}, \mathbf{p},))}$$

- Properties of this solution are:
  - $r \geq 2$ is even
  - Singular surface of dimension $2n - r$ in space of $(\mathbf{x}, \mathbf{p})$ in general, but $2n - r - 1$ if t_f is free (additional constraint that $H(t) = 0$)
  - Additional necessary condition for the singular arc to be extremal is that:
$$(-1)^{r/2} \frac{\partial}{\partial u_i} \left[\frac{d^r}{dt^r} H_u \right] \geq 0$$
  - Note that in the example above,
$$\frac{\partial}{\partial u_i} \left[\frac{d^r}{dt^r} H_u \right] \sim D$$

Last Example

- Goddard problem: thrust program for maximum altitude of a sounding rocket [Bryson and Ho, p. 253]. Given the EOM:

$$\begin{aligned} \dot{v} &= \frac{1}{m}[F(t) - D(v,h)] - g \\ \dot{h} &= v \\ \dot{m} &= \frac{-F(t)}{c} \end{aligned}$$

  where g is a constant, and drag model is $D(v,h) = \frac{1}{2}\rho v^2 C_d S e^{-\beta h}$

- **Problem**: Find $0 \le F(t) \le F_{\max}$ to maximize $h(t_f)$ with $v(0) = h(0) = 0$ and $m(0), m(t_f)$ are given

- The Hamiltonian is

$$H = p_1 \left(\frac{1}{m}[F(t) - D(v,h)] - g \right) + p_2 v - p_3 \frac{F(t)}{c}$$

  and since $v(t_f)$ is not specified and we are maximizing $h(t_f)$,

$$p_2(t_f) = -1 \qquad p_1(t_f) = 0$$

  – Note that $H(t) = 0$ since the final time is not specified.

- The costate EOM are:

$$\dot{\mathbf{p}} = \begin{bmatrix} \frac{1}{m}\frac{\partial D}{\partial v} & -1 & 0 \\ \frac{1}{m}\frac{\partial D}{\partial h} & 0 & 0 \\ \frac{F-D}{m^2} & 0 & 0 \end{bmatrix} \mathbf{p}$$

- H is linear in the controls, and the minimum is found by minimizing $(\frac{p_1}{m} - \frac{p_3}{c})F(t)$, which clearly has 3 possible solutions:

$$\begin{array}{cc} F = F_{\max} & (\frac{p_1}{m} - \frac{p_3}{c}) < 0 \\ 0 < F < F_{\max} \text{ if} & (\frac{p_1}{m} - \frac{p_3}{c}) = 0 \\ F = 0 & (\frac{p_1}{m} - \frac{p_3}{c}) > 0 \end{array}$$

  – Middle expression corresponds to a singular arc.

- Note: on a singular arc, must have $H_u = p_1 c - p_3 m = 0$ for finite interval, so then $\dot{H}_u = 0$ and $\ddot{H}_u = 0$, which means

$$\left(\frac{\partial D}{\partial v} + \frac{D}{c}\right) p_1 - m p_2 = 0$$

and

$$F = D + mg + \frac{m}{D + 2c\frac{\partial D}{\partial v} + c^2\frac{\partial^2 D}{\partial v^2}} \cdot \left[-g(D + c\frac{\partial D}{\partial v}) + c(c - v)\frac{\partial D}{\partial h} - vc^2\frac{\partial^2 D}{\partial v \partial h}\right] \tag{10.16}$$

which is a nonlinear feedback control law for thrust on a singular arc.

– For this particular drag model, the feedback law simplifies to:

$$F = D + mg + \frac{mg}{1 + 4(c/v) + 2(c/v)^2}\left[\frac{\beta c^2}{g}\left(1 + \frac{v}{c}\right) - 1 - 2\frac{c}{v}\right]$$

and the singular surface is: $mg = \left(1 + \frac{v}{c}\right) D$

- Constraints $H(t) = 0$, $H_u = 0$, and $\dot{H}_u = 0$ provide a condition that defines a surface for the singular arc in v, h, m space:

$$D + mg - \frac{v}{c}D - v\frac{\partial D}{\partial v} = 0 \tag{10.17}$$

- It can then be shown that the solution typically consists of 3 arcs:
  1. $F = F_{\max}$ until 10.17 is satisfied.
  2. Follow singular arc using 10.16 feedback law until $m(t) = m(t_f)$.
  3. $F = 0$ until $v = 0$.

  which is of the form "bang-singular-bang"

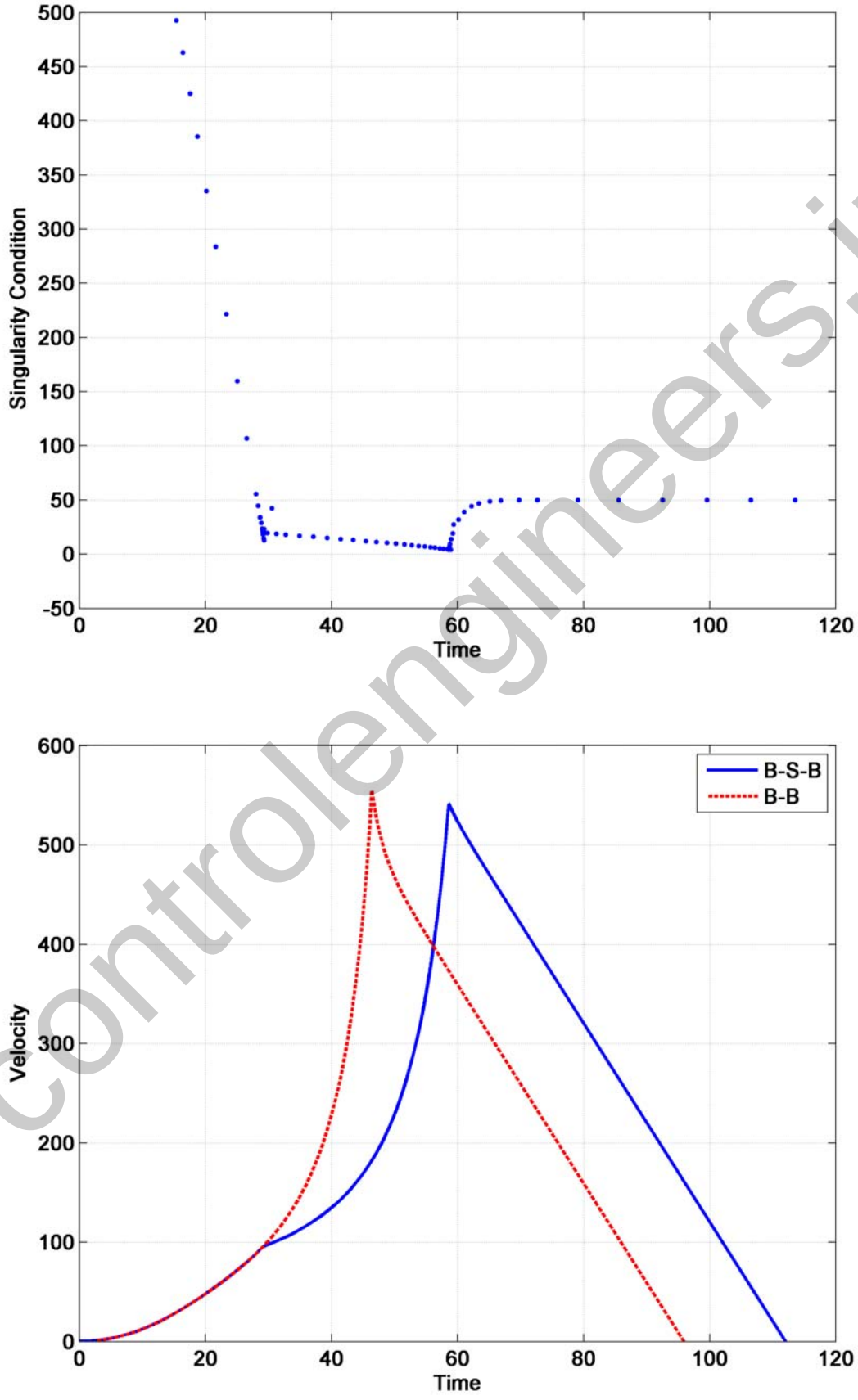


Figure 10.1: Goddard Problem

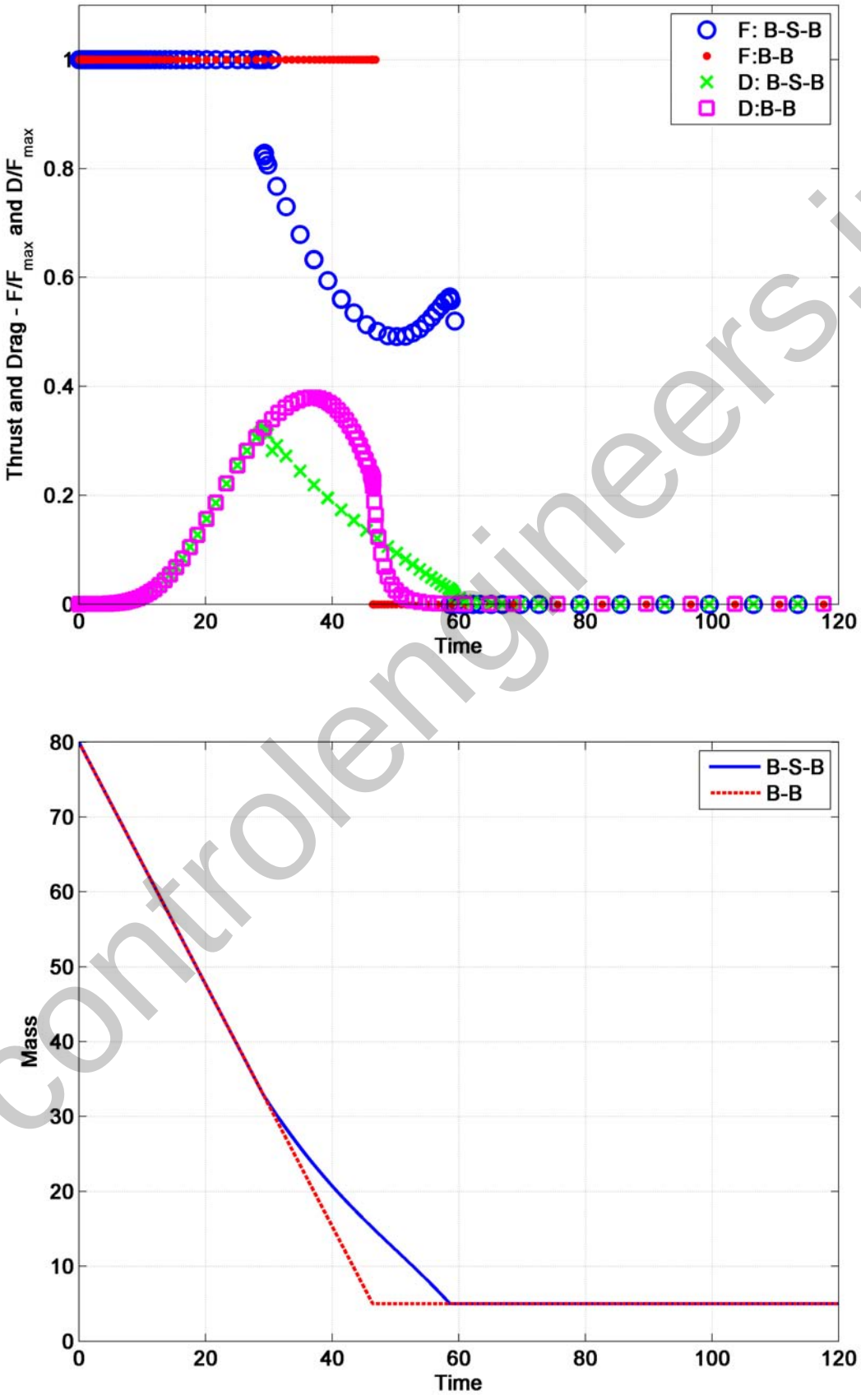


Figure 10.2: Goddard Problem

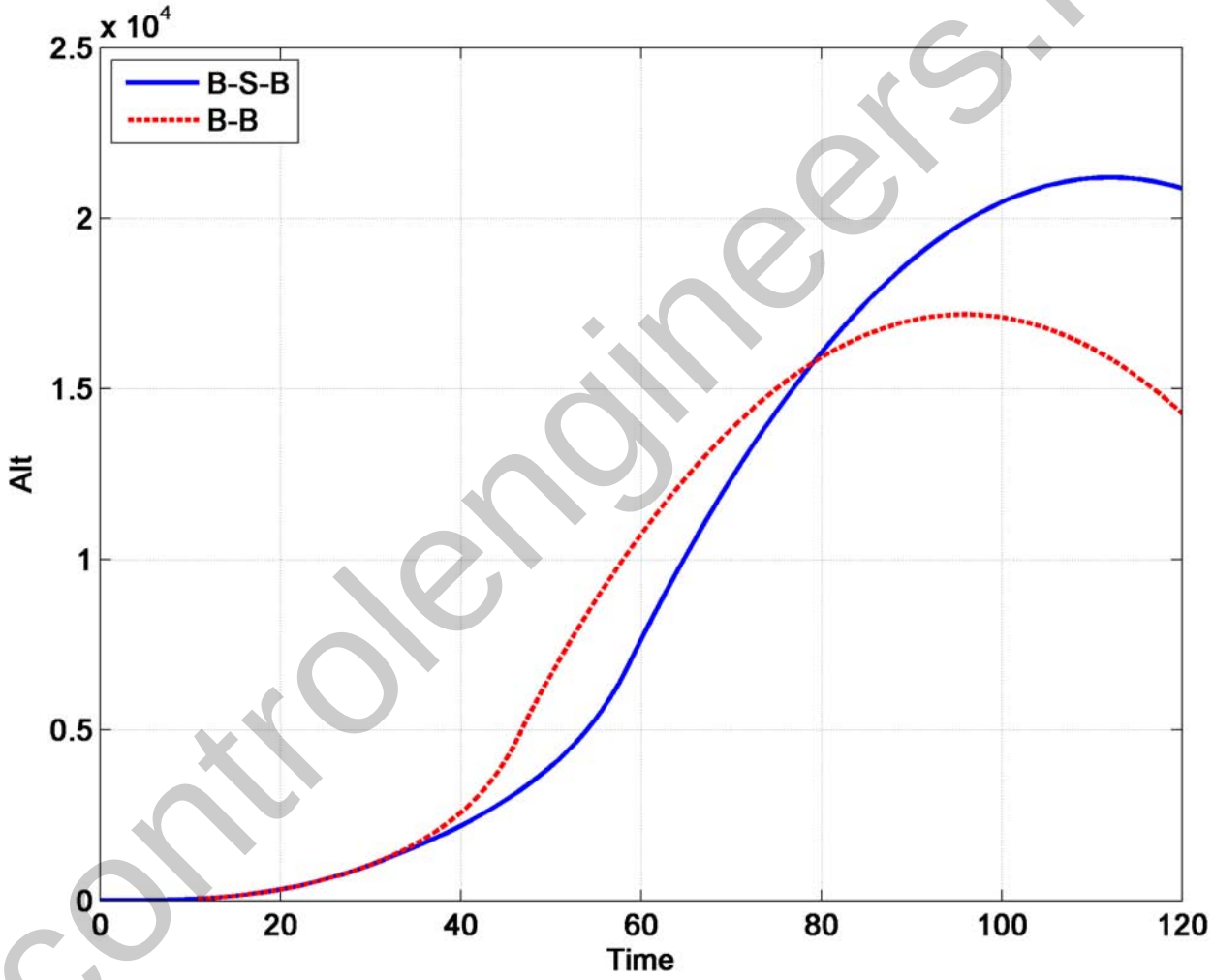


Figure 10.3: Goddard Problem

16.323 Lecture 11

Estimators/Observers

- Bryson Chapter 12
- Gelb – Optimal Estimation

Estimators/Observers

- **Problem:** So far we have assumed that we have full access to the state $\mathbf{x}(t)$ when we designed our controllers.
  - Most often all of this information is not available.
  - And certainly there is usually error in our knowledge of $\mathbf{x}$.

- Usually can only feedback information that is developed from the sensors measurements.
  - Could try "output feedback" $\mathbf{u} = K\mathbf{x} \quad \Rightarrow \quad \mathbf{u} = \hat{K}\mathbf{y}$
  - But this is type of controller is hard to design.

- **Alternative approach:** Develop a replica of the dynamic system that provides an "estimate" of the system states based on the measured output of the system.

- **New plan:** called a "separation principle"
  1. Develop estimate of $\mathbf{x}(t)$, called $\hat{\mathbf{x}}(t)$.
  2. Then switch from $\mathbf{u} = -K\mathbf{x}(t)$ to $\mathbf{u} = -K\hat{\mathbf{x}}(t)$.

- Two key questions:
  - How do we find $\hat{\mathbf{x}}(t)$?
  - Will this new plan work? (yes, and very well)

Estimation Schemes

- Assume that the system model is of the form:

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \ , \quad \mathbf{x}(0) \text{ unknown} \\ \mathbf{y} &= C_y\mathbf{x} \end{aligned}$$

  where
  - A, B, and C_y are known – possibly time-varying, but that is suppressed here.
  - $\mathbf{u}(t)$ is known
  - Measurable outputs are $\mathbf{y}(t)$ from $C_y \neq I$

- **Goal:** Develop a dynamic system whose state

$$\hat{\mathbf{x}}(t) = \mathbf{x}(t) \quad \forall t \geq 0$$

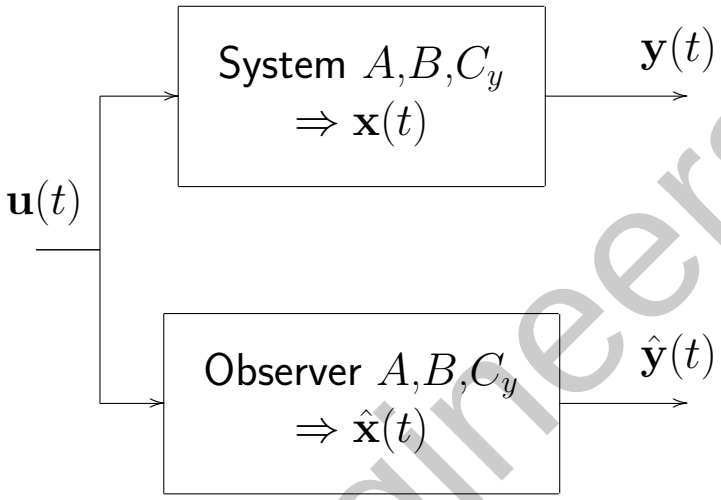
  Two primary approaches:
  - Open-loop.
  - Closed-loop.

Open-loop Estimator

- Given that we know the plant matrices and the inputs, we can just perform a simulation that runs in parallel with the system

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}} + B\mathbf{u}(t)$$

  – Then $\hat{\mathbf{x}}(t) \equiv \mathbf{x}(t)\ \forall\ t$ provided that $\hat{\mathbf{x}}(0) = \mathbf{x}(0)$



- To analyze this case, start with:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t)\\ \dot{\hat{\mathbf{x}}}(t) &= A\hat{\mathbf{x}}(t) + B\mathbf{u}(t)\end{aligned}$$

- Define the **estimation error**: $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$.
  – Now want $\tilde{\mathbf{x}}(t) = 0\ \forall\ t$, but is this realistic?

- **Major Problem:** We do not know $\mathbf{x}(0)$

- Subtract to get:

$$\frac{d}{dt}(\mathbf{x} - \hat{\mathbf{x}}) = A(\mathbf{x} - \hat{\mathbf{x}}) \quad \Rightarrow \quad \dot{\tilde{\mathbf{x}}}(t) = A\tilde{\mathbf{x}}$$

  which has the solution

$$\tilde{\mathbf{x}}(t) = e^{At}\tilde{\mathbf{x}}(0)$$

  – Gives the estimation error in terms of the initial error.

- Does this guarantee that $\tilde{\mathbf{x}} = 0 \ \forall \ t$?
  Or even that $\tilde{\mathbf{x}} \to 0$ as $t \to \infty$? (which is a more realistic goal).

  – Response is fine if $\tilde{\mathbf{x}}(0) = 0$. But what if $\tilde{\mathbf{x}}(0) \neq 0$?

- If A stable, then $\tilde{\mathbf{x}} \to 0$ as $t \to \infty$, but the dynamics of the estimation error are completely determined by the open-loop dynamics of the system (eigenvalues of A).
  – Could be very slow.
  – No obvious way to modify the estimation error dynamics.

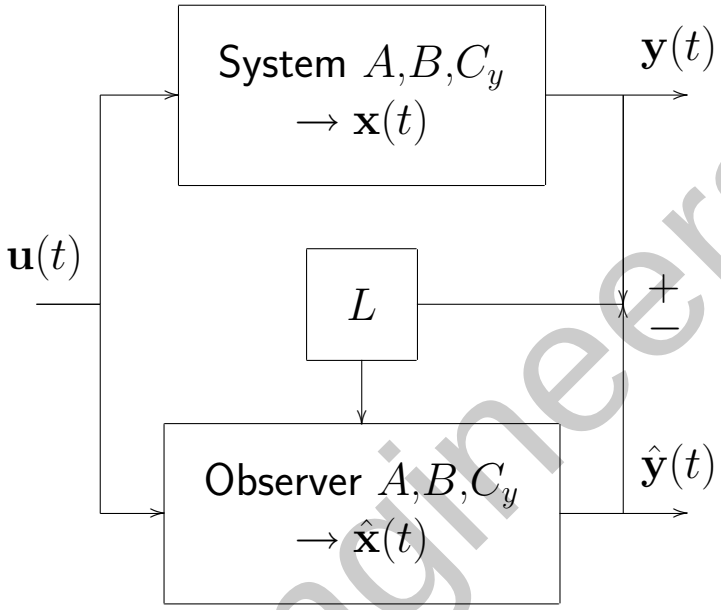
- Open-loop estimation **is not a very good idea.**

Closed-loop Estimator

- Obvious fix to problem: use the additional information available:
  – How well does the estimated output match the measured output?

$$\text{Compare: } \mathbf{y} = C_y\mathbf{x} \text{ with } \hat{\mathbf{y}} = C_y\hat{\mathbf{x}}$$

  – Then form $\tilde{\mathbf{y}} = \mathbf{y} - \hat{\mathbf{y}} \equiv C_y\tilde{\mathbf{x}}$



- **Approach:** Feedback $\tilde{\mathbf{y}}$ to improve our estimate of the state. Basic form of the estimator is:

$$\begin{aligned}\dot{\hat{\mathbf{x}}}(t) &= A\hat{\mathbf{x}}(t) + B\mathbf{u}(t) + \boxed{L\tilde{\mathbf{y}}(t)} \\ \hat{\mathbf{y}}(t) &= C_y\hat{\mathbf{x}}(t)\end{aligned}$$

  where L is a **user selectable gain matrix.**

- **Analysis:**

$$\begin{aligned}\dot{\tilde{\mathbf{x}}} = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}} &= [A\mathbf{x} + B\mathbf{u}] - [A\hat{\mathbf{x}} + B\mathbf{u} + L(\mathbf{y} - \hat{\mathbf{y}})] \\ &= A(\mathbf{x} - \hat{\mathbf{x}}) - L(C\mathbf{x} - C_y\hat{\mathbf{x}}) \\ &= A\tilde{\mathbf{x}} - LC_y\tilde{\mathbf{x}} = (A - LC_y)\tilde{\mathbf{x}}\end{aligned}$$

- So the closed-loop estimation error dynamics are now

$$\dot{\tilde{\mathbf{x}}} = (A - LC_y)\tilde{\mathbf{x}} \quad \text{with solution} \quad \tilde{\mathbf{x}}(t) = e^{(A-LC_y)t}\,\tilde{\mathbf{x}}(0)$$

- **Bottom line:** Can select the gain L to attempt to improve the convergence of the estimation error (and/or speed it up).
  - But now must worry about observability of the system $[A, C_y]$.

- Note the similarity:
  - **Regulator Problem:** pick K for $A - BK$
    - ◇ Choose $K \in \mathcal{R}^{1\times n}$ (SISO) such that the closed-loop poles

$$\det(sI - A + BK) = \Phi_c(s)$$

      are in the desired locations.
  - **Estimator Problem:** pick L for $A - LC_y$
    - ◇ Choose $L \in \mathcal{R}^{n\times 1}$ (SISO) such that the closed-loop poles

$$\det(sI - A + LC_y) = \Phi_o(s)$$

      are in the desired locations.

- These problems are obviously very similar – in fact they are called **dual problems**
  - Note: poles of $(A - LC_y)$ and $(A - LC_y)^T$ are identical.
  - Also have that $(A - LC_y)^T = A^T - C_y^T L^T$
  - So designing L^T for this transposed system looks like a standard regulator problem $(A - BK)$ where

$$\begin{aligned} A &\Rightarrow A^T \\ B &\Rightarrow C_y^T \\ K &\Rightarrow L^T \end{aligned}$$

Estimator Example 10–1

- Simple system (see page 11-23)

$$A = \begin{bmatrix} -1 & 1.5 \\ 1 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad x(0) = \begin{bmatrix} -0.5 \\ -1 \end{bmatrix}$$
$$C_y = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = 0$$

  - Assume that the initial conditions are not well known.
  - System stable, but $\lambda_{\max}(A) = -0.18$
  - Test observability:

$$\texttt{rank} \begin{bmatrix} C_y \\ C_y A \end{bmatrix} = \texttt{rank} \begin{bmatrix} 1 & 0 \\ -1 & 1.5 \end{bmatrix}$$

- Use open and closed-loop estimators. Since the initial conditions are not well known, use $\hat{\mathbf{x}}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

- Open-loop estimator:

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= A\hat{\mathbf{x}} + B\mathbf{u} \\ \hat{\mathbf{y}} &= C_y\hat{\mathbf{x}} \end{aligned}$$

- Closed-loop estimator:

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= A\hat{\mathbf{x}} + B\mathbf{u} + L\tilde{\mathbf{y}} = A\hat{\mathbf{x}} + B\mathbf{u} + L(\mathbf{y} - \hat{\mathbf{y}}) \\ &= (A - LC_y)\hat{\mathbf{x}} + B\mathbf{u} + L\mathbf{y} \\ \hat{\mathbf{y}} &= C_y\hat{\mathbf{x}} \end{aligned}$$

  - Dynamic system with poles $\lambda_i(A - LC_y)$ that takes the measured plant outputs as an input and generates an estimate of $\mathbf{x}$.
  - Use `place` command to set closed-loop pole locations

- Typically simulate both systems together for simplicity

- Open-loop case:

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u}\\ \mathbf{y} &= C_y\mathbf{x}\\ \dot{\hat{\mathbf{x}}} &= A\hat{\mathbf{x}} + B\mathbf{u}\\ \hat{\mathbf{y}} &= C_y\hat{\mathbf{x}}\end{aligned}$$

$$\Rightarrow \begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\hat{\mathbf{x}}}\end{bmatrix} = \begin{bmatrix}A & 0\\ 0 & A\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{x}}\end{bmatrix} + \begin{bmatrix}B\\ B\end{bmatrix}\mathbf{u}\ , \quad \begin{bmatrix}\mathbf{x}(0)\\ \hat{\mathbf{x}}(0)\end{bmatrix} = \begin{bmatrix}-0.5\\ -1\\ 0\\ 0\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{y}\\ \hat{\mathbf{y}}\end{bmatrix} = \begin{bmatrix}C_y & 0\\ 0 & C_y\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{x}}\end{bmatrix}$$

- Closed-loop case:

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u}\\ \dot{\hat{\mathbf{x}}} &= (A - LC_y)\hat{\mathbf{x}} + B\mathbf{u} + LC_y\mathbf{x}\end{aligned}$$

$$\Rightarrow \begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\hat{\mathbf{x}}}\end{bmatrix} = \begin{bmatrix}A & 0\\ LC_y & A - LC_y\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{x}}\end{bmatrix} + \begin{bmatrix}B\\ B\end{bmatrix}\mathbf{u}$$

- Example uses a strong $\mathbf{u}(t)$ to shake things up

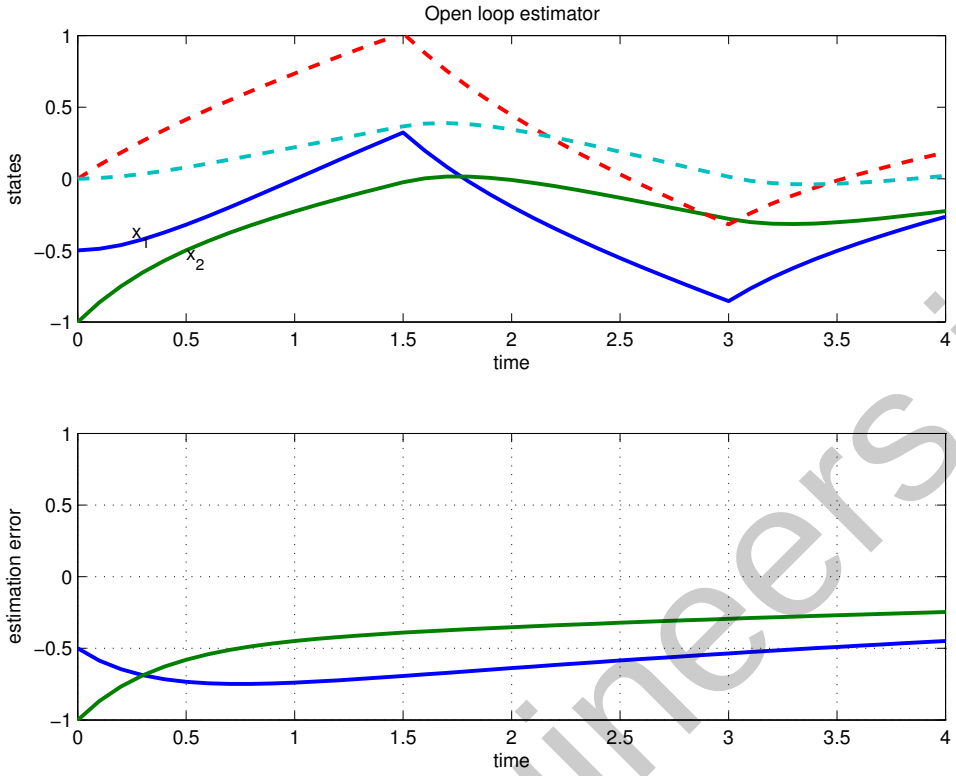


Figure 11.1: Open-loop estimator. Estimation error converges to zero, but very slowly.

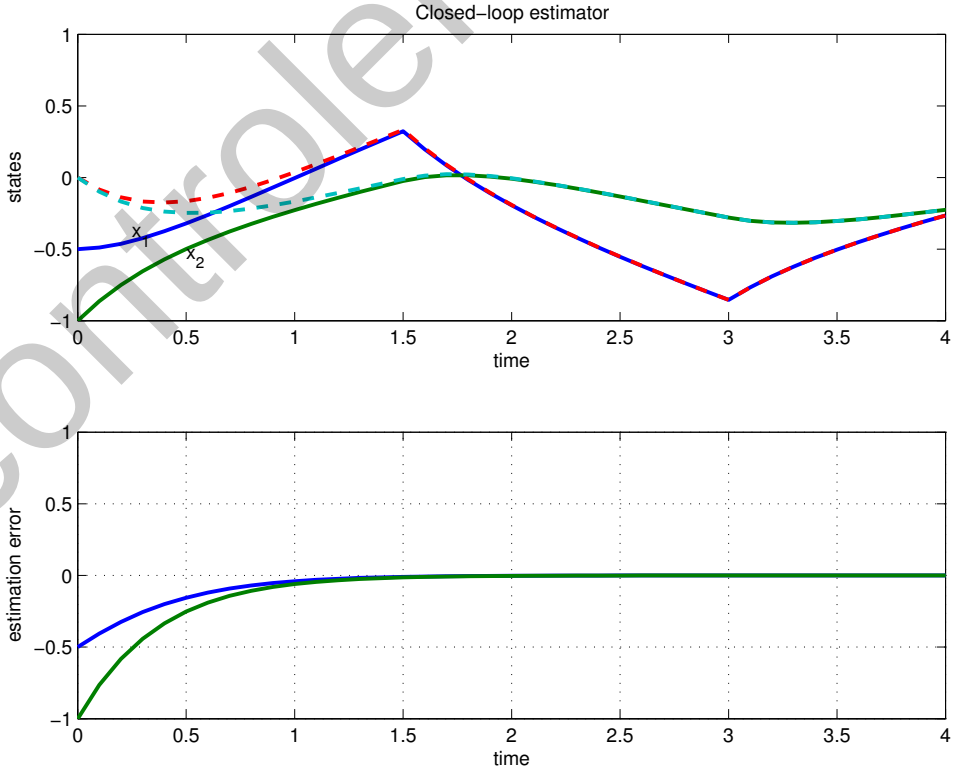


Figure 11.2: Closed-loop estimator. Convergence looks much better.

Estimator Poles?

- Location heuristics for poles still apply – use Bessel, ITAE, ...
  - Main difference: probably want to make the estimator faster than you intend to make the regulator – should enhance the control, which is based on $\hat{\mathbf{x}}(t)$.
  - ROT: Factor of 2–3 in the time constant $\zeta\omega_n$ associated with the regulator poles.

- **Note:** When designing a regulator, were concerned with "bandwidth" of the control getting too high $\Rightarrow$ often results in control commands that *saturate* the actuators and/or change rapidly.

- Different concerns for the estimator:
  - Loop closed inside computer, so saturation not a problem.
  - However, the measurements $\mathbf{y}$ are often "noisy", and we need to be careful how we use them to develop our state estimates.

$\Rightarrow$ **High bandwidth estimators** tend to accentuate the effect of sensing noise in the estimate.
  - State estimates tend to "track" the measurements, which are fluctuating randomly due to the noise.

$\Rightarrow$ **Low bandwidth estimators** have lower gains and tend to rely more heavily on the plant model
  - Essentially an open-loop estimator – tends to ignore the measurements and just uses the plant model.

Optimal Estimator

- Can also develop an **optimal estimator** for this type of system.
  - Given duality of regulator and estimator, would expect to see close connection between optimal estimator and regulator (LQR)

- Key step is to **balance** the effect of the various types of random noise in the system on the estimator:

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} + B_w\mathbf{w} \\ \mathbf{y} &= C_y\mathbf{x} + \mathbf{v}\end{aligned}$$

  - $\mathbf{w}$: "process noise" – models uncertainty in the system model.
  - $\mathbf{v}$: "sensor noise" – models uncertainty in the measurements.

- Typically assume that $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are zero mean $E[\mathbf{w}(t)] = 0$ and
  - Uncorrelated Gaussian white random noises: no correlation between the noise at one time instant and another

$$\begin{aligned}E[\mathbf{w}(t_1)\mathbf{w}(t_2)^T] &= R_{\mathrm{ww}}(t_1)\delta(t_1 - t_2) \quad \Rightarrow \mathbf{w}(t) \sim \mathcal{N}(0, R_{\mathrm{ww}}) \\ E[\mathbf{v}(t_1)\mathbf{v}(t_2)^T] &= R_{\mathrm{vv}}(t_1)\delta(t_1 - t_2) \quad \Rightarrow \mathbf{v}(t) \sim \mathcal{N}(0, R_{\mathrm{vv}}) \\ E[\mathbf{w}(t_1)\mathbf{v}(t_2)^T] &= 0\end{aligned}$$

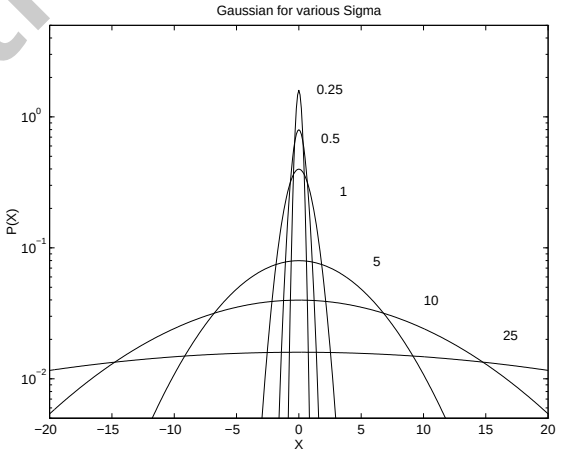


Figure 11.3: Example of impact of covariance $= \sigma^2$ on the distribution of the PDF – wide distribution corresponds to large uncertainty in the variable

Analysis

- With noise in the system, the model is of the form:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + B_w\mathbf{w} \ , \quad \mathbf{y} = C_y\mathbf{x} + \mathbf{v}$$

  – And the estimator is of the form:

$$\dot{\hat{\mathbf{x}}} = A\hat{\mathbf{x}} + B\mathbf{u} + L(\mathbf{y} - \hat{\mathbf{y}}) \ , \quad \hat{\mathbf{y}} = C_y\hat{\mathbf{x}}$$

- **Analysis:** in this case:

$$\begin{aligned}
\dot{\tilde{\mathbf{x}}} &= \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}} = [A\mathbf{x} + B\mathbf{u} + B_w\mathbf{w}] - [A\hat{\mathbf{x}} + B\mathbf{u} + L(\mathbf{y} - \hat{\mathbf{y}})] \\
&= A(\mathbf{x} - \hat{\mathbf{x}}) - L(C_y\mathbf{x} - C_y\hat{\mathbf{x}}) + B_w\mathbf{w} - L\mathbf{v} \\
&= A\tilde{\mathbf{x}} - LC_y\tilde{\mathbf{x}} + B_w\mathbf{w} - L\mathbf{v} \\
&= (A - LC_y)\tilde{\mathbf{x}} + B_w\mathbf{w} - L\mathbf{v} \qquad (11.18)
\end{aligned}$$

- This equation of the estimation error explicitly shows the **conflict** in the estimator design process. Must **balance** between:
  – Speed of the estimator decay rate, which is governed by

$$\mathtt{Re}[\lambda_i(A - LC_y)]$$

  – Impact of the sensing noise $\mathbf{v}$ through the gain L

- Fast state reconstruction requires rapid decay rate – typically requires a large L, but that tends to magnify the effect of $\mathbf{v}$ on the estimation process.
  – The effect of the process noise is always there, but the choice of L will tend to mitigate/accentuate the effect of $\mathbf{v}$ on $\tilde{\mathbf{x}}(t)$.

- **Kalman Filter** needs to provide an optimal balance between the two conflicting problems for a given "size" of the process and sensing noises.

- Note that Eq. 11.18 is of the form of a linear time-varying system driven by white Gaussian noise
  - Can predict the **mean square value** of the state (estimation error in this case) $Q(t) = E[\tilde{\mathbf{x}}(t)\tilde{\mathbf{x}}(t)^T]$ over time using $Q(0) = Q_0$ and

$$\begin{aligned}\dot{Q}(t) &= [A - LC_y]\,Q(t) + Q(t)\,[A - LC_y]^T \\ &\quad + \begin{bmatrix} B_w & -L \end{bmatrix} \begin{bmatrix} R_{\text{ww}} & 0 \\ 0 & R_{\text{vv}} \end{bmatrix} \begin{bmatrix} B_w^T \\ -L^T \end{bmatrix} \\ &= [A - LC_y]\,Q(t) + Q(t)\,[A - LC_y]^T + B_w R_{\text{ww}} B_w^T + L R_{\text{vv}} L^T\end{aligned}$$

  - Called a matrix **differential Lyapunov Equation**[16]
- Note that ideally would like to minimize $Q(t)$ or $\texttt{trace}\ Q(t)$, but that is difficult to do & describe easily[17].
- Instead, consider option of trying to minimize $\texttt{trace}\ \dot{Q}(t)$, the argument being that then $\int_0^t \texttt{trace}\ \dot{Q}(\tau)d\tau$ is small.
  - Not quite right, but good enough to develop some insights
- To proceed note that

$$\frac{\partial}{\partial X}\texttt{trace}[AXB] = \frac{\partial}{\partial X}\texttt{trace}[B^T X^T A^T] = A^T B^T$$

  and

$$\frac{\partial}{\partial X}\texttt{trace}[AXBX^T C] = A^T C^T X B^T + CAXB$$

- So for minimum we require that

$$\frac{\partial}{\partial L}\texttt{trace}\ \dot{Q} = -2Q^T C_y^T + 2LR_{\text{vv}} = 0$$

  which implies that

$$L = Q(t)C_y^T R_{\text{vv}}^{-1}$$

[16] See K+S, chapter 1.11 for details.

[17] My 16.324 discuss how to pose the problem in discrete time and then let $\Delta t \to 0$ to recover the continuous time results.

- Note that if we use this expression for L in the original differential Lyapunov Equation, we obtain:

$$\begin{aligned}
\dot{Q}(t) &= [A - LC_y]\,Q(t) + Q(t)\,[A - LC_y]^T + B_w R_{\rm ww} B_w^T + L R_{\rm vv} L^T \\
&= \left[A - Q(t)C_y^T R_{\rm vv}^{-1} C_y\right] Q(t) + Q(t)\left[A - Q(t)C_y^T R_{\rm vv}^{-1} C_y\right]^T \\
&\quad + B_w R_{\rm ww} B_w^T + Q(t)C_y^T R_{\rm vv}^{-1} R_{\rm vv} (Q(t)C_y^T R_{\rm vv}^{-1})^T \\
&= AQ(t) + Q(t)A^T - 2Q(t)C_y^T R_{\rm vv}^{-1} C_y Q(t) + B_w R_{\rm ww} B_w^T \\
&\quad + Q(t)C_y^T R_{\rm vv}^{-1} C_y Q(t) \\
\dot{Q}(t) &= AQ(t) + Q(t)A^T + B_w R_{\rm ww} B_w^T - Q(t)C_y^T R_{\rm vv}^{-1} C_y Q(t)
\end{aligned}$$

  which is obviously a matrix differential Riccati equation.

Optimal Kalman Filter

- **Goal:** develop an estimator $\hat{\mathbf{x}}(t)$ which is a linear function of the measurements $\mathbf{y}(\tau)$ $(0 \leq \tau \leq t)$ and **minimizes** the function

$$\begin{aligned} J &= \mathtt{trace}(Q(t)) \\ Q(t) &= E\left[\{\mathbf{x}(t) - \hat{\mathbf{x}}(t)\}\{\mathbf{x}(t) - \hat{\mathbf{x}}(t)\}^T\right] \end{aligned}$$

  which is the **covariance** for the **estimation error**.

- **Solution:** is a closed-loop estimator [18]

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}} + L(t)(\mathbf{y}(t) - C_y\hat{\mathbf{x}}(t))$$

  where $L(t) = Q(t)C_y^T R_{\text{vv}}^{-1}$ and $Q(t) \geq 0$ solves

$$\dot{Q}(t) = AQ(t) + Q(t)A^T + B_w R_{\text{ww}} B_w^T - Q(t)C_y^T R_{\text{vv}}^{-1} C_y Q(t)$$

  - Note that $\hat{x}(0)$ and $Q(0)$ are known
  - Differential equation for $Q(t)$ **solved forward in time**.
  - **Filter form** of the **differential matrix Riccati** equation for the **error covariance**.
  - Note that the $AQ(t) + Q(t)A^T \ldots$ is different than with the regulator which had $P(t)A + A^T P(t) \ldots$

- Called **Kalman-Bucy Filter** – **linear quadratic estimator (LQE)**

[18] See OCW notes for 16.322 "Stochastic Estimation and Control" for the details of this derivation.

- Note that an increase in $Q(t)$ corresponds to **increased uncertainty in the state estimate**. $\dot{Q}(t)$ has several contributions:
  - $AQ(t) + Q(t)A^T$ is the homogeneous part
  - $B_w R_{\mathrm{ww}} B_w^T$ increase due to the process measurements
  - $Q(t)C_y^T R_{\mathrm{vv}}^{-1} C_y Q(t)$ decrease due to measurements

- The estimator gain is $L(t) = Q(t)C_y^T R_{\mathrm{vv}}^{-1}$
  - Feedback on the **innovation**, $\mathbf{y} - \hat{\mathbf{y}}$
  - If the uncertainty about the state is high, then $Q(t)$ is large, and so the innovation $\mathbf{y} - C_y\hat{\mathbf{x}}$ is weighted heavily ($L \uparrow$)
  - If the measurements are very accurate $R_{\mathrm{vv}} \downarrow$, then the measurements are heavily weighted

Steady State

- Assume that [19]
  1. $R_{\rm vv} > 0$, $R_{\rm ww} > 0$
  2. All plant dynamics are constant in time
  3. $[A, C_y]$ detectable
  4. $[A, B_w]$ stabilizable

- Then, as with the LQR problem, the covariance of the LQE quickly settles down to a constant Q_{ss} independent of $Q(0)$, as $t \to \infty$ where

$$AQ_{ss} + Q_{ss}A^T + B_w R_{\rm ww} B_w^T - Q_{ss} C_y^T R_{\rm vv}^{-1} C_y Q_{ss} = 0$$

  – Stabilizable/detectable gives a unique $Q_{ss} \geq 0$
  – $Q_{ss} > 0$ iff $[A, B_w]$ controllable
  – $L_{ss} = Q_{ss} C_y^T R_{\rm vv}^{-1}$

- If Q_{ss} exists, the steady state filter

$$\begin{aligned} \dot{\hat{\mathbf{x}}}(t) &= A\hat{\mathbf{x}} + L_{ss}(\mathbf{y}(t) - C_y\hat{\mathbf{x}}(t)) \\ &= (A - L_{ss}C_y)\hat{\mathbf{x}}(t) + L_{ss}\mathbf{y}(t) \end{aligned}$$

  is asymptotically stable iff (1)–(4) above hold.

[19] Compare this with 4–10

Filter Interpretation

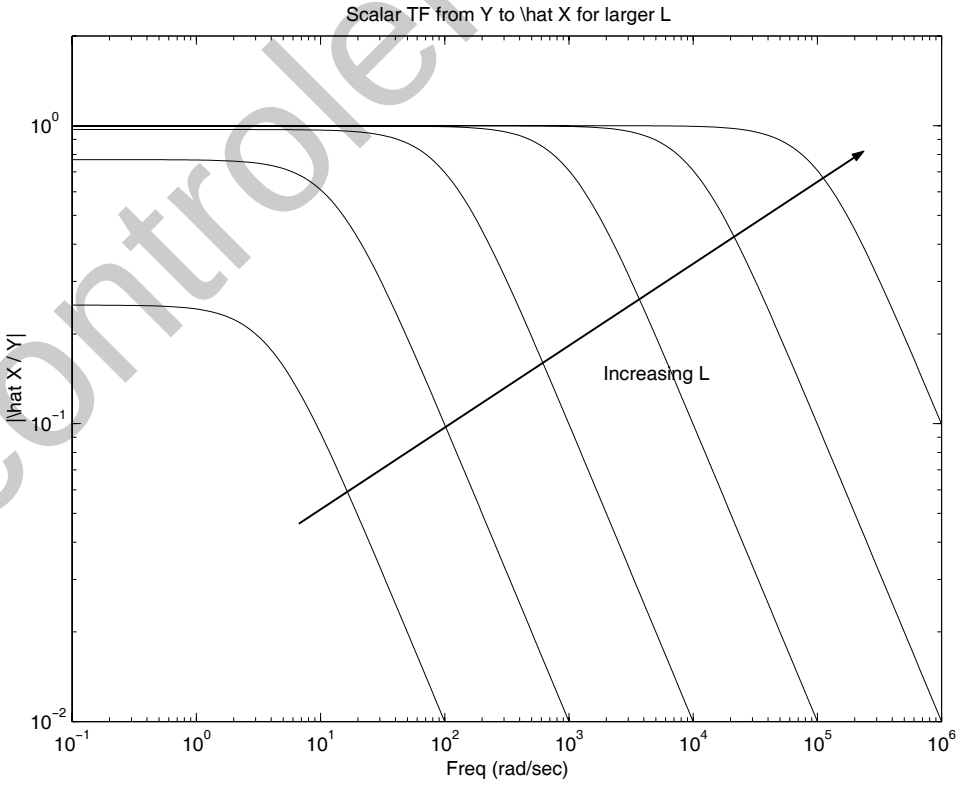
- Given that $\dot{\hat{\mathbf{x}}} = (A - LC_y)\hat{\mathbf{x}} + L\mathbf{y}$

- Consider a scalar system, and take the Laplace transform of both sides to get:

$$\frac{\hat{X}(s)}{Y(s)} = \frac{L}{sI - (A - LC_y)}$$

- This is the transfer function from the "measurement" to the "estimated state"
  - It looks like a low-pass filter.

- Clearly, by lowering R_{vv}, and thus increasing L, we are pushing out the pole.
  - DC gain asymptotes to $1/C_y$ as $L \to \infty$



Example 10–2

- Lightly Damped Harmonic Oscillator

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w$$

  and $y = x_1 + v$, where $R_{\rm ww} = 1$ and $R_{\rm vv} = r$.

  – Can sense the position state of the oscillator, but want to develop an estimator to reconstruct the velocity state.

- **Symmetric root locus** exists for the optimal estimator. Can find location of the optimal poles using a SRL based on the TF

$$G_{yw}(s) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s & -1 \\ \omega_0^2 & s \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{s^2 + \omega_0^2} = \frac{N(s)}{D(s)}$$

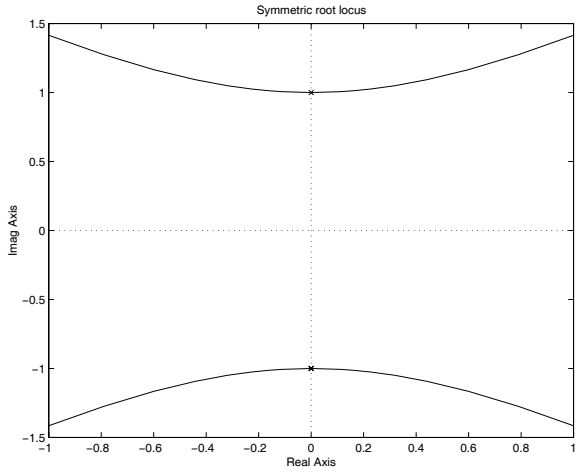
  – SRL for the closed-loop poles $\lambda_i(A - LC)$ of the estimator which are the LHP roots of:

$$D(s)D(-s) \pm \frac{R_{\rm ww}}{R_{\rm vv}} N(s)N(-s) = 0$$

  – Pick sign to ensure that there are no poles on the $\mathbf{j}\omega$-axis (other than for a gain of zero)

  – So we must find the LHP roots of

$$\left[s^2 + \omega_0^2\right] \left[(-s)^2 + \omega_0^2\right] + \frac{1}{r} = (s^2 + \omega_0^2)^2 + \frac{1}{r} = 0$$



- Note that as $r \to 0$ (clean sensor), the estimator poles tend to ∞ along the ± 45 deg asymptotes, so the poles are approximately

$$s \approx \frac{-1 \pm j}{\sqrt{r}} \quad \Rightarrow \quad \Phi_e(s) = s^2 + \frac{2}{\sqrt{r}}s + \frac{2}{r} = 0$$

- Can use these estimate pole locations in `acker`, to get that

$$\begin{aligned} L &= \left(\begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix}^2 + \frac{2}{\sqrt{r}} \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} + \frac{2}{r} I \right) \begin{bmatrix} C \\ CA \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{2}{r} - \omega_0^2 & \frac{2}{\sqrt{r}} \\ -\frac{2}{\sqrt{r}}\omega_0^2 & \frac{2}{r} - \omega_0^2 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 1 \end{bmatrix} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{r}} \\ \frac{2}{r} - \omega_0^2 \end{bmatrix} \end{aligned}$$

- Given L, A, and C, we can develop the estimator transfer function from the measurement y to the $\hat{x}_2$

$$\begin{aligned} \frac{\hat{x}_2}{y} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \left(sI - \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} + \begin{bmatrix} \frac{2}{\sqrt{r}} \\ \frac{2}{r} - \omega_0^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} \frac{2}{\sqrt{r}} \\ \frac{2}{r} - \omega_0^2 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s + \frac{2}{\sqrt{r}} & -1 \\ \frac{2}{r} & s \end{bmatrix}^{-1} \begin{bmatrix} \frac{2}{\sqrt{r}} \\ \frac{2}{r} - \omega_0^2 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s & 1 \\ \frac{-2}{r} & s + \frac{2}{\sqrt{r}} \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{r}} \\ \frac{2}{r} - \omega_0^2 \end{bmatrix} \frac{1}{s^2 + \frac{2}{\sqrt{r}}s + \frac{2}{r}} \\ &= \frac{\frac{-2}{r}\frac{2}{\sqrt{r}} + (s + \frac{2}{\sqrt{r}})(\frac{2}{r} - \omega_0^2)}{s^2 + \frac{2}{\sqrt{r}}s + \frac{2}{r}} \approx \frac{s - \sqrt{r}\omega_0^2}{s^2 + \frac{2}{\sqrt{r}}s + \frac{2}{r}} \end{aligned}$$

- Filter zero asymptotes to $s = 0$ as $r \to 0$ and the two poles $\to \infty$
- Resulting estimator looks like a "band-limited" differentiator.
  - Expected because we measure position and want to estimate velocity.
  - Frequency band over which filter performs differentiation determined by the "relative cleanliness" of the measurements.

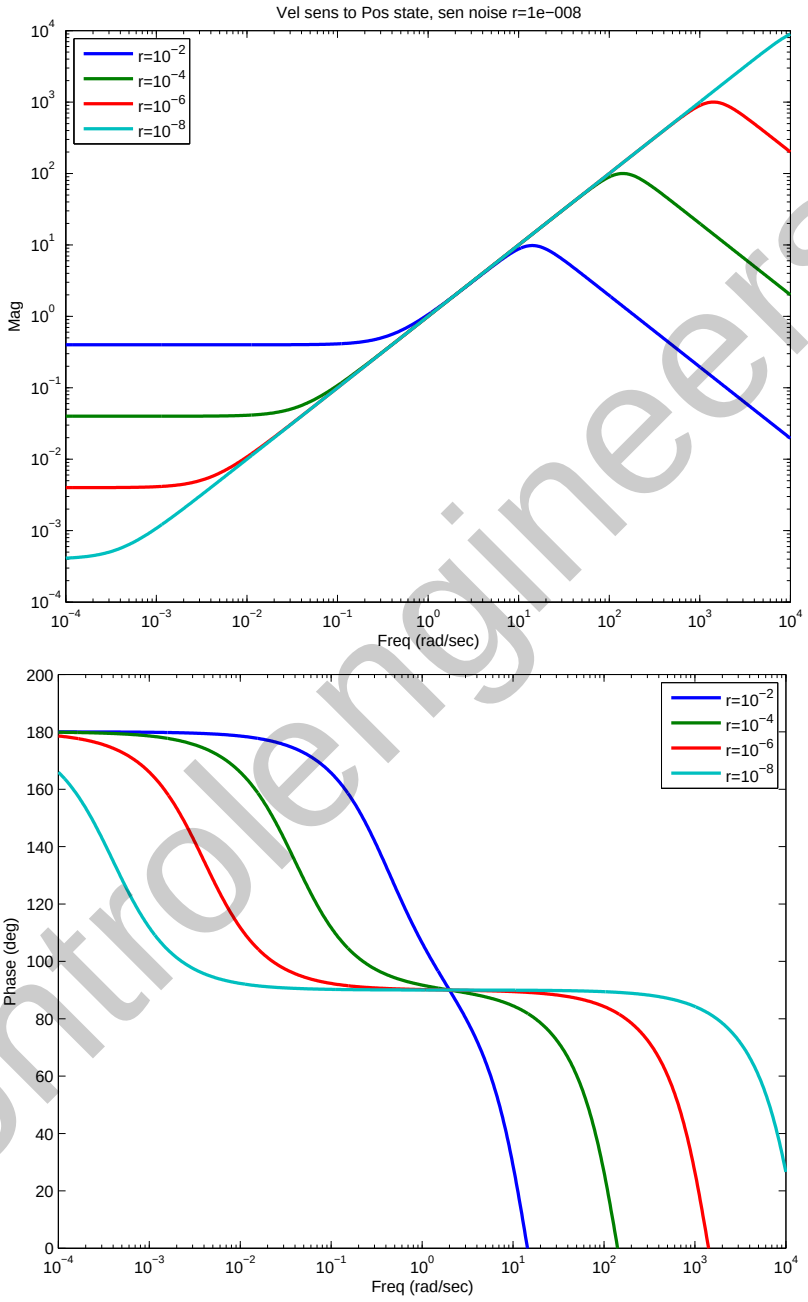


Figure 11.4: Bandlimited differentiation of the position measurement from LQE: $r = 10^{-2}$, $r = 10^{-4}$, $r = 10^{-6}$, and $r = 10^{-8}$

Final Thoughts

- Note that the feedback gain L in the estimator only stabilizes the estimation error.
  - If the system is unstable, then the state estimates will also go to ∞, with zero error from the actual states.

- Estimation is an important concept of its own.
  - Not always just "part of the control system"
  - Critical issue for guidance and navigation system

- More complete discussion requires that we study stochastic processes and optimization theory.

- **Estimation is all about which do you trust more: your measurements or your model.**

- Strong duality between LQR and LQE problems

$$\begin{array}{ccc} A & \leftrightarrow & A^T \\ B & \leftrightarrow & C_y^T \\ C_z & \leftrightarrow & B_w^T \\ R_{\mathrm{zz}} & \leftrightarrow & R_{\mathrm{ww}} \\ R_{\mathrm{uu}} & \leftrightarrow & R_{\mathrm{vv}} \\ K(t) & \leftrightarrow & L^T(t_f - t) \\ P(t) & \leftrightarrow & Q(t_f - t) \end{array}$$

Basic Estimator (examp1.m) (See page 11-7)

```
% Examples of estimator performance
% Jonathan How, MIT
% 16.333 Fall 2005
%
% plant dynamics
%
a=[-1 1.5;1 -2];b=[1 0]';c=[1 0];d=0;
%
% estimator gain calc
%
l=place(a',c',[-3 -4]);l=l'
%
% plant initial cond
xo=[-.5;-1];
% extimator initial cond
xe=[0 0]';
t=[0:.1:10];
%
% inputs
%
u=0;u=[ones(15,1);-ones(15,1);ones(15,1)/2;-ones(15,1)/2;zeros(41,1)];
%
% open-loop extimator
%
A_ol=[a zeros(size(a));zeros(size(a)) a];
B_ol=[b;b];
C_ol=[c zeros(size(c));zeros(size(c)) c];
D_ol=zeros(2,1);
%
% closed-loop extimator
%
A_cl=[a zeros(size(a));l*c a-l*c];B_cl=[b;b];
C_cl=[c zeros(size(c));zeros(size(c)) c];D_cl=zeros(2,1);

[y_cl,x_cl]=lsim(A_cl,B_cl,C_cl,D_cl,u,t,[xo;xe]);
[y_ol,x_ol]=lsim(A_ol,B_ol,C_ol,D_ol,u,t,[xo;xe]);

figure(1);clf;subplot(211)
plot(t,x_cl(:,[1 2]),t,x_cl(:,[3 4]),'--','LineWidth',2);axis([0 4 -1 1]);
title('Closed-loop estimator');ylabel('states');xlabel('time')
text(.25,-.4,'x_1');text(.5,-.55,'x_2');subplot(212)
plot(t,x_cl(:,[1 2])-x_cl(:,[3 4]),'LineWidth',2)
%setlines;
axis([0 4 -1 1]);grid on
ylabel('estimation error');xlabel('time')

figure(2);clf;subplot(211)
plot(t,x_ol(:,[1 2]),t,x_ol(:,[3 4]),'--','LineWidth',2);axis([0 4 -1 1])
title('Open loop estimator');ylabel('states');xlabel('time')
text(.25,-.4,'x_1');text(.5,-.55,'x_2');subplot(212)
plot(t,x_ol(:,[1 2])-x_ol(:,[3 4]),'LineWidth',2)
%setlines;
axis([0 4 -1 1]);grid on
ylabel('estimation error');xlabel('time')

print -depsc -f1 est11.eps; jpdf('est11')
print -depsc -f2 est12.eps; jpdf('est12')
```

Filter Interpretation

```
% Simple LQE example showing SRL
% 16.323 Spring 2007
% Jonathan How
%
a=[0 1;-4 0];
c=[1 0]; % pos sensor
c2=[0 1]; % vel state out
f=logspace(-4,4,800);

r=1e-2;
l=polyvalm([1 2/sqrt(r) 2/r],a)*inv([c;c*a])*[0 1]'
[nn,dd]=ss2tf(a-l*c,l,c2,0); % to the vel estimate
g=freqresp(nn,dd,f*j);
[r roots(nn)]
figure(1)
subplot(211)
f1=f;g1=g;
loglog(f,abs(g))
%hold on;fill([5e2 5e2 1e3 1e3 5e2]',[1e4 1e-4 1e-4 1e4 1e4]','c');hold off
xlabel('Freq (rad/sec)')
ylabel('Mag')
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
axis([1e-3 1e3 1e-4 1e4])
subplot(212)
semilogx(f,unwrap(angle(g))*180/pi)
xlabel('Freq (rad/sec)')
ylabel('Phase (deg)')
axis([1e-3 1e3 0 200])

figure(2)
r=1e-4;
l=polyvalm([1 2/sqrt(r) 2/r],a)*inv([c;c*a])*[0 1]'
[nn,dd]=ss2tf(a-l*c,l,c2,0); % to the vel estimate
g=freqresp(nn,dd,f*j);
[r roots(nn)]
subplot(211)
f2=f;g2=g;
loglog(f,abs(g))
%hold on;fill([5e2 5e2 1e3 1e3 5e2]',[1e4 1e-4 1e-4 1e4 1e4]','c');hold off
xlabel('Freq (rad/sec)')
ylabel('Mag')
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
axis([1e-3 1e3 1e-4 1e4])
subplot(212)
semilogx(f,unwrap(angle(g))*180/pi)
xlabel('Freq (rad/sec)')
ylabel('Phase (deg)')
%bode(nn,dd);
axis([1e-3 1e3 0 200])

figure(3)
r=1e-6;
l=polyvalm([1 2/sqrt(r) 2/r],a)*inv([c;c*a])*[0 1]'
[nn,dd]=ss2tf(a-l*c,l,c2,0); % to the vel estimate
g=freqresp(nn,dd,f*j);
[r roots(nn)]
subplot(211)
f3=f;g3=g;
loglog(f,abs(g))
%hold on;fill([5e2 5e2 1e3 1e3 5e2]',[1e4 1e-4 1e-4 1e4 1e4]','c');hold off
xlabel('Freq (rad/sec)')
ylabel('Mag')
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
axis([1e-3 1e3 1e-4 1e4])
subplot(212)
semilogx(f,unwrap(angle(g))*180/pi)
xlabel('Freq (rad/sec)')
```

```
ylabel('Phase (deg)')
%bode(nn,dd);
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
axis([1e-3 1e3 0 200])

figure(4)
r=1e-8;
l=polyvalm([1 2/sqrt(r) 2/r],a)*inv([c;c*a])*[0 1]'
[nn,dd]=ss2tf(a-l*c,l,c2,0); % to the vel estimate
g=freqresp(nn,dd,f*j);
[r roots(nn)]
f4=f;g4=g;
subplot(211)
loglog(f,abs(g))
%hold on;fill([5e2 5e2 1e3 1e3 5e2]',[1e4 1e-4 1e-4 1e4 1e4]','c');hold off
xlabel('Freq (rad/sec)')
ylabel('Mag')
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
axis([1e-3 1e3 1e-4 1e4])
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
subplot(212)
semilogx(f,unwrap(angle(g))*180/pi)
xlabel('Freq (rad/sec)')
ylabel('Phase (deg)')
%bode(nn,dd);
axis([1e-3 1e3 0 200])

print -depsc -f1 filt1.eps; jpdf('filt1')
print -depsc -f2 filt2.eps;jpdf('filt2')
print -depsc -f3 filt3.eps;jpdf('filt3')
print -depsc -f4 filt4.eps;jpdf('filt4')

figure(5);clf
%subplot(211)
loglog(f1,abs(g1),f2,abs(g2),f3,abs(g3),f4,abs(g4),'Linewidth',2)
%hold on;fill([5e2 5e2 1e3 1e3 5e2]',[1e4 1e-4 1e-4 1e4 1e4]','c');hold off
xlabel('Freq (rad/sec)')
ylabel('Mag')
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
axis([1e-4 1e4 1e-4 1e4])
title(['Vel sens to Pos state, sen noise r=',num2str(r)])
legend('r=10^{-2}','r=10^{-4}','r=10^{-6}','r=10^{-8}','Location','NorthWest')
%subplot(212)
figure(6);clf
semilogx(f1,unwrap(angle(g1))*180/pi,f2,unwrap(angle(g2))*180/pi,...
    f3,unwrap(angle(g3))*180/pi,f4,unwrap(angle(g4))*180/pi,'Linewidth',2);hold off
xlabel('Freq (rad/sec)')
ylabel('Phase (deg)')
legend('r=10^{-2}','r=10^{-4}','r=10^{-6}','r=10^{-8}')
%bode(nn,dd);
axis([1e-4 1e4 0 200])
print -depsc -f5 filt5.eps;jpdf('filt5')
print -depsc -f6 filt6.eps;jpdf('filt6')
```

16.323 Lecture 12

Stochastic Optimal Control

- Kwaknernaak and Sivan Chapter 3.6, 5
- Bryson Chapter 14
- Stengel Chapter 5

Stochastic Optimal Control

- **Goal:** design optimal compensators for systems with incomplete and noisy measurements
  - Consider this first simplified step: assume that we have noisy system with perfect measurement of the state.

- **System dynamics:**

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

  - Assume that $\mathbf{w}(t)$ is a white Gaussian noise [20] $\Rightarrow \mathcal{N}(0, R_{\text{ww}})$
  - The initial conditions are random variables too, with

$$E[\mathbf{x}(t_0)] = 0, \text{ and } E[\mathbf{x}(t_0)\mathbf{x}^T(t_0)] = X_0$$

  - Assume that a perfect measure of $\mathbf{x}(t)$ is available for feedback.

- Given the noise in the system, need to modify our cost functions from before $\Rightarrow$ consider the **average** response of the closed-loop system

$$J_s = E\left\{\frac{1}{2}\mathbf{x}^T(t_f)P_{t_f}\mathbf{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}(\mathbf{x}^T(t)R_{\text{xx}}(t)\mathbf{x}(t) + \mathbf{u}^T(t)R_{\text{uu}}(t)\mathbf{u}(t))dt\right\}$$

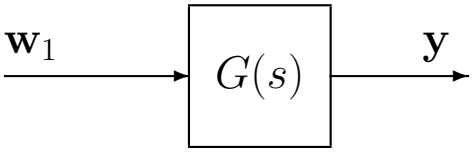
  - Average over all possible realizations of the disturbances.

- **Key observation:** since $\mathbf{w}(t)$ is white, then by definition, the correlation times-scales are very short compared to the system dynamics
  - Impossible to predict $\mathbf{w}(\tau)$ for $\tau > t$, even with perfect knowledge of the state for $\tau \leq t$
  - Furthermore, by definition, the system state $\mathbf{x}(t)$ encapsulates all past information about the system
  - Then the optimal controller for this case is **identical** to the deterministic one considered before.

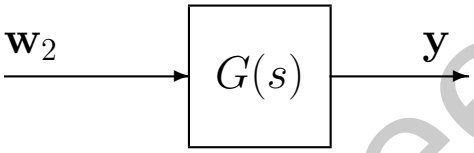
[20] 16.322 Notes

Spectral Factorization

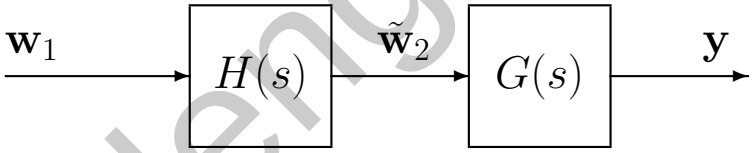
- Had the process noise $\mathbf{w}(t)$ had "color" (i.e., not white), then we need to include a **shaping filter** that captures the spectral content (i.e., temporal correlation) of the noise $\Phi(s)$
  - Previous picture: system is $\mathbf{y} = G(s)\mathbf{w}_1$, with white noise input



  - New picture: system is $\mathbf{y} = G(s)\mathbf{w}_2$, with shaped noise input



- Account for the spectral content using a shaping filter $H(s)$, so that the picture now is of a system $\mathbf{y} = G(s)H(s)\mathbf{w}_1$, with a white noise input



  - Then must design filter $H(s)$ so that the output is a noise $\tilde{\mathbf{w}}_2$ that has the frequency content that we need

- How design $H(s)$? **Spectral Factorization** – design a stable minimum phase linear transfer function that replicates the desired spectrum of $\mathbf{w}_2$.
  - Basis of approach: If $e_2 = H(s)e_1$ and e_1 is white, then the spectrum of e_2 is given by

$$\Phi_{e_2}(j\omega) = H(j\omega)H(-j\omega)\Phi_{e_1}(j\omega)$$

    where $\Phi_{e_1}(j\omega) = 1$ because it is white.

- Typically $\Phi_{w_2}(j\omega)$ will be given as an expression in ω^2, and we factor that into two parts, one of which is stable minimum phase, so if

$$\begin{aligned}\Phi_{w_2}(j\omega) &= \frac{2\sigma^2\alpha^2}{\omega^2+\alpha^2} \\ &= \frac{\sqrt{2}\sigma\alpha}{\alpha+j\omega}\cdot\frac{\sqrt{2}\sigma\alpha}{\alpha-j\omega} = H(j\omega)H(-j\omega)\end{aligned}$$

so clearly $H(s)=\frac{\sqrt{2}\sigma\alpha}{s+\alpha}$ which we write in state space form as

$$\begin{aligned}\dot{x}_H &= -\alpha x_H + \sqrt{2}\alpha\sigma w_1 \\ w_2 &= x_H\end{aligned}$$

- More generally, the shaping filter will be

$$\begin{aligned}\dot{\mathbf{x}}_H &= A_H\mathbf{x}_H + B_H\mathbf{w}_1 \\ \mathbf{w}_2 &= C_H\mathbf{x}_H\end{aligned}$$

which we then augment to the plant dynamics, to get:

$$\begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\mathbf{x}}_H\end{bmatrix} = \begin{bmatrix}A & B_wC_H\\ 0 & A_H\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{x}_H\end{bmatrix} + \begin{bmatrix}B_u\\ 0\end{bmatrix}\mathbf{u} + \begin{bmatrix}0\\ B_H\end{bmatrix}\mathbf{w}_1$$

$$\mathbf{y} = \begin{bmatrix}C_y & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{x}_H\end{bmatrix}$$

where the noise input $\mathbf{w}_1$ is a white Gaussian noise.

- Clearly this augmented system has the same form as the original system that we analyzed - there are just more states to capture the spectral content of the original shaped noise.

- Now consider the stochastic LQR problem for this case.
  - Modify the state weighting matrix so that

$$\tilde{R}_{\mathrm{xx}} = \begin{bmatrix} R_{\mathrm{xx}} & 0 \\ 0 & 0 \end{bmatrix}$$

    $\Rightarrow$ i.e. no weighting on the filter states – Why is that allowed?
  - Then, as before, the stochastic LQR solution for the augmented system is the same as the deterministic LQR solution (6–9)

$$\mathbf{u} = -\begin{bmatrix} K & K_d \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_H \end{bmatrix}$$

  - So the full state feedback controller requires access to the state in the shaping filter, which is fictitious and needs to be estimated

- Interesting result is that the gain K on the system states is **completely independent** of the properties of the disturbance
  - In fact, if the solution of the steady state Riccati equation in this case is partitioned as

$$P_{\mathrm{aug}} = \left[\begin{array}{c|c} P_{\mathrm{xx}} & P_{\mathrm{xx_H}} \\ \hline P_{\mathrm{x_Hx}} & P_{\mathrm{x_Hx_H}} \end{array}\right]$$

    it is easy to show that
    - ◇ P_{xx} can be solved for independently, and
    - ◇ Is the same as it would be in the deterministic case with the disturbances omitted [21]
  - Of course the control inputs that are also based on $\mathbf{x}_H$ will improve the performance of the system $\Rightarrow$ **disturbance feedforward**.

[21] K+S pg 262

Performance Analysis

- Recall that the specific initial conditions do not effect the LQR controller, but they do impact the cost-to-go from t_0
  - Consider the stochastic LQR problem, but with $\mathbf{w}(t) \equiv 0$ so that the only uncertainty is in the initial conditions
  - Have already shown that LQR cost can be written in terms of the solution of the Riccati equation (4–7):

$$\begin{aligned} J_{LQR} &= \frac{1}{2}\mathbf{x}^T(t_0)P(t_0)\mathbf{x}(t_0) \\ \Rightarrow J_s &= E\left\{\frac{1}{2}\mathbf{x}^T(t_0)P(t_0)\mathbf{x}(t_0)\right\} \\ &= \frac{1}{2}E\left\{\mathtt{trace}[P(t_0)\mathbf{x}(t_0)\mathbf{x}^T(t_0)]\right\} \\ &= \frac{1}{2}\mathtt{trace}[P(t_0)X_0] \end{aligned}$$

  which gives expected cost-to-go with uncertain IC.

- Now return to case with $\mathbf{w} \neq 0$ – consider the average performance of the stochastic LQR controller.

- To do this, recognize that if we apply the LQR control, we have a system where the cost is based on $\mathbf{z}^T R_{zz}\mathbf{z} = \mathbf{x}^T R_{xx}\mathbf{x}$ for the closed-loop system:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= (A(t) - B_u(t)K(t))\mathbf{x}(t) + B_w(t)\mathbf{w}(t) \\ \mathbf{z}(t) &= C_z(t)\mathbf{x}(t) \end{aligned}$$

- This is of the form of a linear time-varying system driven by white Gaussian noise – called a **Gauss-Markov Random process**[22].

[22] Bryson 11.4

- For a Gauss-Markov system we can predict the **mean square value** of the state $X(t) = E[\mathbf{x}(t)\mathbf{x}(t)^T]$ over time using $X(0) = X_0$ and

$$\dot{X}(t) = [A(t) - B_u(t)K(t)]\,X(t) + X(t)\,[A(t) - B_u(t)K(t)]^T + B_w R_{\rm ww} B_w^T$$

  – Matrix **differential Lyapunov Equation**.

- Can also extract the mean square control values using

$$E[\mathbf{u}(t)\mathbf{u}(t)^T] = K(t)X(t)K(t)^T$$

- Now write performance evaluation as:

$$\begin{aligned} J_s &= \frac{1}{2}E\left\{\mathbf{x}^T(t_f)P_{t_f}\mathbf{x}(t_f) + \int_{t_0}^{t_f}(\mathbf{x}^T(t)R_{\rm xx}(t)\mathbf{x}(t) + \mathbf{u}^T(t)R_{\rm uu}(t)\mathbf{u}(t))dt\right\} \\ &= \frac{1}{2}E\left\{\mathtt{trace}\left[P_{t_f}\mathbf{x}(t_f)\mathbf{x}^T(t_f) + \int_{t_0}^{t_f}(R_{\rm xx}(t)\mathbf{x}(t)\mathbf{x}^T(t) + R_{\rm uu}(t)\mathbf{u}(t)\mathbf{u}^T(t))dt\right]\right\} \\ &= \frac{1}{2}\mathtt{trace}\left[P_{t_f}X(t_f) + \int_{t_0}^{t_f}(R_{\rm xx}(t)X(t) + R_{\rm uu}(t)K(t)X(t)K(t)^T)dt\right] \end{aligned}$$

- Not too useful in this form, but if $P(t)$ is the solution of the LQR Riccati equation, then can show that the cost can be written as:

$$J_s = \frac{1}{2}\mathtt{trace}\left\{P(t_0)X(t_0) + \int_{t_0}^{t_f}(P(t)B_w R_{\rm ww} B_w^T)dt\right\}$$

  – First part, $\frac{1}{2}\mathtt{trace}\{P(t_0)X(t_0)\}$ is the same cost-to-go from the uncertain initial condition that we identified on 11–5

  – Second part shows that the cost increases as a result of the process noise acting on the system.

- **Sketch of Proof:** first note that

$$P(t_0)X(t_0) - P_{t_f}X(t_f) + \int_{t_0}^{t_f} \frac{d}{dt}(P(t)X(t))dt = 0$$

$$\begin{aligned} J_s &= \frac{1}{2}\mathtt{trace}\left[P_{t_f}X(t_f) + P(t_0)X(t_0) - P_{t_f}X(t_f)\right] \\ &+ \frac{1}{2}\mathtt{trace}\left[\int_{t_0}^{t_f}\{R_{\rm xx}(t)X(t) + R_{\rm uu}(t)K(t)X(t)K(t)^T\}dt\right] \\ &+ \frac{1}{2}\mathtt{trace}\left[\int_{t_0}^{t_f}\{\dot{P}(t)X(t) + P(t)\dot{X}(t)\}dt\right] \end{aligned}$$

and (first reduces to standard CARE if $K(t) = R_{\rm uu}^{-1}B_u^T P(t)$)

$$\begin{aligned} -\dot{P}(t)X(t) &= (A - B_uK(t))^T P(t)X(t) + P(t)(A - B_uK(t))X(t) \\ &\quad + R_{\rm xx}X(t) + K(t)^T R_{\rm uu}K(t)X(t) \end{aligned}$$

$$\begin{aligned} P(t)\dot{X}(t) &= P(t)(A - B_uK(t))X(t) + P(t)X(t)(A - B_uK(t))^T \\ &\quad + P(t)B_w R_{\rm ww}B_w^T \end{aligned}$$

- Rearrange terms within the trace and then cancel terms to get final result.

Steady State Values

- Problems exist if we set $t_0 = 0$ and $t_f \to \infty$ because performance will be infinite
  - Modify the cost to consider the time-average

$$J_a = \lim_{t_f \to \infty} \frac{1}{t_f - t_0} J_s$$

  - No impact on necessary conditions since this is still a fixed end-time problem.
  - But now the initial conditions become irrelevant, and we only need focus on the integral part of the cost.

- For LTI system with stationary process noise (constant $R_{\rm ww}$) and well-posed time-invariant control problem (steady gain $\mathbf{u}(t) = -K_{ss}\mathbf{x}(t)$) mean square value of state settles down to a constant

$$\lim_{t_f \to \infty} X(t) = X_{ss}$$

$$0 = (A - B_u K_{ss}) X_{ss} + X_{ss} (A - B_u K_{ss})^T + B_w R_{\rm ww} B_w^T$$

  - Can show that **time-averaged mean square performance** is

$$J_a = \frac{1}{2} \mathtt{trace} \left([R_{\rm xx} + K_{ss}^T R_{\rm uu} K_{ss}] X_{ss} \right)$$

$$\equiv \frac{1}{2} \mathtt{trace} [P_{ss} B_w R_{\rm ww} B_w^T]$$

- **Main point:** this gives a direct path to computing the expected performance of a closed-loop system
  - Process noise enters into computation of X_{ss}

Missile Example

- Consider a missile roll attitude control system with ω the roll angular velocity, δ the aileron deflection, Q the aileron effectiveness, and ϕ the roll angle, then

$$\dot{\delta} = u \qquad \dot{\omega} = -\frac{1}{\tau}\omega + \frac{Q}{\tau}\delta + n(t) \qquad \dot{\phi} = \omega$$

  where $n(t)$ is a noise input.

- Then this can be written as:

$$\begin{bmatrix} \dot{\delta} \\ \dot{\omega} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -1/\tau & Q/\tau & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta \\ \omega \\ \phi \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} n$$

- Use $\tau = 1$, $Q = 10$, $R_{\rm uu} = 1/(\pi)^2$ and

$$R_{\rm xx} = \begin{bmatrix} (\pi/12)^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & (\pi/180)^2 \end{bmatrix}$$

  then solve LQR problem to get feedback gains:

```
K=lqr(A,B,Rxx,Ruu)
```

$$K = [26.9 \quad 29.0 \quad 180.0]$$

- Then if $n(t)$ has a spectral density of 1000 (deg/sec^2)$^2\cdot$ sec [23]

- Find RMS response of the system from

```
X=lyap(A-B*K,Bw*Rww*Bw')
```

$$X = \begin{bmatrix} 95 & -42 & -7 \\ -42 & 73 & 0 \\ -7 & 0 & 0.87 \end{bmatrix}$$

  and that $\sqrt{E[\phi^2]} \approx 0.93$deg

[23] Process noise input to a derivative of ω, so the units of $n(t)$ must be deg/sec^2, but since $E[n(t)n(\tau)] = R_{\rm ww}\delta(t-\tau)$ and $\int \delta(t)dt = 1$, then the units of $\delta(t)$ are 1/sec and thus the units of $R_{\rm ww}$ are (rad/sec^2)$^2\cdot$ sec=rad^2/sec^3

Full Control Problem

- **Goal:** design an optimal controller for a system with **incomplete and noisy measurements**

- **Setup:** for the system (possibly time-varying)

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B_u\mathbf{u} + B_w\mathbf{w} \\ \mathbf{z} &= C_z\mathbf{x} \\ \mathbf{y} &= C_y\mathbf{x} + \mathbf{v}\end{aligned}$$

  with
  - White, Gaussian noises $\mathbf{w} \sim \mathcal{N}(0, R_{\rm ww})$ and $\mathbf{v} \sim \mathcal{N}(0, R_{\rm vv})$, with $R_{\rm ww} > 0$ and $R_{\rm vv} > 0$
  - Initial conditions $\mathbf{x}(t_0)$, a stochastic vector with $E[\mathbf{x}(t_0)] = \bar{\mathbf{x}}_0$ and $E[(\mathbf{x}(t_0) - \bar{\mathbf{x}}_0)(\mathbf{x}(t_0) - \bar{\mathbf{x}}_0)^T] = Q_0$ so that

$$\mathbf{x}(t_0) \sim N(\bar{\mathbf{x}}_0, Q_0)$$

- **Cost:**

$$J = E\left\{\frac{1}{2}\mathbf{x}^T(t_f)P_{t_f}\mathbf{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}(\mathbf{z}^T(t)R_{\rm zz}\mathbf{z}(t) + \mathbf{u}^T(t)R_{\rm uu}\mathbf{u}(t))dt\right\}$$

  with $R_{\rm zz} > 0$, $R_{\rm uu} > 0$, $P_{t_f} \geq 0$

- **Stochastic Optimal Output Feedback Problem:** Find

$$\mathbf{u}(t) = \mathbf{f}[\mathbf{y}(\tau), t_0 \leq \tau \leq t] \qquad t_0 \leq t \leq t_f$$

  that minimizes J

- The solution is the Linear Quadratic Gaussian Controller, which uses
  - **LQE** (10–15) to get optimal state estimates $\hat{\mathbf{x}}(t)$ from $\mathbf{y}(t)$ using gain $L(t)$
  - **LQR** to get the optimal feedback control $\mathbf{u}(t) = -K(t)\mathbf{x}$
  - **Separation principle** to implement $\mathbf{u}(t) = -K(t)\hat{\mathbf{x}}(t)$

Solution: LQG

- Regulator: $\mathbf{u}(t) = -K(t)\hat{\mathbf{x}}(t)$

$$\begin{aligned} K(t) &= R_{\text{uu}}^{-1} B_u^T P(t) \\ -\dot{P}(t) &= A^T P(t) + P(t)A + C_z^T R_{\text{zz}} C_z - P(t) B_u R_{\text{uu}}^{-1} B_u^T P(t) \\ P(t_f) &= P_{t_f} \end{aligned}$$

- Estimator from:

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}} + B_u \mathbf{u} + L(t)(\mathbf{y}(t) - C_y \hat{\mathbf{x}}(t))$$

where $\hat{\mathbf{x}}(t_0) = \bar{\mathbf{x}}_0$ and $Q(t_0) = Q_0$

$$\begin{aligned} \dot{Q}(t) &= AQ(t) + Q(t)A^T + B_w R_{\text{ww}} B_w^T - Q(t) C_y^T R_{\text{vv}}^{-1} C_y Q(t) \\ L(t) &= Q(t) C_y^T R_{\text{vv}}^{-1} \end{aligned}$$

- A compact form of the compensator is:

$$\begin{aligned} \dot{\mathbf{x}}_c &= A_c \mathbf{x}_c + B_c \mathbf{y} \\ \mathbf{u} &= -C_c \mathbf{x}_c \end{aligned}$$

with $\mathbf{x}_c \equiv \hat{\mathbf{x}}$ and

$$\begin{aligned} A_c &= A - B_u K(t) - L(t) C_y \\ B_c &= L(t) \\ C_c &= K(t) \end{aligned}$$

- Valid for SISO and MIMO systems. Plant dynamics can also be time-varying, but suppressed for simplicity.
  - Obviously compensator is constant if we use the steady state regulator and estimator gains for an LTI system.

Infinite Horizon LQG

- Assuming LTI plant
- As with the stochastic LQR case, use time averaged cost
  - To ensure that estimator settles down, must take $t_0 \to -\infty$ and $t_f \to \infty$, so that for any t, $t_0 \ll t \ll t_f$

$$\bar{J} = \lim_{\substack{t_f \to \infty \\ t_0 \to -\infty}} \frac{1}{t_f - t_0} J$$

  - Again, this changes the cost, but not the optimality conditions

- Analysis of $\bar{J}$ shows that it can be evaluated as

$$\begin{aligned} \bar{J} &= E[\mathbf{z}^T(t) R_{\rm zz} \mathbf{z}(t) + \mathbf{u}^T(t) R_{\rm uu} \mathbf{u}(t)] \\ &= \mathtt{Tr}[P_{ss} L_{ss} R_{\rm vv} L_{ss}^T + Q_{ss} C_z^T R_{\rm zz} C_z] \\ &= \mathtt{Tr}[P_{ss} B_w R_{\rm ww} B_w^T + Q_{ss} K_{ss}^T R_{\rm uu} K_{ss}] \end{aligned}$$

  where P_{ss} and Q_{ss} are the steady state solutions of

$$\begin{aligned} A^T P_{ss} + P_{ss} A + C_z^T R_{\rm zz} C_z - P_{ss} B_u R_{\rm uu}^{-1} B_u^T P_{ss} &= 0 \\ A Q_{ss} + Q_{ss} A^T + B_w R_{\rm ww} B_w^T - Q_{ss} C_y^T R_{\rm vv}^{-1} C_y Q_{ss} &= 0 \end{aligned}$$

  with

$$K_{ss} = R_{\rm uu}^{-1} B_u^T P_{ss} \quad \text{and} \quad L_{ss} = Q_{ss} C_y^T R_{\rm vv}^{-1}$$

- Can evaluate the steady state performance from the solution of 2 Riccati equations
  - More complicated than stochastic LQR because $\bar{J}$ must account for performance degradation associated with estimation error.
  - Since in general $\hat{\mathbf{x}}(t) \neq \mathbf{x}(t)$, have two contributions to the cost
    - ◇ Regulation error $\mathbf{x} \neq 0$
    - ◇ Estimation error $\tilde{\mathbf{x}} \neq 0$

Interpretations

- Note that

$$\begin{aligned}\bar{J} &= \mathtt{Tr}[P_{ss}L_{ss}R_{\rm vv}L_{ss}^T + Q_{ss}C_z^T R_{\rm zz}C_z] \\ &= \mathtt{Tr}[P_{ss}B_w R_{\rm ww}B_w^T + Q_{ss}K_{ss}^T R_{\rm uu}K_{ss}]\end{aligned}$$

  both of which contain terms that are functions of the control and estimation problems.

- To see how both terms contribute, let the regulator get very fast $\Rightarrow R_{\rm uu} \to 0$. A full analysis requires that we then determine what happens to P_{ss} and thus $\bar{J}$. But what is clear is that:

$$\lim_{R_{\rm uu}\to 0} \bar{J} \geq \mathtt{Tr}[Q_{ss}C_z^T R_{\rm zz}C_z]$$

  which is independent of $R_{\rm uu}$
  - Thus even in the limit of no control penalty, the performance is **lower bounded** by term associated with estimation error Q_{ss}.
- Similarly, can see that $\lim_{R_{\rm vv}\to 0} \bar{J} \geq \mathtt{Tr}[P_{ss}B_w R_{\rm ww}B_w^T]$ which is related to the regulation error and provides a lower bound on the performance with a fast estimator
  - Note that this is the average cost for the stochastic LQR problem.

- Both cases illustrate that it is futile to make either the estimator or regulator much "faster" than the other
  - The ultimate performance is limited, and you quickly reach the "knee in the curve" for which further increases in the authority of one over the other provide diminishing returns.
  - Also suggests that it is not obvious that either one of them should be faster than the other.

- **Rule of Thumb:** for given $R_{\rm zz}$ and $R_{\rm ww}$, select $R_{\rm uu}$ and $R_{\rm vv}$ so that the performance contributions due to the estimation and regulation error are comparable.

Separation Theorem

- Now consider what happens when the control $\mathbf{u} = -K\mathbf{x}$ is changed to the new control $\mathbf{u} = -K\hat{\mathbf{x}}$ (same K).
  - Assume steady state values here, but not needed.
  - Previous looks at this would have analyzed the closed-loop stability, as follows, but we also want to analyze performance.

$$\begin{aligned} \text{plant}: \quad & \dot{\mathbf{x}} = A\mathbf{x} + B_u\mathbf{u} + B_w\mathbf{w} \\ & \mathbf{z} = C_z\mathbf{x} \\ & \mathbf{y} = C_y\mathbf{x} + \mathbf{v} \\ \text{compensator}: \quad & \dot{\mathbf{x}}_c = A_c\mathbf{x}_c + B_c\mathbf{y} \\ & \mathbf{u} = -C_c\mathbf{x}_c \end{aligned}$$

- Which give the closed-loop dynamics

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_c \end{bmatrix} = \begin{bmatrix} A & -B_uC_c \\ B_cC_y & A_c \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_c \end{bmatrix} + \begin{bmatrix} B_w & 0 \\ 0 & B_c \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix}$$

$$\mathbf{z} = \begin{bmatrix} C_z & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_c \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} C_y & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_c \end{bmatrix} + \mathbf{v}$$

- It is not obvious that this system will even be stable: $\lambda_i(A_{\rm cl}) < 0$?
  - To analyze, introduce $\mathbf{n} = \mathbf{x} - \mathbf{x}_c$, and the *similarity transform*

$$T = \begin{bmatrix} I & 0 \\ I & -I \end{bmatrix} = T^{-1} \qquad \Rightarrow \qquad \begin{bmatrix} \mathbf{x} \\ \mathbf{n} \end{bmatrix} = T \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_c \end{bmatrix}$$

  so that $A_{\rm cl} \;\Rightarrow\; TA_{\rm cl}T^{-1} \equiv \overline{A_{\rm cl}}$ and when you work through the math, you get

$$\overline{A_{\rm cl}} = \begin{bmatrix} A - B_uK & B_uK \\ 0 & A - LC_y \end{bmatrix}$$

- **Absolutely key points:**
  1. $\lambda_i(A_{\rm cl}) \equiv \lambda_i(\overline{A_{\rm cl}})$
  2. $\overline{A_{\rm cl}}$ is block upper triangular, so can find poles by inspection:

$$\det(sI - \overline{A_{\rm cl}}) = \det(sI - (A - B_u K)) \cdot \det(sI - (A - LC_y))$$

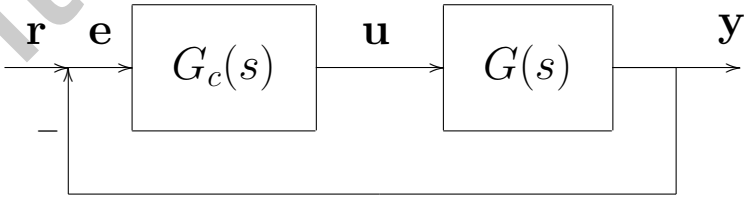
The closed-loop poles of the system consist of the union of the regulator and estimator poles

  - This shows that we can design **any** estimator and regulator separately with confidence that the combination will stabilize the system.
    - ◇ Also means that the LQR/LQE problems decouple in terms of being able to predict the stability of the overall closed-loop system.

- Let $G_c(s)$ be the compensator transfer function (matrix) where

$$\mathbf{u} = -C_c(sI - A_c)^{-1}B_c\mathbf{y} = -G_c(s)\mathbf{y}$$

  - Reason for this is that when implementing the controller, we often do not just feedback $-\mathbf{y}(t)$, but instead have to include a *reference command* $\mathbf{r}(t)$
  - Use **servo approach** and feed back $\mathbf{e}(t) = \mathbf{r}(t) - \mathbf{y}(t)$ instead



  - So now $\mathbf{u} = G_c\mathbf{e} = G_c(\mathbf{r}-\mathbf{y})$, and if $\mathbf{r} = 0$, then have $\mathbf{u} = G_c(-\mathbf{y})$

- Important points:
  - Closed-loop system will be stable, but the compensator dynamics need not be.
  - Often very simple and useful to provide classical interpretations of the compensator dynamics $G_c(s)$.

Performance Optimality

- Performance optimality of this strategy is a little harder to establish
  - Now saying more than just that the separation principle is a "good" idea $\Rightarrow$ are trying to say that it is the "best" possible solution

- **Approach:**
  - Rewrite cost and system in terms of the estimator states and dynamics $\Rightarrow$ recall we have access to these
  - Design a stochastic LQR for this revised system $\Rightarrow$ full state feedback on $\hat{\mathbf{x}}(t)$

- Start with the cost (use a similar process for the terminal cost)

$$\begin{aligned} E[\mathbf{z}^T R_{zz}\mathbf{z}] &= E[\mathbf{x}^T R_{xx}\mathbf{x}] && \{\pm\hat{\mathbf{x}}\} \\ &= E[(\mathbf{x}-\hat{\mathbf{x}}+\hat{\mathbf{x}})^T R_{xx}(\mathbf{x}-\hat{\mathbf{x}}+\hat{\mathbf{x}})] && \{\tilde{\mathbf{x}}=\mathbf{x}-\hat{\mathbf{x}}\} \\ &= E[\tilde{\mathbf{x}}^T R_{xx}\tilde{\mathbf{x}}] + 2E[\tilde{\mathbf{x}}^T R_{xx}\hat{\mathbf{x}}] + E[\hat{\mathbf{x}}^T R_{xx}\hat{\mathbf{x}}] \end{aligned}$$

- Note that $\hat{\mathbf{x}}(t)$ is the minimum mean square estimate of $\mathbf{x}(t)$ given $\mathbf{y}(\tau)$, $\mathbf{u}(\tau)$, $t_0 \le \tau \le t$.
  - Key property of that estimate is that $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ are uncorrelated[24]

$$E[\tilde{\mathbf{x}}^T R_{xx}\hat{\mathbf{x}}] = \mathtt{trace}[E\{\tilde{\mathbf{x}}\hat{\mathbf{x}}^T\}R_{xx}] = 0$$

- Also,

$$E[\tilde{\mathbf{x}}^T R_{xx}\tilde{\mathbf{x}}] = E[\mathtt{trace}(R_{xx}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T)] = \mathtt{trace}(R_{xx}Q)$$

  where Q is the solution of the LQE Riccati equation (11–11)

- So, in summary we have:

$$E[\mathbf{x}^T R_{xx}\mathbf{x}] = \mathtt{trace}(R_{xx}Q) + E[\hat{\mathbf{x}}^T R_{xx}\hat{\mathbf{x}}]$$

[24] Gelb, pg 112

- Now the main part of the cost function can be rewritten as

$$\begin{aligned} J &= E\left\{\frac{1}{2}\int_{t_0}^{t_f}(\mathbf{z}^T(t)R_{\rm zz}\mathbf{z}(t)+\mathbf{u}^T(t)R_{\rm uu}\mathbf{u}(t))dt\right\} \\ &= E\left\{\frac{1}{2}\int_{t_0}^{t_f}(\hat{\mathbf{x}}^T(t)R_{\rm xx}\hat{\mathbf{x}}(t)+\mathbf{u}^T(t)R_{\rm uu}\mathbf{u}(t))dt\right\} \\ &\quad +\frac{1}{2}\int_{t_0}^{t_f}(\mathtt{trace}(R_{\rm xx}Q))dt \end{aligned}$$

  - The last term is independent of the control $\mathbf{u}(t)$ $\Rightarrow$ it is only a function of the estimation error
  - Objective now is to choose the control $\mathbf{u}(t)$ to minimize the first term

- But first we need another key fact[25]: If the optimal estimator is

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}}(t)+B_u\mathbf{u}(t)+L(t)(\mathbf{y}(t)-C_y\hat{\mathbf{x}}(t))$$

  then by definition, the innovations process

$$\mathbf{i}(t) \equiv \mathbf{y}(t)-C_y\hat{\mathbf{x}}(t)$$

  is a white Gaussian process, so that $\mathbf{i}(t)\sim\mathcal{N}(0, R_{\rm vv}+C_yQC_y^T)$

- Then we can rewrite the estimator as

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}}(t)+B_u\mathbf{u}(t)+L(t)\mathbf{i}(t)$$

  which is an LTI system with $\mathbf{i}(t)$ acting as the process noise through a computable $L(t)$.

[25] Gelb, pg 317

- So combining the above, we must pick $\mathbf{u}(t)$ to minimize

$$J = E\left\{\frac{1}{2}\int_{t_0}^{t_f}(\hat{\mathbf{x}}^T(t)R_{\rm xx}\hat{\mathbf{x}}(t) + \mathbf{u}^T(t)R_{\rm uu}\mathbf{u}(t))dt\right\} + \text{term ind. of } \mathbf{u}(t)$$

subject to the dynamics

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}}(t) + B_u\mathbf{u}(t) + L(t)\mathbf{i}(t)$$

  - Which is a strange looking Stochastic LQR problem
  - As we saw before, the solution is independent of the driving process noise

$$\mathbf{u}(t) = -K(t)\hat{\mathbf{x}}(t)$$

  - Where $K(t)$ is found from the LQR with the data A, B_u, $R_{\rm xx}$, and $R_{\rm uu}$, and thus will be identical to the original problem.

- Combination of LQE/LQR gives performance optimal result.

Simple Example

$$\begin{aligned}\dot{\mathbf{x}} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u + \begin{bmatrix} 0 \\ 1 \end{bmatrix}w \\ z &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\mathbf{x} \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x} + v\end{aligned}$$

where in the LQG problem we have

$$R_{zz} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad R_{uu} = 1 \qquad R_{vv} = 1 \qquad R_{ww} = 1$$

- Solve the SS LQG problem to find that

$$\begin{aligned}\mathtt{Tr}[P_{ss}L_{ss}R_{vv}L_{ss}^T] &= 8.0 \quad \mathtt{Tr}[Q_{ss}C_z^T R_{zz}C_z] = 2.8 \\ \mathtt{Tr}[P_{ss}B_w R_{ww}B_w^T] &= 1.7 \quad \mathtt{Tr}[Q_{ss}K_{ss}^T R_{uu}K_{ss}] = 9.1\end{aligned}$$

- Suggests to me that we need to improve the estimation error $\Rightarrow$ that R_{vv} is too large. Repeat with

$$R_{zz} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad R_{uu} = 1 \qquad R_{vv} = 0.1 \qquad R_{ww} = 1$$

$$\begin{aligned}\mathtt{Tr}[P_{ss}L_{ss}R_{vv}L_{ss}^T] &= 4.1 \quad \mathtt{Tr}[Q_{ss}C_z^T R_{zz}C_z] = 1.0 \\ \mathtt{Tr}[P_{ss}B_w R_{ww}B_w^T] &= 1.7 \quad \mathtt{Tr}[Q_{ss}K_{ss}^T R_{uu}K_{ss}] = 3.7\end{aligned}$$

and

$$R_{zz} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad R_{uu} = 1 \qquad R_{vv} = 0.01 \qquad R_{ww} = 1$$

$$\begin{aligned}\mathtt{Tr}[P_{ss}L_{ss}R_{vv}L_{ss}^T] &= 3.0 \quad \mathtt{Tr}[Q_{ss}C_z^T R_{zz}C_z] = 0.5 \\ \mathtt{Tr}[P_{ss}B_w R_{ww}B_w^T] &= 1.7 \quad \mathtt{Tr}[Q_{ss}K_{ss}^T R_{uu}K_{ss}] = 1.7\end{aligned}$$

- LQG analysis code

```
A=[0 1;0 0];%
Bu=[0 1]';%
Bw=[0 1]'; %
Cy=[1 0];%
Cz=[1 0;0 1];%
Rww=1;%
Rvv=1;%
Rzz=diag([1 1]);%
Ruu=1;%
[K,P]=lqr(A,Bu,Cz*Rzz*Cz',Ruu);%
[L,Q]=lqr(A',Cy',Bw*Rww*Bw',Rvv);L=L';%
N1=trace(P*(L*Rvv*L'))%
N2=trace(Q*(Cz'*Rzz*Cz))%
N3=trace(P*(Bw*Rww*Bw'))%
N4=trace(Q*(K'*Ruu*K))%
[N1 N2;N3 N4]
```

- Consider the linearized longitudinal dynamics of a hypothetical helicopter. The model of the helicopter requires four state variables:
  - $\theta(t)$:fuselage pitch angle (radians)
  - q(t):pitch rate (radians/second)
  - u(t):horizontal velocity of CG (meters/second)
  - x(t):horizontal distance of CG from desired hover (meters)

  The control variable is:
  - δ (t): tilt angle of rotor thrust vector (radians)

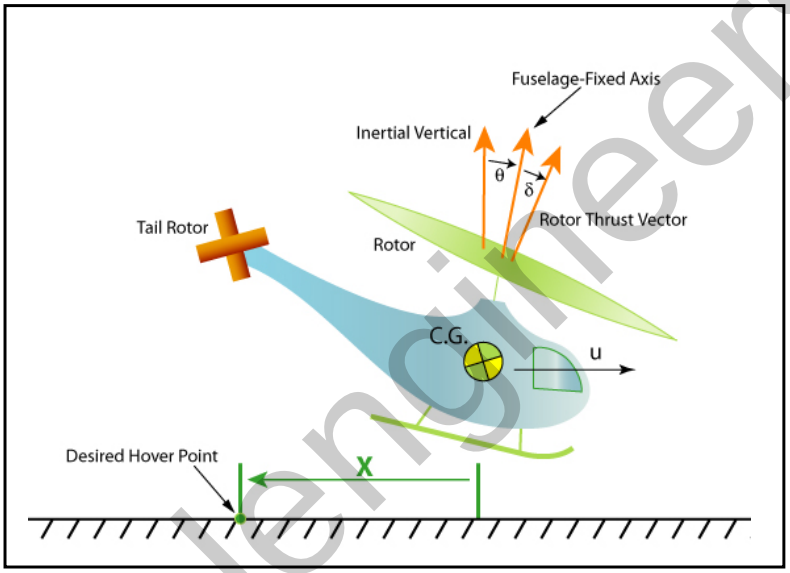


Figure by MIT OpenCourseWare.

Figure 12.1: Helicopter in Hover

- The linearized equation of motion are:

$$\begin{aligned}
\dot{\theta}(t) &= q(t) \\
\dot{q}(t) &= -0.415q(t) - 0.011u(t) + 6.27\delta(t) - 0.011w(t) \\
\dot{u}(t) &= 9.8\theta(t) - 1.43q(t) - .0198u(t) + 9.8\delta(t) - 0.0198w(t) \\
\dot{x}(t) &= u(t)
\end{aligned}$$

  - $w(t)$ represents a horizontal wind disturbance
  - Model $w(t)$ as the output of a first order system driven by zero mean, continuous time, unit intensity Gaussian white noise $\xi(t)$:

$$\dot{w}(t) = -0.2w(t) + 6\xi(t)$$

- First, treat original (non-augmented) plant dynamics.
  - Design LQR controller so that an initial hover position error, $x(0) = 1$ m is reduced to zero (to within 5%) in approximately 4 sec.

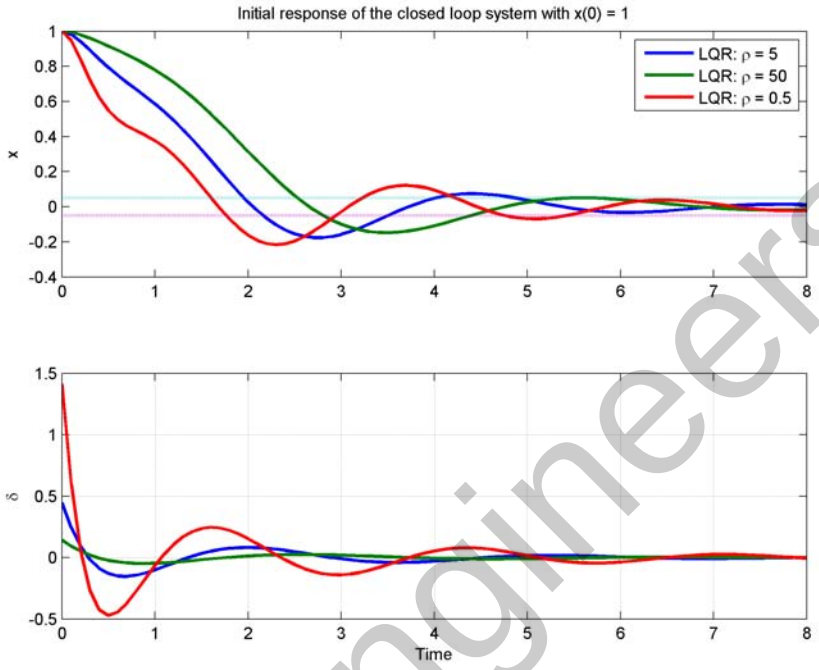


Figure 12.2: Results show that $R_{\rm uu} = 5$ gives reasonable performance.

- Augment the noise model, and using the same control gains, form the closed-loop system which includes the wind disturbance $w(t)$ as part of the state vector.
- Solve necessary Lyapunov equations to determine the (steady-state) variance of the position hover error, $x(t)$ and rotor angle $\delta(t)$.
  - Without feedforward:

$$\sqrt{E[x^2]} = 0.048 \qquad \sqrt{E[\delta^2]} = 0.017$$

- Then design a LQR for the augmented system and repeat the process.
  - With feedforward:

$$\sqrt{E[x^2]} = 0.0019 \qquad \sqrt{E[\delta^2]} = 0.0168$$

- Now do stochastic simulation of closed-loop system using $\Delta t = 0.1$.
  - Note the subtly here that the design was for a continuous system, but the simulation will be discrete
  - Are assuming that the integration step is constant.
  - Need to create ζ using the `randn` function, which gives zero mean unit variance Gaussian noise.
  - To scale it correctly for a discrete simulation, multiply the output of `randn` by $1/\sqrt{\Delta t}$, where Δt is the integration step size.[26]
  - Could also just convert the entire system to its discrete time equivalent, and then use a process noise that has a covariance

$$Q_d = R_{\text{ww}}/\Delta t$$

[26] Franklin and Powell, *Digital Control of Dynamic Systems*

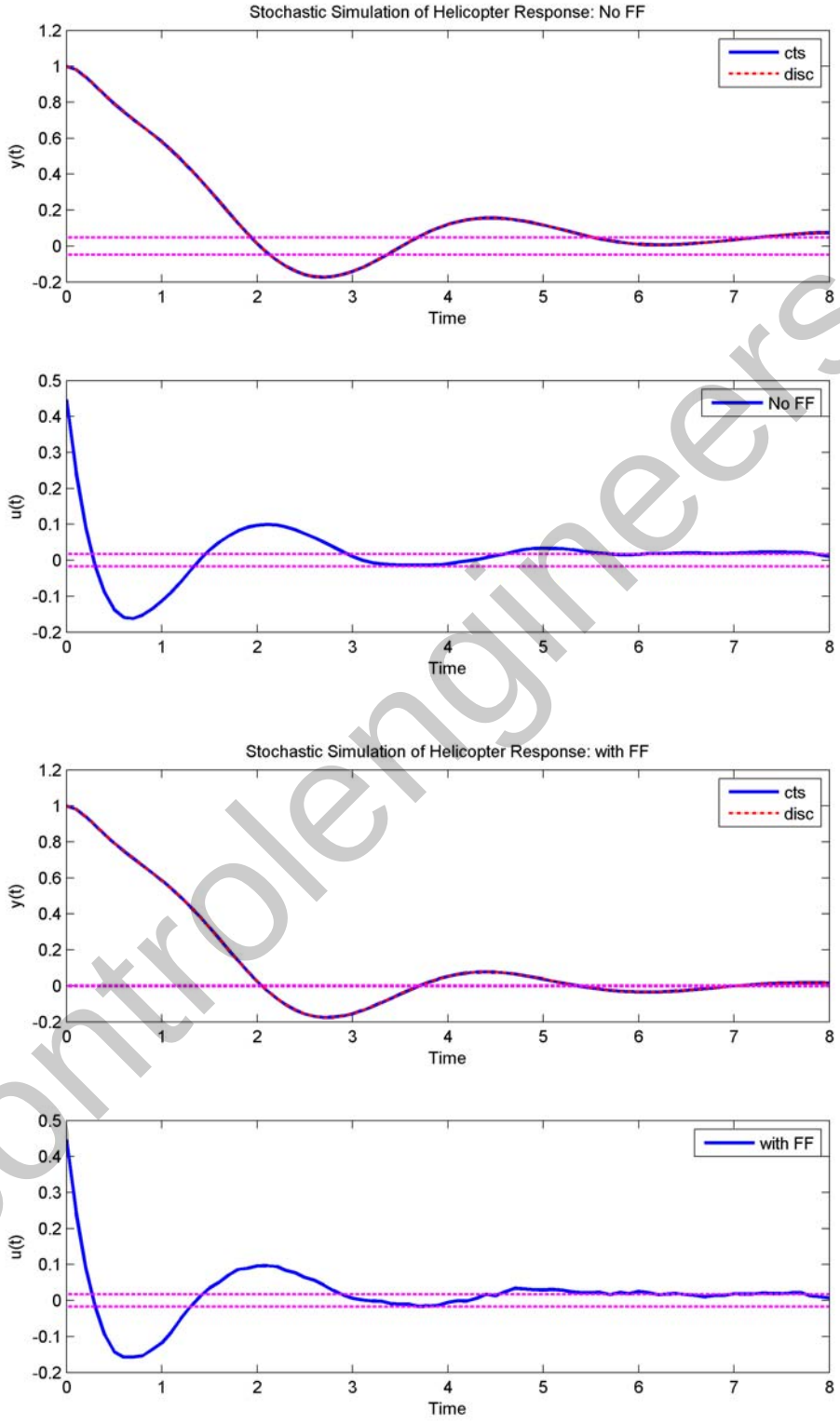


Figure 12.3: Stochastic Simulations with and without disturbance feedforward.

Helicopter stochastic simulation

```
% 16.323 Spring 2008
% Stochastic Simulation of Helicopter LQR
% Jon How
%
clear all, clf, randn('seed',sum(100*clock));
% linearized dynamics of the system
A = [ 0 1 0 0; 0 -0.415 -0.011 0;9.8 -1.43 -0.0198 0;0 0 1 0];
Bw = [0 -0.011 -0.0198 0]';
Bu = [0 6.27 9.8 0]';
Cz = [0 0 0 1];
Rxx = Cz'*Cz;
rho = 5;
Rww=1;

% lqr control
[K,S,E]=lqr(A,Bu,Rxx,rho);
[K2,S,E]=lqr(A,Bu,Rxx,10*rho);
[K3,S,E]=lqr(A,Bu,Rxx,rho/10);

% initial response with given x0
x0 = [0 0 0 1]';
Ts=0.1; % small discrete step to simulate the cts dynamics
tf=8;t=0:Ts:tf;
[y,x] = initial(A-Bu*K,zeros(4,1),Cz,0,x0,t);
[y2,x2] = initial(A-Bu*K2,zeros(4,1),Cz,0,x0,t);
[y3,x3] = initial(A-Bu*K3,zeros(4,1),Cz,0,x0,t);
subplot(211), plot(t,[y y2 y3],[0 8],.05*[1 1],':',[0 8],.05*[-1 -1],':','LineWidth',2)
ylabel('x');title('Initial response of the closed loop system with x(0) = 1')
h = legend(['LQR: \rho = ',num2str(rho)],['LQR: \rho = ',num2str(rho*10)],['LQR: \rho = ',num2str(rho/10)]);
axes(h)
subplot(212), plot(t,[(K*x')' (K2*x2')' (K3*x3')'],'LineWidth',2);grid on
xlabel('Time'), ylabel('\delta')
print -r300 -dpng  heli1.png

% shaping filter
Ah=-0.2;Bh=6;Ch=1;
% augment the filter dyanmics
Aa = [A Bw*Ch; zeros(1,4) Ah];
Bua = [Bu;0];
Bwa = [zeros(4,1); Bh];
Cza = [Cz 0];
Ka = [K 0]; % i.e. no dist FF
Acla = Aa-Bua*Ka; % close the loop using NO dist FF
Pass = lyap(Acla,Bwa*Rww*Bwa'); % compute SS response to the dist
vx = Cza*Pass*Cza'; % state resp
vd = Ka*Pass*Ka'; % control resp

zeta = sqrt(Rww/Ts)*randn(length(t),1); % discrete equivalent noise
[y,x] = lsim(Acla,Bwa,Cza,0,zeta,t,[x0;0]); % cts closed-loop sim
%
% second simulation approach: discrete time
%
Fa=c2d(ss(Acla,Bwa,Cza,0),Ts); % discretize the closed-loop dynamics
[dy,dx] = lsim(Fa,zeta,[],[x0;0]); % stochastic sim in discrete time
u = Ka*x'; % find control commands given the state response

% disturbance FF
[KK,SS,EE]=lqr(Aa,Bua,Cza'*Cza,rho); % now K will have dist FF
Acl=Aa-Bua*KK;
PP=lyap(Acl,Bwa*Rww*Bwa');
vxa = Cza*PP*Cza';
vda = KK*PP*KK';
[ya,xa] = lsim(Acl,Bwa,Cza,0,zeta,t,[x0;0]); % cts sim
F=c2d(ss(Acl,Bwa,Cza,0),Ts); % discretize the closed-loop dynamics
[dya,dxa] = lsim(F,zeta,[],[x0;0]); % stochastic sim in discrete time
ua = KK*xa'; % find control commands given the state response

```

```
68  figure(2);
69  subplot(211)
70  plot(t,y,'LineWidth',2)
71  hold on;
72  plot(t,dy,'r-.','LineWidth',1.5)
73  plot([0 max(t)],sqrt(vx)*[1 1],'m--',[0 max(t)],-sqrt(vx)*[1 1],'m--','LineWidth',1.5);
74  hold off
75  xlabel('Time');ylabel('y(t)');legend('cts','disc')
76  title('Stochastic Simulation of Helicopter Response: No FF')
77  subplot(212)
78  plot(t,u,'LineWidth',2)
79  xlabel('Time');ylabel('u(t)');legend('No FF')
80  hold on;
81  plot([0 max(t)],sqrt(vd)*[1 1],'m--',[0 max(t)],-sqrt(vd)*[1 1],'m--','LineWidth',1.5);
82  hold off
83
84  figure(3);
85  subplot(211)
86  plot(t,ya,'LineWidth',2)
87  hold on;
88  plot(t,dya,'r-.','LineWidth',1.5)
89  plot([0 max(t)],sqrt(vxa)*[1 1],'m--',[0 max(t)],-sqrt(vxa)*[1 1],'m--','LineWidth',1.5);
90  hold off
91  xlabel('Time');ylabel('y(t)');legend('cts','disc')
92  title('Stochastic Simulation of Helicopter Response: with FF')
93  subplot(212)
94  plot(t,ua,'LineWidth',2)
95  xlabel('Time');ylabel('u(t)');legend('with FF')
96  hold on;
97  plot([0 max(t)],sqrt(vda)*[1 1],'m--',[0 max(t)],-sqrt(vda)*[1 1],'m--','LineWidth',1.5);
98  hold off
99
100 print -f2 -r300 -dpng heli2.png
101 print -f3 -r300 -dpng heli3.png
```

LQG for Helicopter

- Now consider what happens if we reduce the measurable states and use LQG for the helicopter control/simulation
- Consider full vehicle state measurement (i.e., not the disturbance state)

$$C_y = \begin{bmatrix} I_4 & 0 \end{bmatrix}$$

- Consider only partial vehicle state measurement

$$C_y = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- Set $R_{\rm vv}$ small.

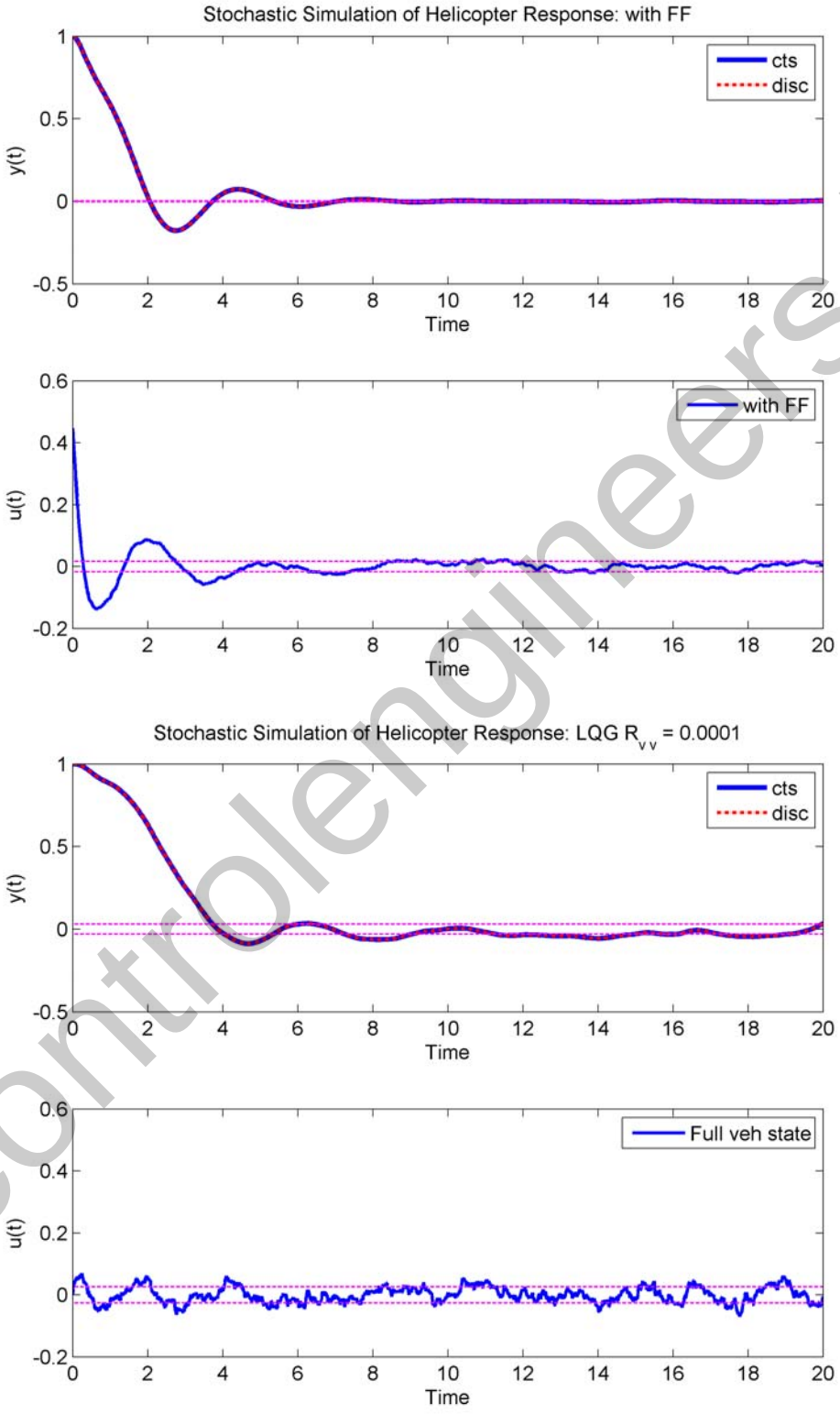


Figure 12.4: LQR with disturbance feedforward compared to LQG

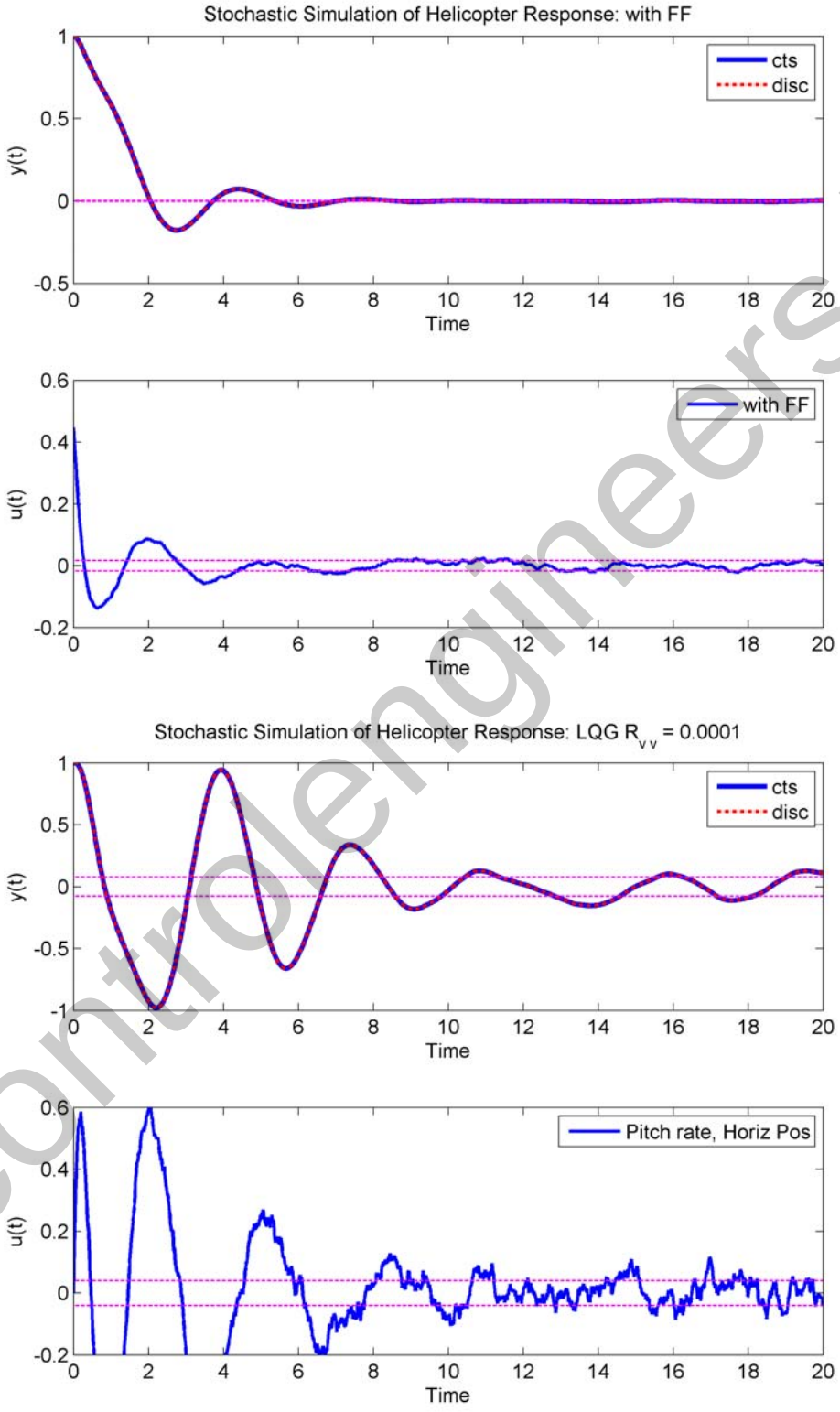


Figure 12.5: Second LQR with disturbance feedforward compared to LQG

Helicopter LQG

```
% 16.323 Spring 2008
% Stochastic Simulation of Helicopter LQR - from Bryson's Book
% Jon How
%
clear all, clf, randn('seed',sum(100*clock));
set(0,'DefaultAxesFontName','arial')
set(0,'DefaultAxesFontSize',12)
set(0,'DefaultTextFontName','arial')
% linearized dynamics of the system state=[theta q dotx x]
A = [ 0 1 0 0; 0 -0.415 -0.011 0;9.8 -1.43 -0.0198 0;0 0 1 0];
Bw = [0 -0.011 -0.0198 0]';
Bu = [0 6.27 9.8 0]';
Cz = [0 0 0 1];
Rxx = Cz'*Cz; Rww=1;
rho = 5;
% lqr control
[K,S,E]=lqr(A,Bu,Rxx,rho);

% initial response with given x0
x0 = [0 0 0 1]';
Ts=0.01; % small discrete step to simulate the cts dynamics
tf=20;t=0:Ts:tf;nt=length(t);
% Now consider shaped noise with shaping filter
Ah=-0.2;Bh=6;Ch=1;
% augment the filter dyanmics
Aa = [A Bw*Ch; zeros(1,4) Ah];
Bua = [Bu;0];
Bwa = [zeros(4,1); Bh];
Cza = [Cz 0];
x0a=[x0;0];
%zeta = Rww/sqrt(Ts)*randn(length(t),1); % discrete equivalent noise
zeta = sqrt(Rww/Ts)*randn(length(t),1); % discrete equivalent noise

%%% Now consider disturbance FF
[KK,SS,EE]=lqr(Aa,Bua,Cza'*Cza,rho); % now K will have dist FF
Acl=Aa-Bua*KK;
PP=lyap(Acl,Bwa*Rww*Bwa');
vxa = Cza*PP*Cza'; %state
vda = KK*PP*KK'; %control
%
[ya,xa] = lsim(Acl,Bwa,Cza,0,zeta,t,x0a); % cts sim
F=c2d(ss(Acl,Bwa,Cza,0),Ts); % discretize the closed-loop dynamics
[dya,dxa] = lsim(F,zeta,[],x0a); % stochastic sim in discrete time
ua = KK*xa'; % find control commands given the state response

%%% Now consider Output Feedback Case
% Assume that we can only measure the system states
% and not the dist one
FULL=1;
if FULL
  Cya=eye(4,5); % full veh state
else
    Cy=[0 1 0 0;0 0 0 1]; % only meas some states
    Cya=[Cy [0;0]];
end
Ncy=size(Cya,1);Rvv=(1e-2)^2*eye(Ncy);
[L,Q,FF]=lqr(Aa',Cya',Bwa*Rww*Bwa',Rvv);L=L';% LQE calc
%closed loop dyn
Acl_lqg=[Aa -Bua*KK;L*Cya Aa-Bua*KK-L*Cya];
Bcl_lqg=[Bwa zeros(5,Ncy);zeros(5,1) L];
Ccl_lqg=[Cza zeros(1,5)];Dcl_lqg=zeros(1,1+Ncy);
x0_lqg=[x0a;zeros(5,1)];
zeta_lqg=zeta;
% now just treat this as a system with more sensor noise acting as more
% process noise
for ii=1:Ncy
  zeta_lqg = [zeta_lqg sqrt(Rvv(ii,ii)/Ts)*randn(nt,1)];% discrete equivalent noise
```

```
68  end
69  [ya_lqg,xa_lqg] = lsim(Acl_lqg,Bcl_lqg,Ccl_lqg,Dcl_lqg,zeta_lqg,t,x0_lqg); % cts sim
70  F_lqg=c2d(ss(Acl_lqg,Bcl_lqg,Ccl_lqg,Dcl_lqg),Ts); % discretize the closed-loop dynamics
71  [dya_lqg,dxa_lqg] = lsim(F_lqg,zeta_lqg,[],x0_lqg); % stochastic sim in discrete time
72  ua_lqg = [zeros(1,5) KK]*xa_lqg'; % find control commands given the state estimate
73
74  %LQG State Perf Prediction
75  X_lqg=lyap(Acl_lqg,Bcl_lqg*[Rww zeros(1,Ncy);zeros(Ncy,1) Rvv]*Bcl_lqg');
76  vx_lqg=Ccl_lqg*X_lqg*Ccl_lqg';
77  vu_lqg=[zeros(1,5) KK]*X_lqg*[zeros(1,5) KK]';
78
79  figure(3);clf
80  subplot(211)
81  plot(t,ya,'LineWidth',3)
82  hold on;
83  plot(t,dya,'r-.','LineWidth',2)
84  plot([0 max(t)],sqrt(vxa)*[1 1],'m--',[0 max(t)],-sqrt(vxa)*[1 1],'m--','LineWidth',1);
85  hold off
86  xlabel('Time');ylabel('y(t)');legend('cts','disc')
87  title('Stochastic Simulation of Helicopter Response: with FF')
88  subplot(212)
89  plot(t,ua,'LineWidth',2)
90  xlabel('Time');ylabel('u(t)');legend('with FF')
91  hold on;
92  plot([0 max(t)],sqrt(vda)*[1 1],'m--',[0 max(t)],-sqrt(vda)*[1 1],'m--','LineWidth',1);
93  axis([0 tf -0.2 .6])
94  hold off
95  print -f3 -r300 -dpng  heli_lqg_1.png;
96
97  figure(4);clf
98  subplot(211)
99  plot(t,ya_lqg,'LineWidth',3)
100 hold on;
101 plot(t,dya_lqg,'r-.','LineWidth',2)
102 plot([0 max(t)],sqrt(vx_lqg)*[1 1],'m--',[0 max(t)],-sqrt(vx_lqg)*[1 1],'m--','LineWidth',1);
103 hold off
104 xlabel('Time');ylabel('y(t)');legend('cts','disc')
105 title(['Stochastic Simulation of Helicopter Response: LQG R_{v v} = ',num2str(Rvv(1,1))])
106 subplot(212)
107 plot(t,ua_lqg,'LineWidth',2)
108 xlabel('Time');ylabel('u(t)');%legend('with FF')
109 if FULL
110     legend('Full veh state')
111 else
112     legend('Pitch rate, Horiz Pos')
113 end
114 hold on;
115 plot([0 max(t)],sqrt(vu_lqg)*[1 1],'m--',[0 max(t)],-sqrt(vu_lqg)*[1 1],'m--','LineWidth',1);
116 axis([0 tf -0.2 .6])
117 hold off
118 if FULL
119     print -f4 -r300 -dpng  heli_lqg_2.png;
120 else
121     print -f4 -r300 -dpng  heli_lqg_3.png;
122 end
```

Example: 747

- **Bryson, page 209** Consider the stabilization of a 747 at 40,000 ft and Mach number of 0.80. The perturbation dynamics from elevator angle to pitch angle are given by

$$\frac{\theta(s)}{\delta_e(s)} = G(s) = \frac{1.16(s+0.0113)(s+0.295)}{[s^2+(0.0676)^2][(s+0.375)^2+(0.882)^2]}$$

1. Note that these aircraft dynamics can be stabilized with a simple lead compensator

$$\frac{\delta_e(s)}{\theta(s)} = 3.50\frac{s+0.6}{s+3.6}$$

2. Can also design an LQG controller for this system by assuming that $B_w = B_u$ and $C_z = C_y$, and then tuning $R_{\rm uu}$ and $R_{\rm vv}$ to get a reasonably balanced performance.
   – Took $R_{\rm ww} = 0.1$ and tuned $R_{\rm vv}$

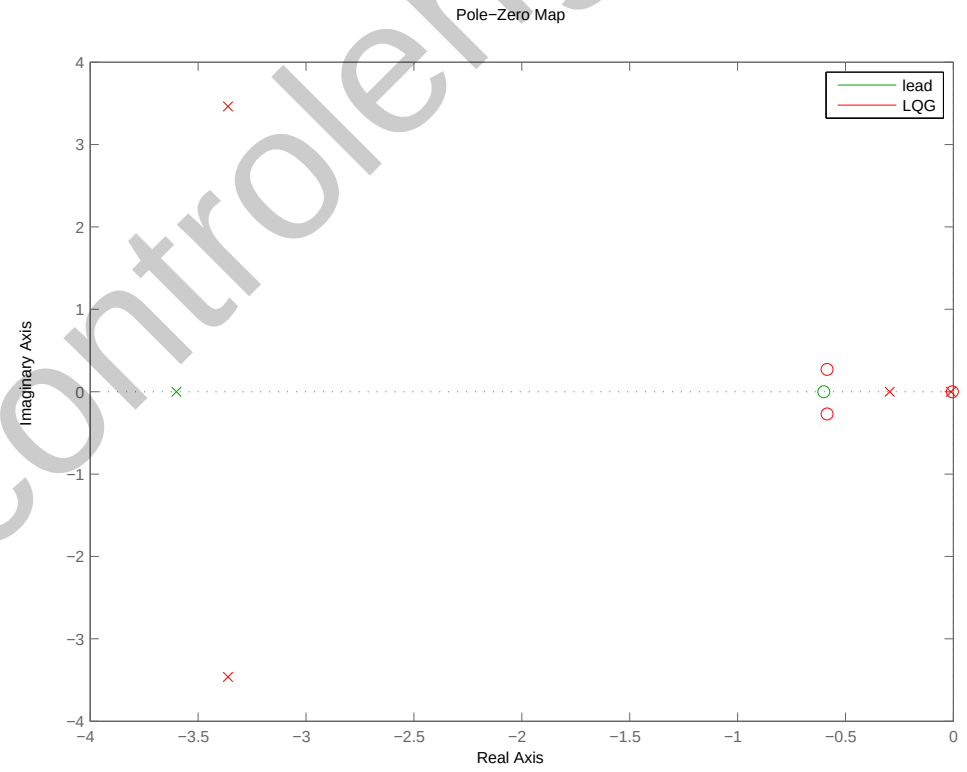


Figure 12.6: B747: Compensators

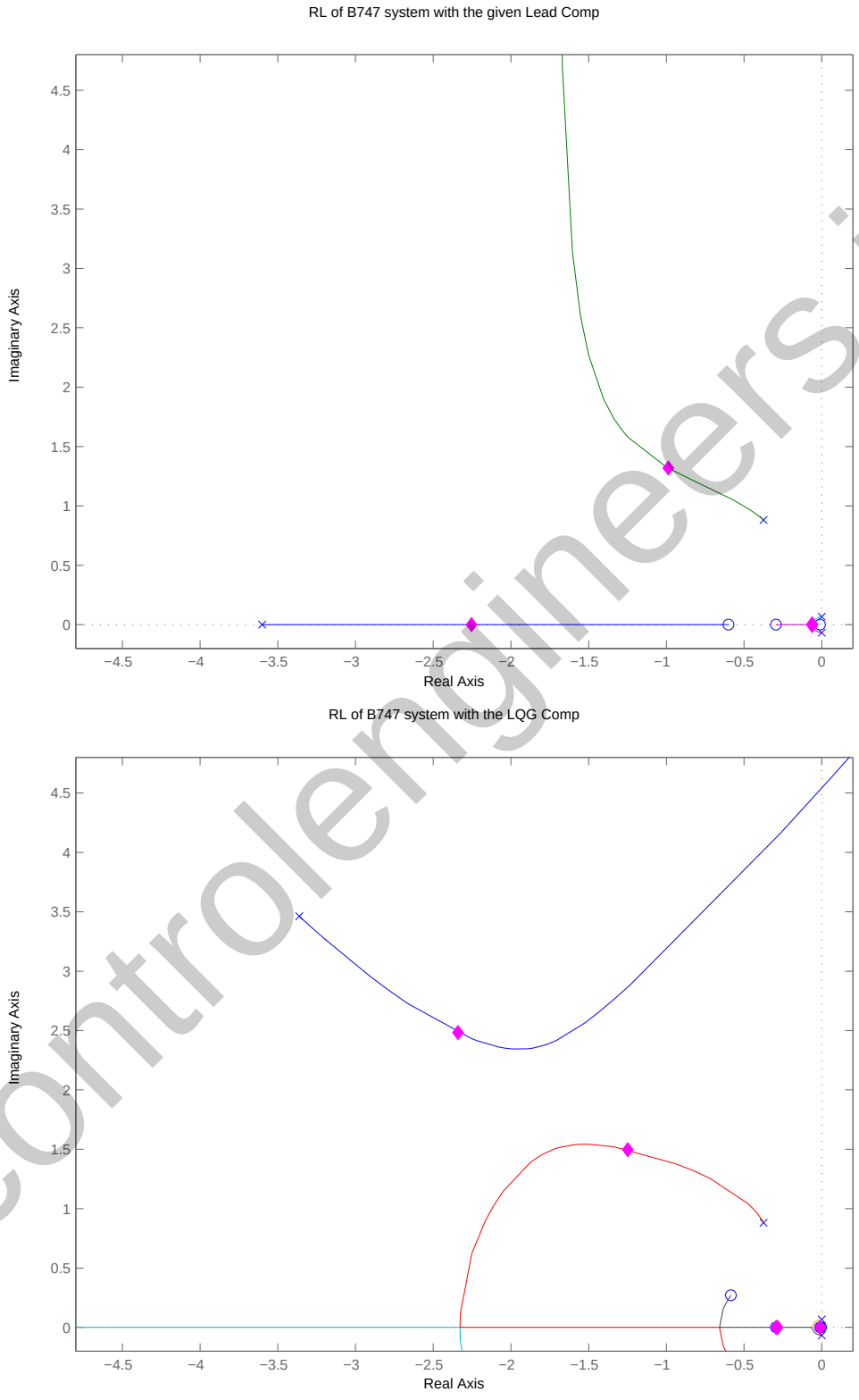


Figure 12.7: B747: root locus (Lead on left, LQG on right shown as a function of the overall compensator gain)

3. Compare the Bode plots of the lead compensator and LQG designs

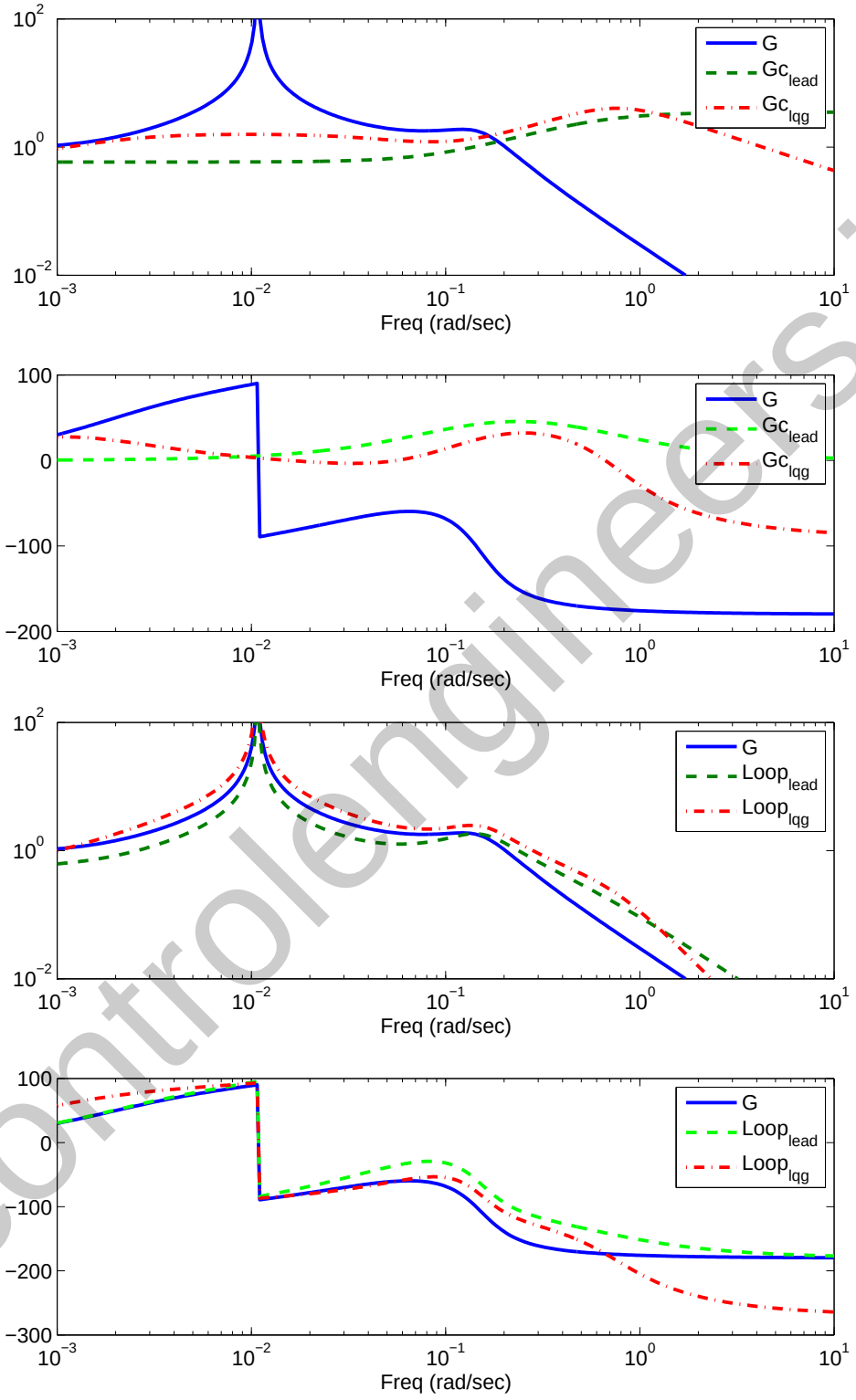


Figure 12.8: B747: Compensators and loop TF

4. Consider the closed-loop TF for the system

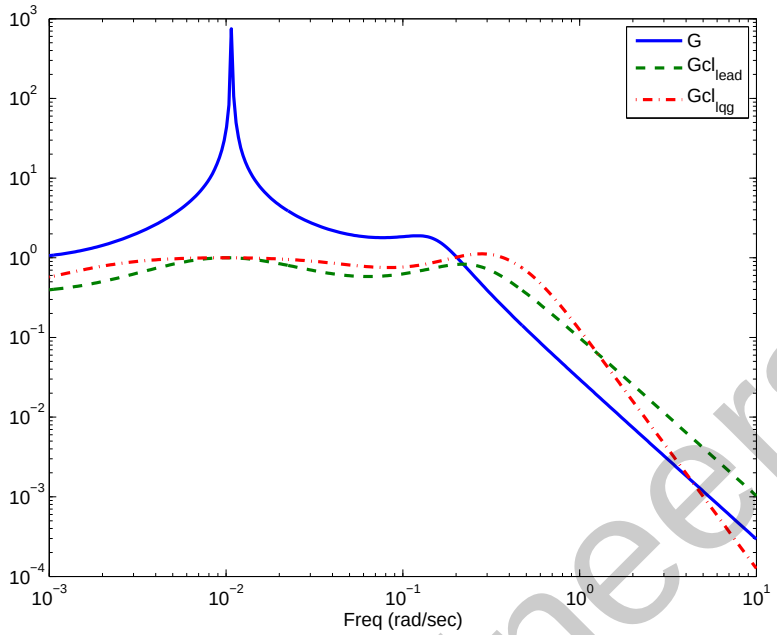


Figure 12.9: B747: closed-loop TF

5. Compare impulse response of two closed-loop systems.

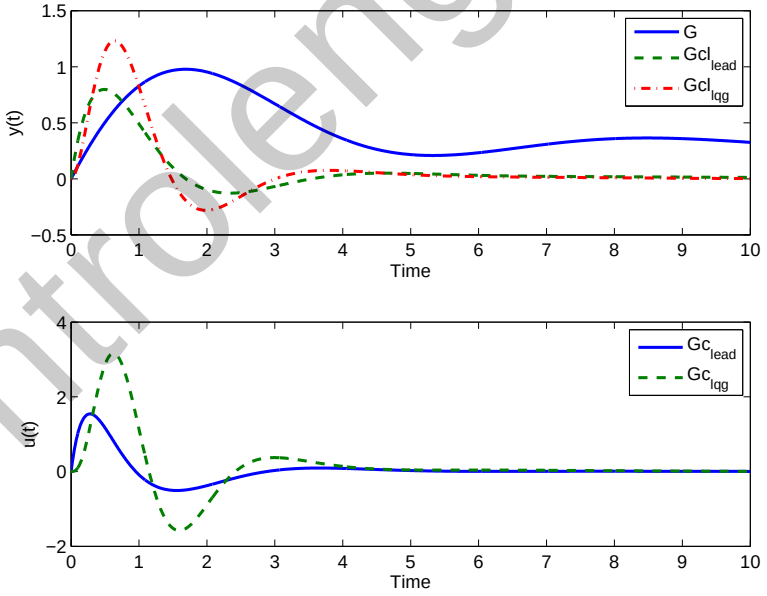


Figure 12.10: B747: Impulse response

6. So while LQG controllers might appear to be glamorous, they are actually quite ordinary for SISO systems.
   – Where they really shine is that it this simple to design a MIMO controller.

B747 LQG

```
% 16.323 B747 example
% Jon How, MIT, Spring 2007
%
clear all
set(0,'DefaultAxesFontName','arial')
set(0,'DefaultAxesFontSize',12)
set(0,'DefaultTextFontName','arial')

gn=1.16*conv([1 .0113],[1 .295]);
gd=conv([1 0 .0676^2],[1 2*.375 .375^2+.882^2]);
% lead comp given
kn=3.5*[1 .6];kd=[1 3.6];

f=logspace(-3,1,300);
g=freqresp(gn,gd,2*pi*f*sqrt(-1));

[nc,dc]=cloop(conv(gn,kn),conv(gd,kd)); % CLP with lead
gc=freqresp(nc,dc,2*pi*f*sqrt(-1)); % CLP with lead
%roots(dc)
%loglog(f,abs([g gc]))

%get state space model
[a,b,c,d]=tf2ss(gn,gd);
% assume that Bu and Bw are the same
% take y=z
Rzz=1;Ruu=0.01;Rww=0.1;Rvv=0.01;
[k,P,e1] = lqr(a,b,c'*Rzz*c,Ruu);
[l,Q,e2] = lqe(a,b,c,Rww,Rvv);
[ac,bc,cc,tdc] = reg(a,b,c,d,k,l);
[knl,kdl]=ss2tf(ac,bc,cc,tdc);
N1=trace(P*(l*Rvv*l'))%
N2=trace(Q*(c'*Rzz*c))%
N3=trace(P*(b*Rww*b'))%
N4=trace(Q*(k'*Ruu*k))%
N=[N1 N2 N1+N2;N3 N4 N3+N4]

[ncl,dcl]=cloop(conv(gn,knl),conv(gd,kdl)); % CLP with lqg
gcl=freqresp(ncl,dcl,2*pi*f*sqrt(-1)); % CLP with lqg
[[roots(dc);0;0;0] roots(dcl)]
figure(2);clf;
loglog(f,abs([g gc gcl])) % mag plot of closed loop system
setlines(2)
legend('G','Gcl_{lead}','Gcl_{lqg}')
xlabel('Freq (rad/sec)')

Gclead=freqresp(kn,kd,2*pi*f*sqrt(-1));
Gclqg=freqresp(knl,kdl,2*pi*f*sqrt(-1));

figure(3);clf;
subplot(211)
loglog(f,abs([g Gclead Gclqg])) % Bode of compesantors
setlines(2)
legend('G','Gc_{lead}','Gc_{lqg}')
xlabel('Freq (rad/sec)')
axis([1e-3 10 1e-2 1e2])
subplot(212)
semilogx(f,180/pi*unwrap(phase([g])));hold on
semilogx(f,180/pi*unwrap(phase([Gclead])),'g')
semilogx(f,180/pi*unwrap(phase([Gclqg])),'r')
xlabel('Freq (rad/sec)')
hold off
setlines(2)
legend('G','Gc_{lead}','Gc_{lqg}')

figure(6);clf;
subplot(211)
loglog(f,abs([g g.*Gclead g.*Gclqg])) % Bode of Loop transfer function
```

```
setlines(2)
legend('G','Loop_{lead}','Loop_{lqg}')
xlabel('Freq (rad/sec)')
axis([1e-3 10 1e-2 1e2])
subplot(212)
semilogx(f,180/pi*unwrap(phase([g])));hold on
semilogx(f,180/pi*unwrap(phase([g.*Gclead])),'g')
semilogx(f,180/pi*unwrap(phase([g.*Gclqg])),'r')
xlabel('Freq (rad/sec)')
hold off
setlines(2)
legend('G','Loop_{lead}','Loop_{lqg}')

% RL of 2 closed-loop systems
figure(1);clf;rlocus(conv(gn,kn),conv(gd,kd));axis(2*[-2.4 0.1 -0.1 2.4])
hold on;plot(roots(dc)+sqrt(-1)*eps,'md','MarkerFaceColor','m');hold off
title('RL of B747 system with the given Lead Comp')
figure(4);clf;rlocus(conv(gn,knl),conv(gd,kdl));axis(2*[-2.4 0.1 -0.1 2.4])
hold on;plot(roots(dcl)+sqrt(-1)*eps,'md','MarkerFaceColor','m');hold off
title('RL of B747 system with the LQG Comp')

% time simulations
Ts=0.01;
[y1,x,t]=impulse(gn,gd,[0:Ts:10]);
[y2]=impulse(nc,dc,t);
[y3]=impulse(ncl,dcl,t);
[ulead]=lsim(kn,kd,y2,t); % noise free sim
[ulqg]=lsim(knl,kdl,y3,t); % noise free sim

figure(5);clf;
subplot(211)
plot(t,[y1 y2 y3])
xlabel('Time')
ylabel('y(t)')
setlines(2)
legend('G','Gcl_{lead}','Gcl_{lqg}')
subplot(212)
plot(t,[ulead ulqg])
xlabel('Time')
ylabel('u(t)')
setlines(2)
legend('Gc_{lead}','Gc_{lqg}')

figure(7)
pzmap(tf(kn,kd),'g',tf(knl,kdl),'r')
legend('lead','LQG')

print -depsc -f1 b747_1.eps;jpdf('b747_1')
print -depsc -f2 b747_2.eps;jpdf('b747_2')
print -depsc -f3 b747_3.eps;jpdf('b747_3')
print -depsc -f4 b747_4.eps;jpdf('b747_4')
print -depsc -f5 b747_5.eps;jpdf('b747_5')
print -depsc -f6 b747_6.eps;jpdf('b747_6')
print -depsc -f7 b747_7.eps;jpdf('b747_7')

```

16.323 Lecture 13

LQG Robustness

- Stengel Chapter 6
- Question: how well do the large gain and phase margins discussed for LQR (6–29) map over to LQG?

LQG

- When we use the combination of an optimal estimator and an optimal regulator to design the controller, the compensator is called

  Linear Quadratic Gaussian (LQG)

  – Special case of the controllers that can be designed using the separation principle.

- The great news about an LQG design is that stability of the closed-loop system is **guaranteed**.
  – The designer is freed from having to perform any detailed mechanics - the entire process is fast and can be automated.

- So the designer can focus on the "performance" related issues, being confident that the LQG design will produce a controller that stabilizes the system.
  – How to specify the state cost function (i.e. selecting $\mathbf{z} = C_z\mathbf{x}$) and what values of $R_{\rm zz}$, $R_{\rm uu}$ to use.
  – Determine how the process and sensor noise enter into the system and what their relative sizes are (i.e. select $R_{\rm ww}$ & $R_{\rm vv}$)

- This sounds great – so what is the catch??

- The remaining issue is that sometimes the controllers designed using these state-space tools are very sensitive to errors in the knowledge of the model.
  – *i.e.,* the compensator might work **very well** if the plant gain $\alpha = 1$, but be unstable if it is $\alpha = 0.9$ or $\alpha = 1.1$.
  – LQG is also prone to plant–pole/compensator–zero cancelation, which tends to be sensitive to modeling errors.
  – J. Doyle, "Guaranteed Margins for LQG Regulators", IEEE *Transactions on Automatic Control*, Vol. 23, No. 4, pp. 756-757, 1978.

- The good news is that the state-space techniques will give you a controller very easily.
  - **You should use the time saved to verify that the one you designed is a "good" controller.**

- There are, of course, different definitions of what makes a controller **good**, but one important criterion is whether **there is a reasonable chance that it would work on the real system as well as it does in Matlab.** $\Rightarrow$ **Robustness**.
  - The controller must be able to tolerate some modeling error, because our models in Matlab are typically inaccurate.
    - ◇ Linearized model
    - ◇ Some parameters poorly known
    - ◇ Ignores some higher frequency dynamics

- Need to develop tools that will give us some insight on how well a controller can tolerate modeling errors.

LQG Example

- Consider the "cart on a stick" system, with the dynamics as given in the following pages. Define

$$q = \begin{bmatrix} \theta \\ x \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

Then with $y = x$

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B_u u \\ y &= C_y \mathbf{x} \end{aligned}$$

- For the parameters given in the notes, the system has an unstable pole at $+5.6$ and one at $s = 0$. There are plant zeros at ± 5.

- Very simple LQG design - main result is fairly independent of the choice of the weighting matrices.

- The resulting compensator is unstable ($+23$!!)
  – This is somewhat expected. (why?)

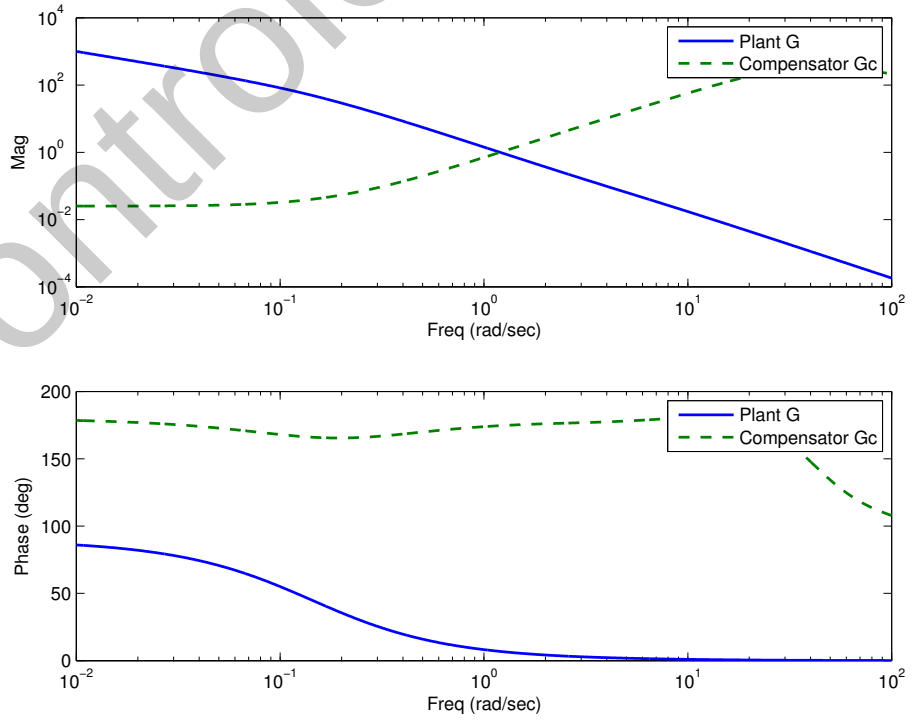


Figure 13.1: Plant and Controller

Example: cart with an inverted pendulum.

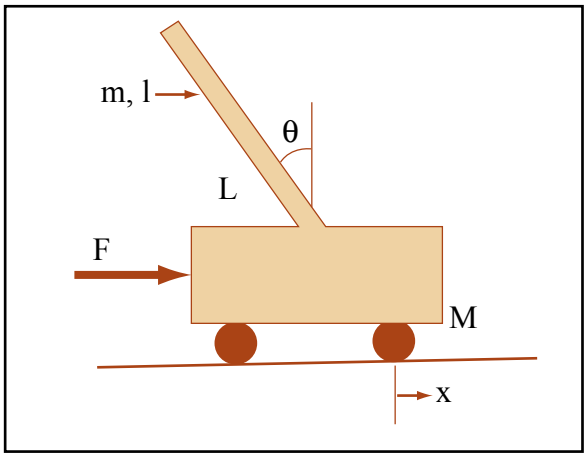


Figure by MIT OpenCourseWare.

- Nonlinear equations of motion can be developed for large angle motion (see 30-32)
- Force actuator, θ sensor

Linearize for small θ

$$(I+mL^2)\ddot{\theta} - mgL\,\theta = mL\,\ddot{x}$$

$$(M+m)\ddot{x} + g\,\dot{x} - mL\,\ddot{\theta} = F$$

$$\begin{bmatrix} (I+mL^2)s^2 - mgL & -mLs^2 \\ -mLs^2 & (M+m)s^2 + Gs \end{bmatrix} \begin{bmatrix} \Theta(s) \\ x(s) \end{bmatrix} = \begin{bmatrix} 0 \\ F(s) \end{bmatrix}$$

$$\frac{\Theta}{F} = \frac{mLs^2}{\left[(I+mL^2)s^2 - mgL\right]\left[(M+m)s^2 + Gs\right] - (mLs^2)^2}$$

Cannot say too much more

Let M= 0.5, m=0.2, G=0.1, I=0.006, L=0.3

➔ gives $\dfrac{\Theta}{F} = \dfrac{4.54s^2}{s^4 + 0.1818s^3 - 31.18s^2 - 4.45s}$

therefore has an unstable pole (as expected)
s= ± 5.6,-0.14,0

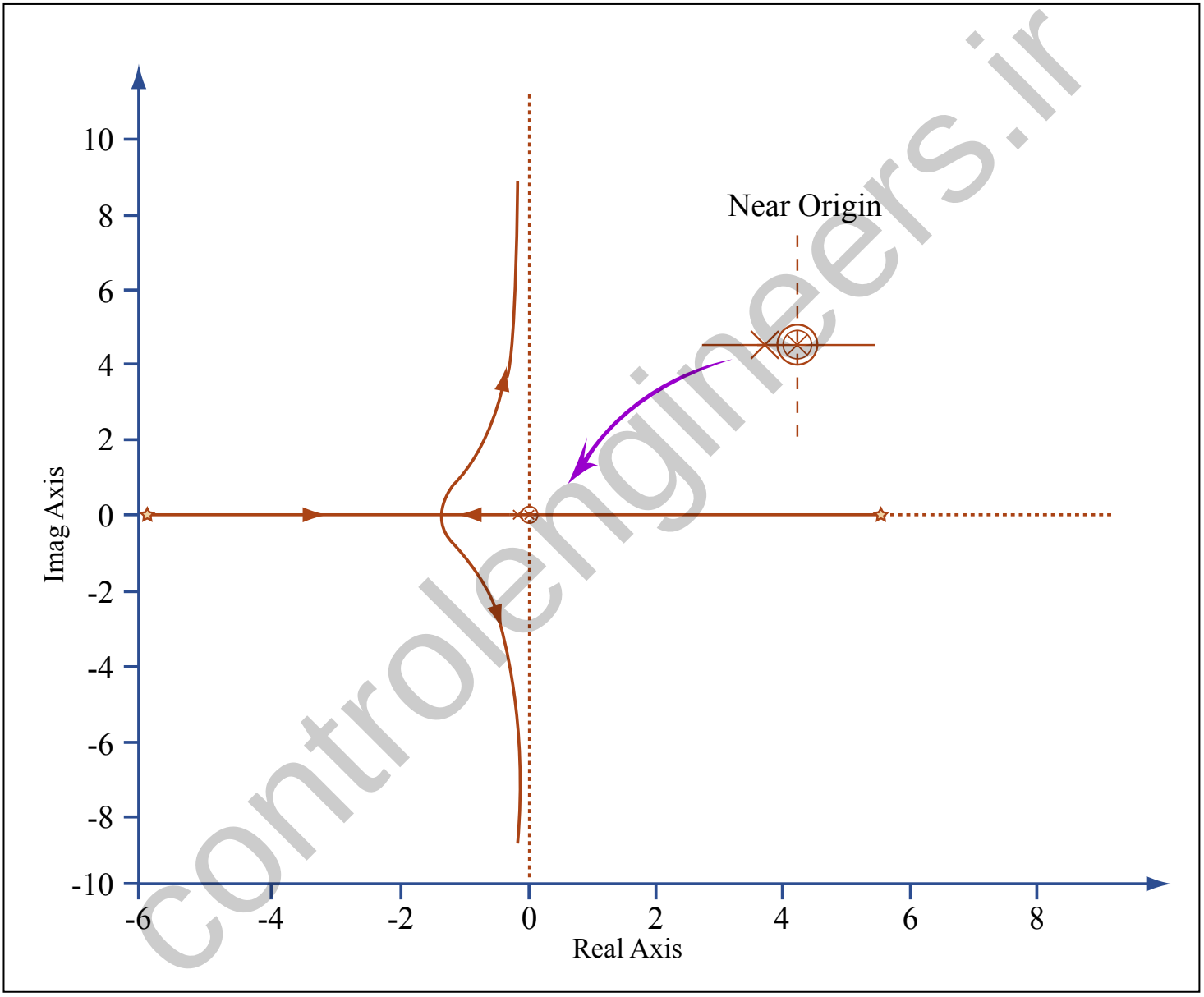


Figure by MIT OpenCourseWare.

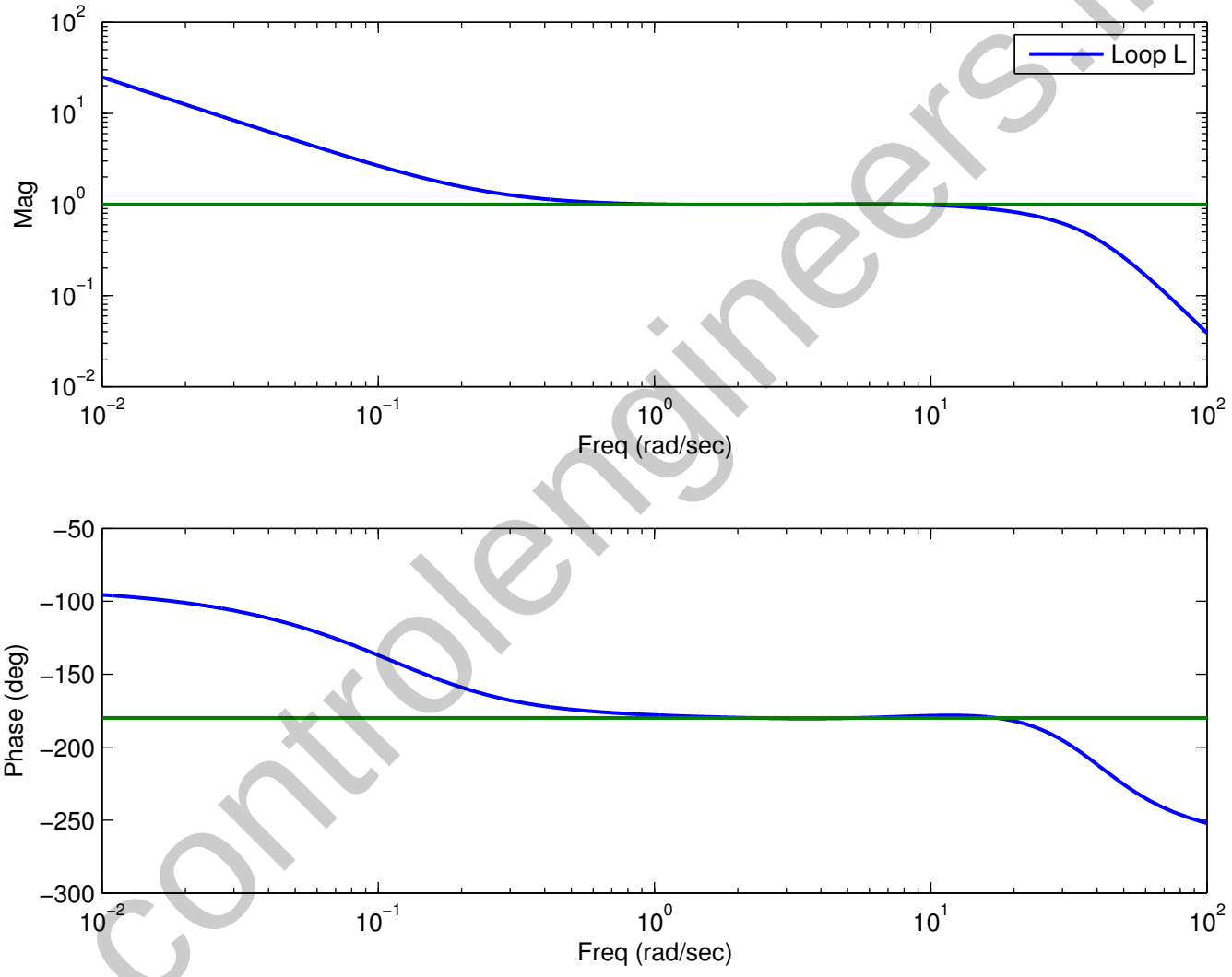


Figure 13.2: Loop and Margins

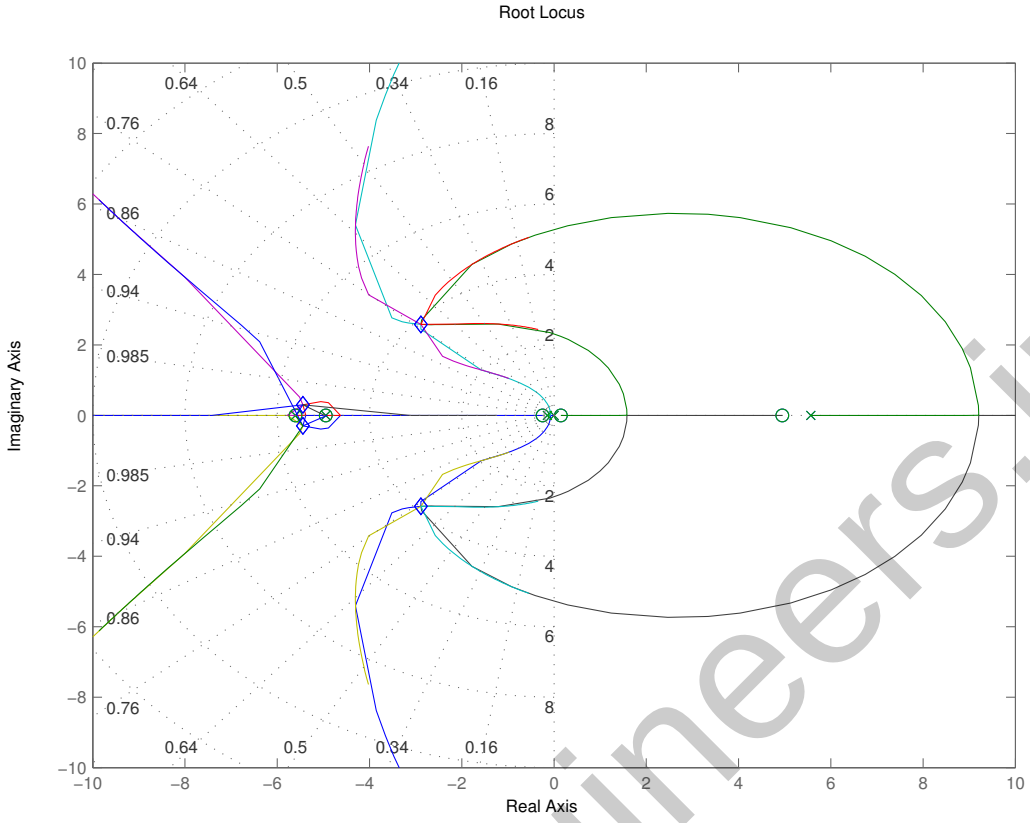
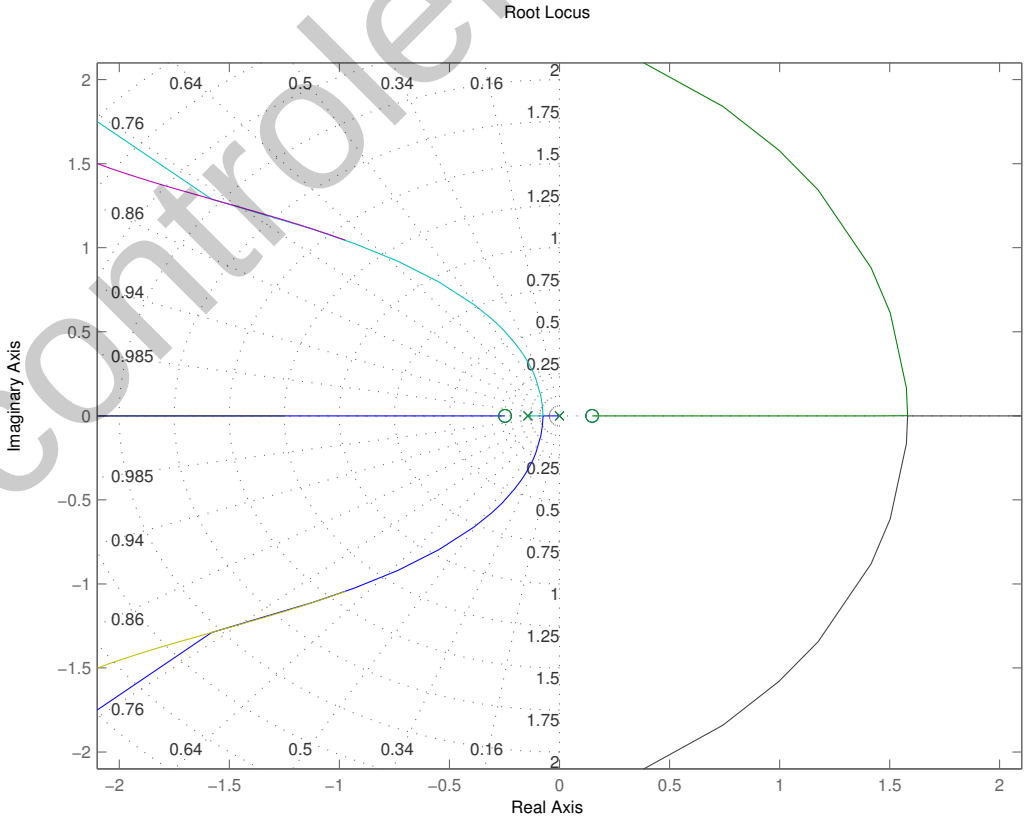


Figure 13.3: Root Locus with frozen compensator dynamics. Shows sensitivity to overall gain – symbols are a gain of [0.995:.0001:1.005].



- Looking at both the Loop TF plots and the root locus, it is clear this system is stable with a gain of 1, but
  - Unstable for a gain of $1 \pm \epsilon$ and/or a slight change in the system phase (possibly due to some unmodeled delays)
  - Very limited chance that this would work on the real system.

- Of course, this is an extreme example and not all systems are like this, but you must analyze to determine what **robustness margins** your controller really has.

- **Question:** what analysis tools should we use?

Analysis Tools to Use?

- Eigenvalues give a definite answer on the stability (or not) of the closed-loop system.
  - Problem is that it is very hard to predict where the closed-loop poles will go as a function of errors in the plant model.

- Consider the case were the model of the system is

$$\dot{x} = A_0 x + Bu$$

  - Controller also based on A_0, so **nominal** closed-loop dynamics:

$$\begin{bmatrix} A_0 & -BK \\ LC & A_0 - BK - LC \end{bmatrix} \Rightarrow \begin{bmatrix} A_0 - BK & BK \\ 0 & A_0 - LC \end{bmatrix}$$

- But what if the **actual** system has dynamics

$$\dot{x} = (A_0 + \Delta A)x + Bu$$

  - Then **perturbed** closed-loop system dynamics are:

$$\begin{bmatrix} A_0 + \Delta A & -BK \\ LC & A_0 - BK - LC \end{bmatrix} \Rightarrow \begin{bmatrix} A_0 + \Delta A - BK & BK \\ \Delta A & A_0 - LC \end{bmatrix}$$

- Transformed $\bar{A}_{cl}$ not upper-block triangular, so perturbed closed-loop eigenvalues are **NOT** the union of regulator & estimator poles.
  - Can find the closed-loop poles for a specific ΔA, but
  - Hard to predict change in location of closed-loop poles for a range of possible modeling errors.

Frequency Domain Tests

- Frequency domain stability tests provide further insights on the **stability margins**.

- Recall from the **Nyquist Stability Theorem:**
  - If the loop transfer function $L(s)$ has P poles in the RHP s-plane (and $\lim_{s\to\infty} L(s)$ is a constant), then for closed-loop stability, the locus of $L(\mathrm{j}\omega)$ for $\omega \in (-\infty, \infty)$ must encircle the critical point $(-1, 0)$ P times in the **counterclockwise** direction [Ogata 528].
  - This provides a binary measure of stability, or not.

- Can use "closeness" of $L(s)$ to the critical point as a measure of "closeness" to changing the number of encirclements.
  - Premise is that the system is stable for the nominal system
    $\Rightarrow$ has the right number of encirclements.

- Goal of the robustness test is to see if the possible perturbations to our system model (due to modeling errors) can **change the number of encirclements**
  - In this case, say that the perturbations can **destabilize** the system.

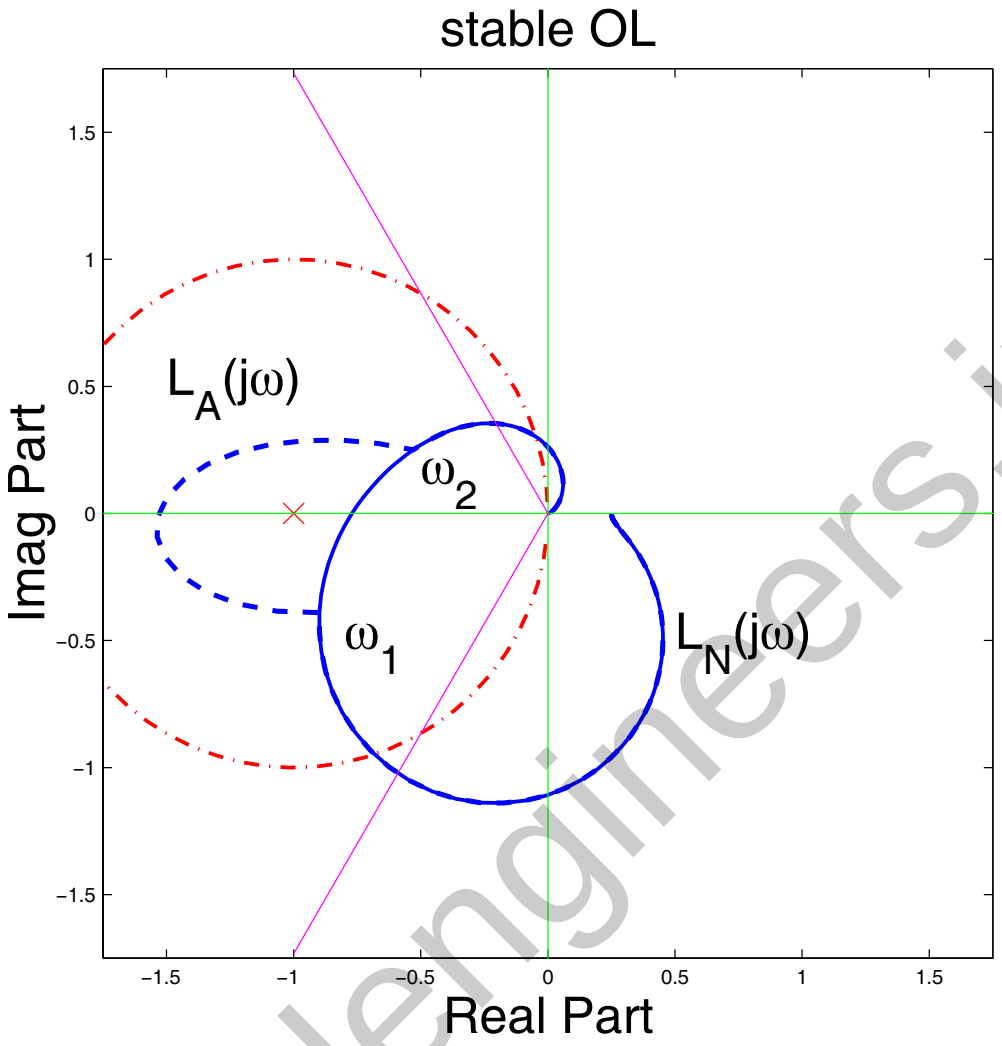


Figure 13.4: Plot of Loop TF $L_N(\mathrm{j}\omega) = G_N(\mathrm{j}\omega)G_c(\mathrm{j}\omega)$ and perturbation $(\omega_1 \to \omega_2)$ that changes the number of encirclements.

- Model error in frequency range $\omega_1 \leq \omega \leq \omega_2$ causes a change in the number of encirclements of the critical point $(-1, 0)$
  - **Nominal** closed-loop system stable $L_N(s) = G_N(s)G_c(s)$
  - **Actual** closed-loop system unstable $L_A(s) = G_A(s)G_c(s)$

- **Bottom line:** Large model errors when $L_N \approx -1$ are very dangerous.

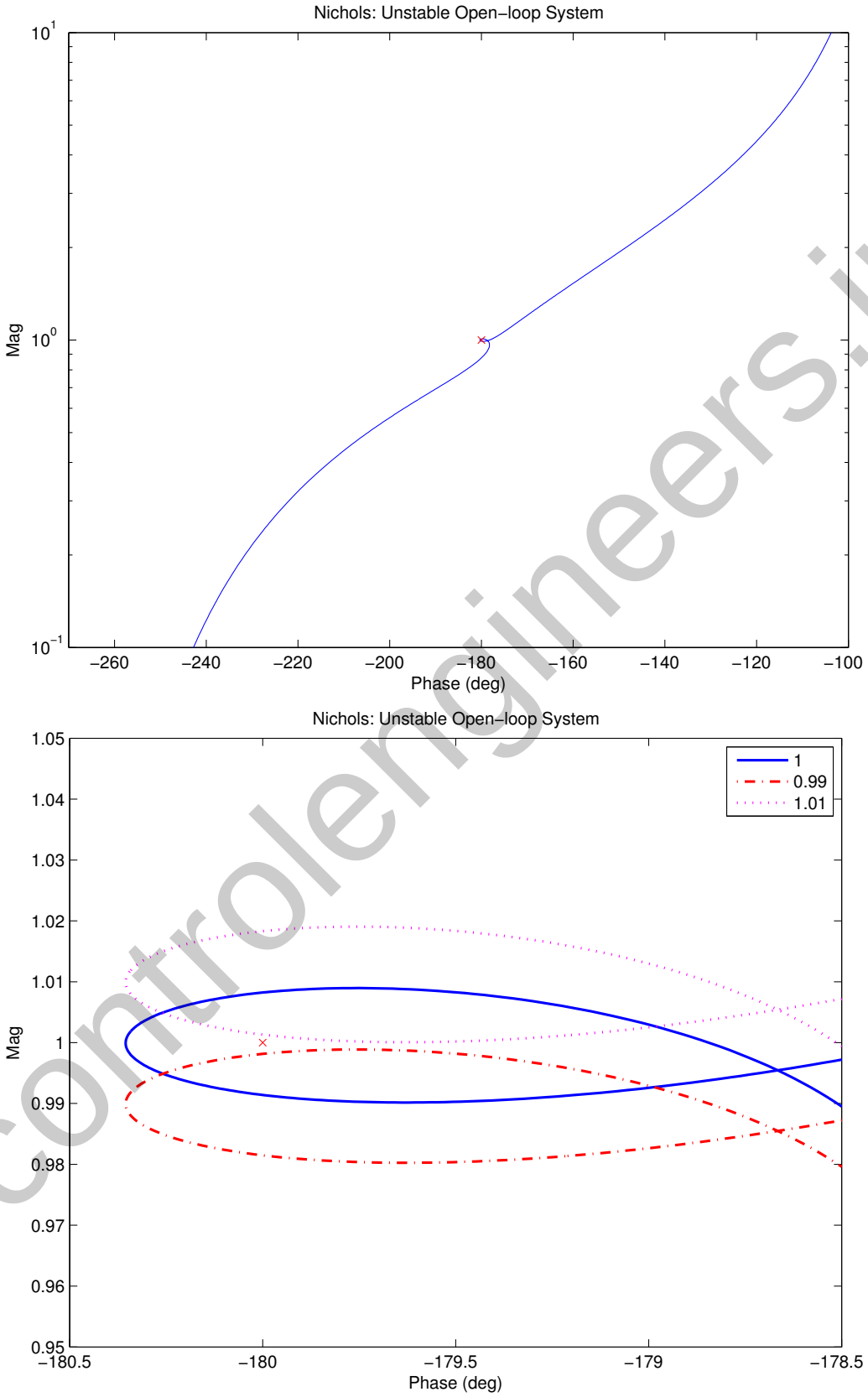


Figure 13.5: Nichols Plot ($|L((\mathrm{j}\omega))|$ vs. $\arg L((\mathrm{j}\omega))$) for the cart example which clearly shows the sensitivity to the overall gain and/or phase lag.

Frequency Domain Test

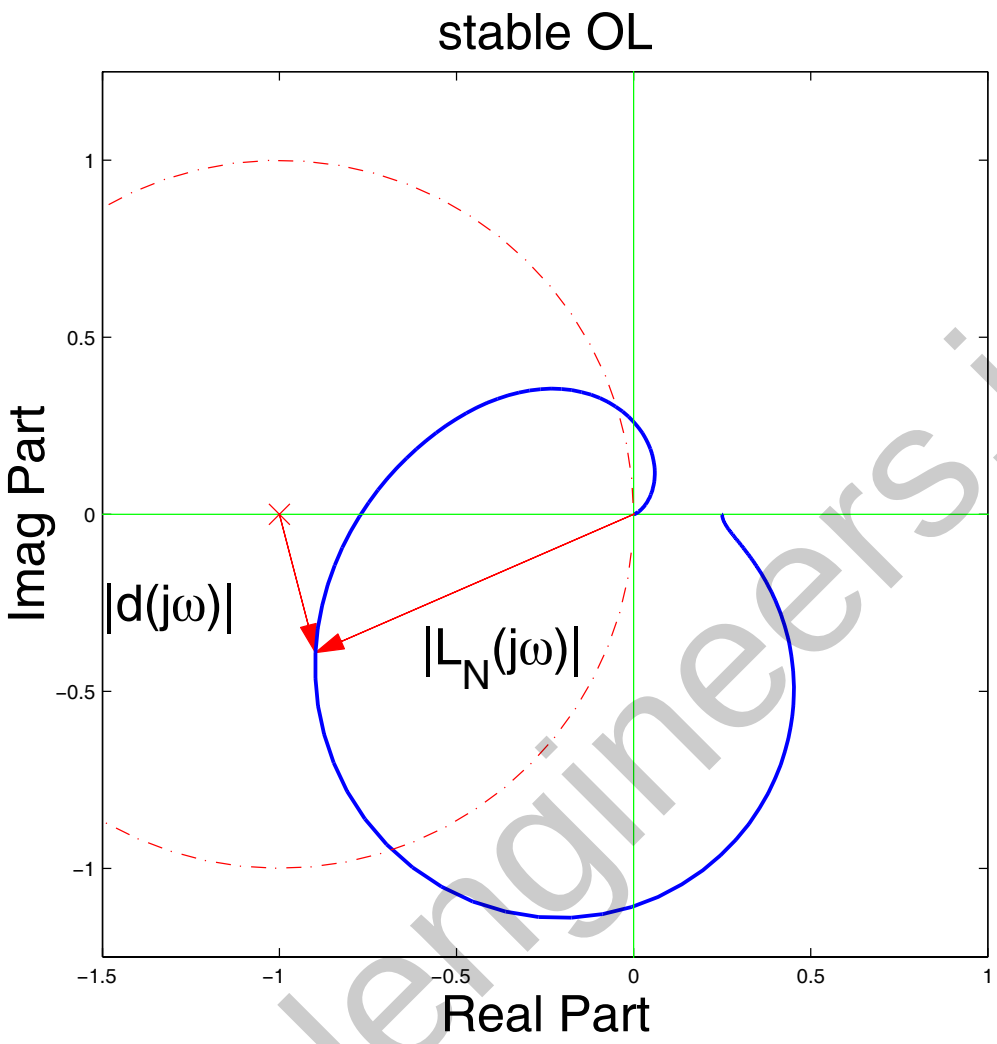


Figure 13.6: Geometric interpretation from Nyquist Plot of Loop TF.

- $|d(\mathrm{j}\omega)|$ measures distance of nominal Nyquist locus to critical point.

- But vector addition gives $$-1 + d(\mathrm{j}\omega) = L_N(\mathrm{j}\omega)$$

$$\Rightarrow \quad d(\mathrm{j}\omega) = 1 + L_N(\mathrm{j}\omega)$$

- Actually more convenient to plot

$$\frac{1}{|d(\mathrm{j}\omega)|} = \frac{1}{|1 + L_N(\mathrm{j}\omega)|} \triangleq |S(\mathrm{j}\omega)|$$

the magnitude of the **sensitivity transfer function** $S(s)$.

- So high sensitivity corresponds to $L_N(\mathrm{j}\omega)$ being **very** close to the critical point.

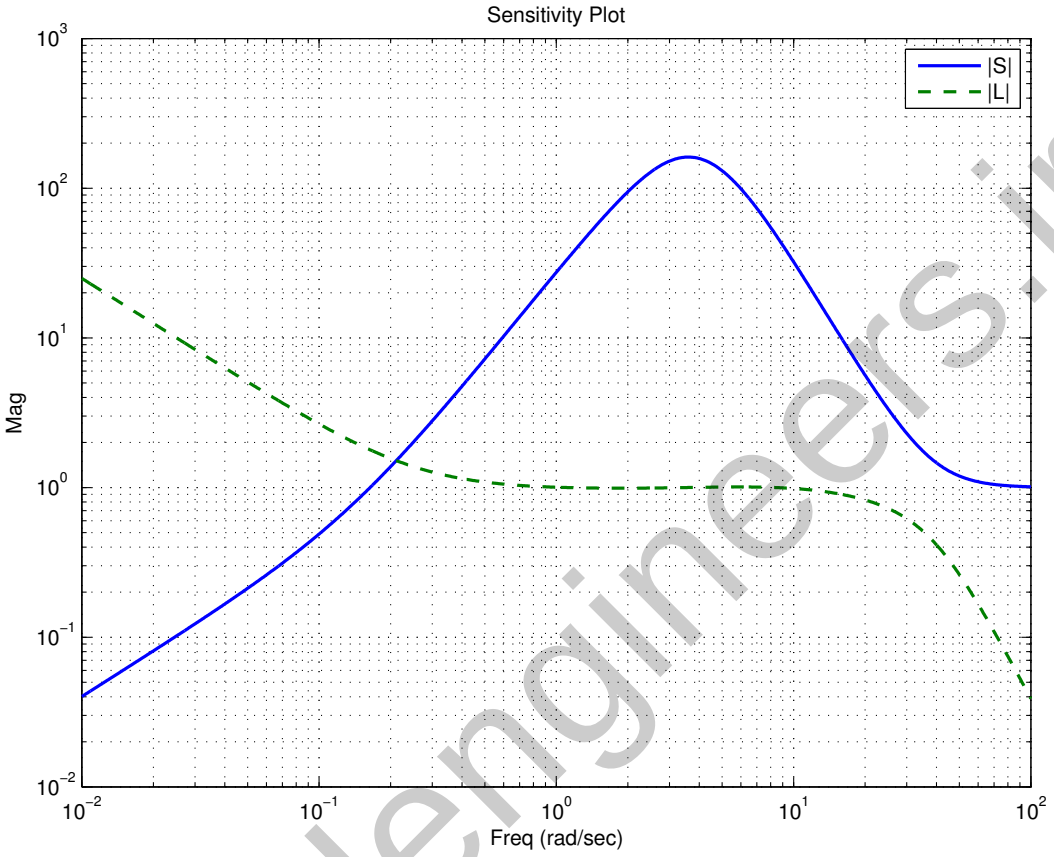


Figure 13.7: Sensitivity plot of the cart problem.

- Ideally you would want the sensitivity to be much lower than this.
  - Same as saying that you want $L(\mathrm{j}\omega)$ to be far from the critical point.
  - Difficulty in this example is that the open-loop system is unstable, so $L(\mathrm{j}\omega)$ must encircle the critical point $\Rightarrow$ hard for $L(\mathrm{j}\omega)$ to get too far away from the critical point.

Summary

- LQG gives you a great way to design a controller for the nominal system.

- But there are no guarantees about the stability/performance if the actual system is slightly different.
  - Basic analysis tool is the **Sensitivity Plot**

- No obvious ways to tailor the specification of the LQG controller to improve any lack of robustness
  - Apart from the obvious "lower the controller bandwidth" approach.
  - And sometimes you need the bandwidth just to stabilize the system.

- Very hard to include additional robustness constraints into LQG
  - See my Ph.D. thesis in 1992.

- Other tools have been developed that allow you to **directly** shape the sensitivity plot $|S(\mathrm{j}\omega)|$
  - Called $\mathcal{H}_\infty$ and μ

- **Good news:** Lack of robustness is something you should look for, but it is not always an issue.

Topic #14

16.31 Feedback Control Systems

MIMO Systems

- Singular Value Decomposition

Multivariable Frequency Response

- In the MIMO case, the system $G(s)$ is described by a $p \times m$ **transfer function matrix** (TFM)
  - Still have that $G(s) = C(sI - A)^{-1}B + D$
  - But $G(s) \rightarrow A, B, C, D$ MUCH less obvious than in SISO case.
  - Also seen that the discussion of poles and zeros of MIMO systems is much more complicated.

- In SISO case we use the Bode plot to develop a measure of the system "size".
  - Given $z = Gw$, where $G(\mathrm{j}\omega) = |G(\mathrm{j}\omega)|e^{j\phi(w)}$
  - Then $w = |w|e^{j(\omega_1 t + \psi)}$ applied to $|G(\mathrm{j}\omega)|e^{j\phi(w)}$ yields

$$|w||G(\mathrm{j}\omega_1)|e^{j(\omega_1 t + \psi + \phi(\omega_1))} = |z|e^{j(\omega_1 t + \psi_o)} \equiv z$$

  - Amplification and phase shift of the input signal obvious in the SISO case.

- MIMO extension?
  - Is the response of the system large or small?

$$G(s) = \begin{bmatrix} 10^3/s & 0 \\ 0 & 10^{-3}/s \end{bmatrix}$$

- For MIMO systems, cannot just plot all of the g_{ij} elements of G
  - Ignores the coupling that might exist between them.
  - So not enlightening.

- **Basic MIMO frequency response:**
  - Restrict all inputs to be at the same frequency
  - Determine how the system responds at that frequency
  - See how this response changes with frequency

- So inputs are $\mathbf{w} = \mathbf{w}_c e^{j\omega t}$, where $\mathbf{w}_c \in \mathbb{C}^m$
  - Then we get $\mathbf{z} = G(s)|_{s=\mathbf{j}\omega}\,\mathbf{w}, \quad \Rightarrow \mathbf{z} = \mathbf{z}_c e^{j\omega t}$ and $\mathbf{z}_c \in \mathbb{C}^p$
  - We need only analyze $\mathbf{z}_c = G(\mathrm{j}\omega)\mathbf{w}_c$

- As in the SISO case, we need a way to establish if the system response is **large** or **small**.
  - How much amplification we can get with a bounded input.

- Consider $\mathbf{z}_c = G(\mathrm{j}\omega)\mathbf{w}_c$ and set $\|\mathbf{w}_c\|_2 = \sqrt{\mathbf{w}_c^H \mathbf{w}_c} \leq 1$. What can we say about the $\|\mathbf{z}_c\|_2$?
  - Answer depends on ω and on the **direction** of the input $\mathbf{w}_c$
  - Best found using **singular values.**

Singular Value Decomposition

- Must perform **SVD** of the matrix $G(s)$ at each frequency $s = \mathbf{j}\omega$

$$G(\mathrm{j}\omega) \in \mathbb{C}^{p\times m} \quad U \in \mathbb{C}^{p\times p} \quad \Sigma \in \mathbb{R}^{p\times m} \quad V \in \mathbb{C}^{m\times m}$$

$$G = U\Sigma V^H$$

  - $U^HU = I$, $UU^H = I$, $V^HV = I$, $VV^H = I$, and Σ is diagonal.

  - Diagonal elements $\sigma_k \geq 0$ of Σ are the **singular values** of G.

$$\sigma_i = \sqrt{\lambda_i(G^HG)} \quad \text{or} \quad \sigma_i = \sqrt{\lambda_i(GG^H)}$$

    the positive ones are the same from both formulas.

  - Columns of matrices U and V (u_i and v_j) are the associated eigenvectors

$$\begin{aligned} G^HGv_j &= \sigma_j^2 v_j \\ GG^Hu_i &= \sigma_i^2 u_i \\ Gv_i &= \sigma_i u_i \end{aligned}$$

- If the rank$(G) = r \leq \min(p, m)$, then
  - $\sigma_k > 0$, $k = 1, \ldots, r$
  - $\sigma_k = 0$, $k = r+1, \ldots, \min(p, m)$
  - Singular values are sorted so that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$

- An SVD gives a **very detailed description** of how a matrix (the system G) acts on a vector (the input $\mathbf{w}$) at a particular frequency.

- So how can we use this result?
  - Fix the size $\|\mathbf{w}_c\|_2 = 1$ of the input, and see how large we can make the output.
  - Since we are working at a single frequency, we just analyze the relation

$$\mathbf{z}_c = G_w \mathbf{w}_c, \qquad G_w \equiv G(s = \mathbf{j}\omega)$$

- Define the maximum and minimum amplifications as:

$$\overline{\sigma} \equiv \max_{\|\mathbf{w}_c\|_2=1} \|\mathbf{z}_c\|_2$$

$$\underline{\sigma} \equiv \min_{\|\mathbf{w}_c\|_2=1} \|\mathbf{z}_c\|_2$$

- Then we have that (let $q = \min(p, m)$)

$$\overline{\sigma} = \sigma_1$$

$$\underline{\sigma} = \begin{cases} \sigma_q & p \geq m \quad \text{“tall”} \\ 0 & p < m \quad \text{“wide”} \end{cases}$$

- Can use $\overline{\sigma}$ and $\underline{\sigma}$ to determine the possible amplification and attenuation of the input signals.

- Since $G(s)$ changes with frequency, so will $\overline{\sigma}$ and $\underline{\sigma}$

SVD Example

- Consider (wide case)

$$G_w = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= U\Sigma V^H$$

so that $\sigma_1 = 5$ and $\sigma_2 = 0.5$

$$\overline{\sigma} \equiv \max_{\|\mathbf{w}_c\|_2=1} \|G_w \mathbf{w}_c\|_2 = 5 = \sigma_1$$

$$\underline{\sigma} \equiv \min_{\|\mathbf{w}_c\|_2=1} \|G_w \mathbf{w}_c\|_2 = 0 \neq \sigma_2$$

- But now consider (tall case)

$$\tilde{G}_w = \begin{bmatrix} 5 & 0 \\ 0 & 0.5 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 0 & 0.5 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= U\Sigma V^H$$

so that $\sigma_1 = 5$ and $\sigma_2 = 0.5$ still.

$$\overline{\sigma} \equiv \max_{\|\mathbf{w}_c\|_2=1} \|G_w \mathbf{w}_c\|_2 = 5 = \sigma_1$$

$$\underline{\sigma} \equiv \min_{\|\mathbf{w}_c\|_2=1} \|G_w \mathbf{w}_c\|_2 = 0.5 = \sigma_2$$

- For MIMO systems, the gains (or σ's) are only part of the story, as we must also consider the **input direction**.

- To analyze this point further, note that we can rewrite

$$G_w = U\Sigma V^H = \begin{bmatrix} u_1 & \dots & u_p \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \\ 0 & & \end{bmatrix} \begin{bmatrix} v_1^H \\ \vdots \\ v_m^H \end{bmatrix}$$

$$= \sum_{i=1}^{m} \sigma_i u_i v_i^H$$

  - Assumed tall case for simplicity, so $p > m$ and $q = m$

- Can now analyze impact of **various alternatives for the input**
  - Only looking at one frequency, so the basic signal is harmonic.
  - But, we are free to pick the relative **sizes** and **phases** of each of the components of the input vector $\mathbf{w}_c$.
    - ◇ These define the **input direction**

- For example, we could pick $\mathbf{w}_c = v_1$, then

$$\mathbf{z}_c = G_w \mathbf{w}_c = \left(\sum_{i=1}^{m} \sigma_i u_i v_i^H \right) v_1 = \sigma_1 u_1$$

since $v_i^H v_j = \delta_{ij}$.

  - Output amplified by σ_1. The relative **sizes** and **phases** of each of the components of the output are given by the vector $\mathbf{z}_c$.

- By selecting other input directions (at the same frequency), we can get quite different amplifications of the input signal

$$\underline{\sigma} \leq \frac{\|G_w \mathbf{w}_c\|_2}{\|\mathbf{w}_c\|_2} \leq \overline{\sigma}$$

- Thus we say that
  - G_w is large if $\underline{\sigma}(G_w) \gg 1$
  - G_w is small if $\overline{\sigma}(G_w) \ll 1$

- **MIMO frequency response** are plots of $\overline{\sigma}(\mathrm{j}\omega)$ and $\underline{\sigma}(\mathrm{j}\omega)$.
  - Then use the singular value vectors to analyze the response at a particular frequency.

16.323 Lecture 15

Signals and System Norms

$\mathcal{H}_\infty$ Synthesis

Different type of optimal controller

SP Skogestad and Postlethwaite(1996) Multivariable Feedback Control Wiley.
JB Burl (2000). Linear Optimal Control Addison-Wesley.
ZDG Zhou, Doyle, and Glover (1996). Robust and Optimal Control Prentice Hall.
MAC Maciejowski (1989) Multivariable Feedback Design Addison Wesley.

Mathematical Background

- **Signal norms** we use norms to measure the size of a signal.
  - Three key properties of a norm:
    1. $\|u\| \geq 0$, and $\|u\| = 0$ iff $u = 0$
    2. $\|\alpha u\| = |\alpha|\|u\|$ $\forall$ scalars α
    3. $\|u + v\| \leq \|u\| + \|v\|$

- Key signal norms

  - 2-norm of $u(t)$ – *Energy of the signal*

$$\|u(t)\|_2 \equiv \left[\int_{-\infty}^{\infty} u^2(t)dt\right]^{1/2}$$

  - ∞-norm of $u(t)$ – *maximum value over time*

$$\|u(t)\|_\infty = \max_t |u(t)|$$

  - Other useful measures include the *Average power*

$$\text{pow}(u(t)) = \left(\lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} u^2(t)dt\right)^{1/2}$$

  $u(t)$ is called a *power signal* if $\text{pow}(u(t)) < \infty$

- **System norms** Consider the system with dynamics $y = G(s)u$
  - Assume $G(s)$ stable, LTI transfer function matrix
  - $g(t)$ is the associated impulse response matrix (causal).

- $\mathcal{H}_2$ norm for the system: (LQG problem)

$$\begin{aligned}\|G\|_2 &= \left(\frac{1}{2\pi}\int_{-\infty}^{\infty} \mathtt{trace}[G^H(\mathrm{j}\omega)G(\mathrm{j}\omega)]d\omega\right)^{1/2} \\ &= \left(\int_0^{\infty} \mathtt{trace}[g^T(\tau)g(\tau)]d\tau\right)^{1/2}\end{aligned}$$

  Two interpretations:
  - For SISO: energy in the output $y(t)$ for a unit impulse input $u(t)$.
  - For MIMO [27]: apply an impulsive input separately to each actuator and measure the response z_i, then

$$\|G\|_2^2 = \sum_i \|z_i\|_2^2$$

  - Can also interpret as the expected RMS value of the output in response to unit-intensity white noise input excitation.

- **Key point:** Can show that

$$\|G\|_2 = \left(\frac{1}{2\pi}\int_{-\infty}^{\infty}\sum_i \sigma_i^2[G(\mathrm{j}\omega)]d\omega\right)^{1/2}$$

  - Where $\sigma_i[G(\mathrm{j}\omega)]$ is the ith singular value[28] [29] of the system $G(s)$ evaluated at $s = \mathbf{j}\omega$
  - $\mathcal{H}_2$ norm concerned with **overall performance** ($\sum_i \sigma_i^2$) **over all** frequencies

[27] ZDG114

[28] http://mathworld.wolfram.com/SingularValueDecomposition.html

[29] http://en.wikipedia.org/wiki/Singular_value_decomposition

- $\mathcal{H}_\infty$ norm for the system:

$$\|G(s)\|_\infty = \sup_\omega \overline{\sigma}[G(\mathrm{j}\omega)]$$

Interpretation:

– $\|G(s)\|_\infty$ is the "energy gain" from the input u to output y

$$\|G(s)\|_\infty = \max_{u(t)\neq 0} \frac{\int_0^\infty y^T(t)y(t)dt}{\int_0^\infty u^T(t)u(t)dt}$$

– Achieve this maximum gain using a **worst case** input signal that is essentially a sinusoid at frequency $\omega^\star$ with input direction that yields $\overline{\sigma}[G(\mathrm{j}\omega^\star)]$ as the amplification.

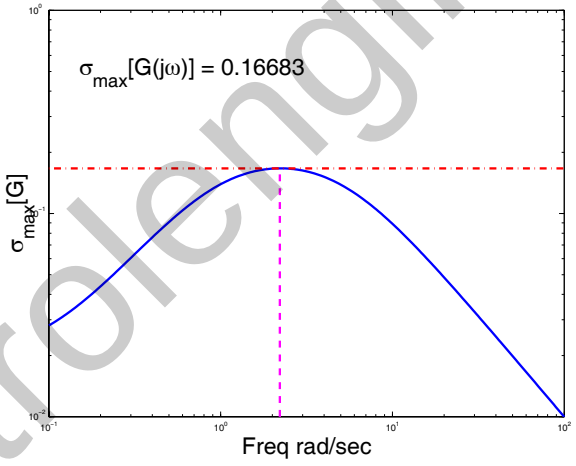


Figure 15.1: Graphical test for the $\|G\|_\infty$.

- Note that we now have

1. Signal norm $\|u(t)\|_\infty = \max_t |u(t)|$
2. Vector norm $\|x\|_\infty = \max_i |x_i|$
3. System norm $\|G(s)\|_\infty = \max_\omega \overline{\sigma}[G(\mathrm{j}\omega)]$

We use the same symbol $\|\cdot\|_\infty$ for all three, but there is typically no confusion, as the norm to be used is always clear by the context.

Key Points / Summary

- So $\mathcal{H}_\infty$ is concerned primarily with the **peaks** in the frequency response, and the $\mathcal{H}_2$ norm is concerned with the **overall** response.

- The $\mathcal{H}_\infty$ norm satisfies the **submultiplicative property**

$$\|GH\|_\infty \leq \|G\|_\infty \cdot \|H\|_\infty$$

  – Will see that this is an essential property for the robustness tests
  – **Does not hold** in general for $\|GH\|_2$

- Reference to $\mathcal{H}_\infty$ **control** is that we would like to design a stabilizing controller that ensures that the peaks in the transfer function matrix of interest are *knocked down*.

$$\text{e.g. want} \quad \max_\omega \overline{\sigma}[T(\mathrm{j}\omega)] \equiv \|T(s)\|_\infty < 0.75$$

- Reference to $\mathcal{H}_2$ **control** is that we would like to design a stabilizing controller that reduces the $\|T(s)\|_2$ as much as possible.
  – Note that $\mathcal{H}_2$ control and LQG are the same thing.

Computation

- Assume that $G(s) = C(sI - A)^{-1}B + D$ with $\mathcal{R}\lambda(A) < 0$, i.e. $G(s)$ stable.

- $\mathcal{H}_2$ norm: requires a strictly proper system $D = 0$

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}$$

  – Define:

  Observability Gramian P_o

$$A^T P_o + P_o A + C^T C = 0 \Leftrightarrow P_o = \int_0^\infty e^{A^T t} C^T C e^{At} dt$$

  Controllability Gramian P_c

$$AP_c + P_c A^T + BB^T = 0 \Leftrightarrow P_c = \int_0^\infty e^{At} BB^T e^{A^T t} dt$$

  then

$$\|G\|_2^2 = \mathtt{trace}\left(B^T P_o B\right) = \mathtt{trace}\left(C P_c C^T\right)$$

  Proof: use the impulse response of the system $G(s)$ and evaluate the time-domain version of the norm.

- $\mathcal{H}_\infty$ norm: Define the **Hamiltonian matrix**

$$H = \left[\begin{array}{c|c} A + B(\gamma^2 I - D^T D)^{-1} D^T C & B(\gamma^2 I - D^T D)^{-1} B^T \\ \hline -C^T (I + D(\gamma^2 I - D^T D)^{-1} D^T) C & -(A + B(\gamma^2 I - D^T D)^{-1} D^T C)^T \end{array}\right]$$

  – Then $\|G(s)\|_\infty < \gamma$ iff $\overline{\sigma}(D) < \gamma$ and H has no eigenvalues on the $\mathbf{j}\omega$-axis.

  – Graphical test $\max_\omega \overline{\sigma}[G(\mathrm{j}\omega)] < \gamma$ replaced with eigenvalue test.

Issues

- Note that it is **not easy** to find $\|G\|_\infty$ directly using the state space techniques
  - It is easy to check if $\|G\|_\infty < \gamma$
  - So we just keep changing γ to find the smallest value for which we can show that $\|G\|_\infty < \gamma$ (called $\gamma_{\min}$)

    $\Rightarrow$ Bisection search algorithm.

- **Bisection search algorithm**

  1. Select γ_u, γ_l so that $\gamma_l \leq \|G\|_\infty \leq \gamma_u$

  2. Test $(\gamma_u - \gamma_l)/\gamma_l < \text{TOL}$.
     Yes $\Rightarrow$ Stop ($\|G\|_\infty \approx \frac{1}{2}(\gamma_u + \gamma_l)$)
     No $\Rightarrow$ go to step 3.

  3. With $\gamma = \frac{1}{2}(\gamma_l + \gamma_u)$, test if $\|G\|_\infty < \gamma$ using $\lambda_i(H)$

  4. If $\lambda_i(H) \in \mathbf{j}\mathcal{R}$, then set $\gamma_l = \gamma$ (test value too low), otherwise set $\gamma_u = \gamma$ and go to step 2.

Application

- Note that we can use the state space tests to analyze the weighted tests that we developed for robust stability

  – For example, we have seen the value in ensuring that the sensitivity remains smaller than a particular value

$$\overline{\sigma}[W_i S(\mathrm{j}\omega)] < 1 \quad \forall \quad \omega$$

- We can test this by determining if $\|W_i(s)S(s)\|_\infty < 1$
  – Use state space models of $G_c(s)$ and $G(s)$ to develop a state space model of

$$S(s) := \left[\begin{array}{c|c} A_s & B_s \\ \hline C_s & 0 \end{array}\right]$$

  – Augment these dynamics with the (stable, min phase) $W_i(s)$ to get a model of $W_i(s)S(s)$

$$W_i(s) := \left[\begin{array}{c|c} A_w & B_w \\ \hline C_w & 0 \end{array}\right]$$

$$W_i(s)S(s) := \left[\begin{array}{cc|c} A_s & 0 & B_s \\ B_w C_s & A_w & 0 \\ \hline 0 & C_w & 0 \end{array}\right]$$

  – Now compute the $\mathcal{H}_\infty$ norm of the combined system $W_i(s)S(s)$.

Riccati Equation Tests

- Note that, with $D = 0$, the $\mathcal{H}_\infty$ **Hamiltonian matrix** becomes

$$H = \begin{bmatrix} A & \frac{1}{\gamma^2} BB^T \\ -C^T C & -A^T \end{bmatrix}$$

  - Know that $\|G\|_\infty < \gamma$ iff H has no eigenvalues on the **j**ω-axis.
  - Equivalent test is if there exists a $X \geq 0$ such that

$$A^T X + XA + C^T C + \frac{1}{\gamma^2} XBB^T X = 0$$

    and $A + \frac{1}{\gamma^2} BB^T X$ is stable.
  - So there is a direction relationship between the Hamiltonian matrix H and the **algebraic Riccati Equation** (ARE)

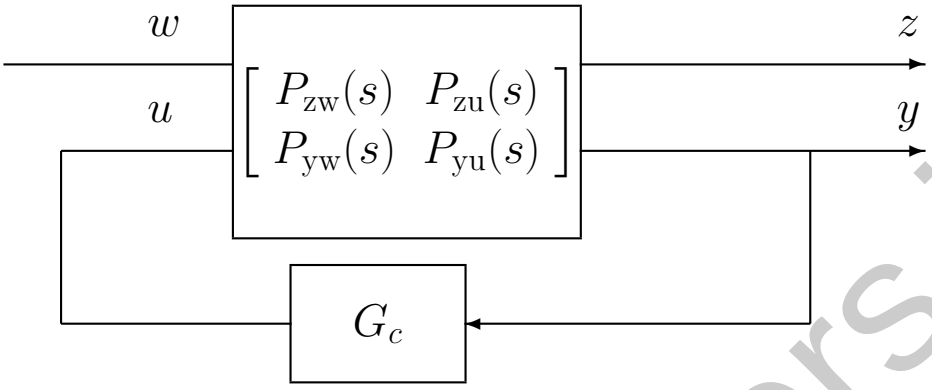
- **Aside:** Compare this ARE with the one that we would get if we used this system in an LQR problem:

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

  - If (A, B, C) stabilizable/detectable, then will always get a solution for the LQR ARE.
  - Sign difference in quadratic term of the $\mathcal{H}_\infty$ ARE makes this equation harder to satisfy. Consistent with the fact that we could have $\|G\|_\infty > \gamma \Rightarrow$ no solution to the $\mathcal{H}_\infty$ ARE.
  - The two Riccati equations look similar, but with the sign change, the solutions can behave **very** differently.

Synthesis

- For the synthesis problem, we typically define a generalized version of the system dynamics



Signals:

- z Performance output
- w Disturbance/ref inputs
- y Sensor outputs
- u Actuator inputs

Generalized plant:

$$P(s) = \begin{bmatrix} P_{zw}(s) & P_{zu}(s) \\ P_{yw}(s) & P_{yu}(s) \end{bmatrix}$$

contains the plant $G(s)$ and all performance and uncertainty weights

- With the loop closed ($u = G_c y$), can show that

$$\left(\frac{z}{w}\right)_{CL} = P_{zw} + P_{zu}G_c(I - P_{yu}G_c)^{-1}P_{yw} \equiv F_l(P, G_c)$$

called a (lower) **Linear Fractional Transformation** (LFT).

- **Design Objective:** Find $G_c(s)$ to stabilize the closed-loop system and minimize $\|F_l(P, G_c)\|_\infty$.

- Hard problem to solve, so we typically consider a suboptimal problem:
  - Find $G_c(s)$ to satisfy $\|F_l(P, G_c)\|_\infty < \gamma$
  - Then use bisection (called a γ **iteration**) to find the smallest value (γ_{opt}) for which $\|F_l(P, G_c)\|_\infty < \gamma_{opt}$

  $\Rightarrow$ hopefully get that G_c approaches G_c^{opt}

- Consider the suboptimal $\mathcal{H}_\infty$ synthesis problem: [30]

$$\text{Find } G_c(s) \text{ to satisfy } \|F_l(P, G_c)\|_\infty < \gamma$$

$$P(s) = \begin{bmatrix} P_{\text{zw}}(s) & P_{\text{zu}}(s) \\ P_{\text{yw}}(s) & P_{\text{yu}}(s) \end{bmatrix} := \left[\begin{array}{c|cc} A & B_{\text{w}} & B_{\text{u}} \\ \hline C_{\text{z}} & 0 & D_{\text{zu}} \\ C_{\text{y}} & D_{\text{yw}} & 0 \end{array}\right]$$

where we assume that:

1. $(A, B_{\text{u}}, C_{\text{y}})$ is stabilizable/detectable (essential)
2. $(A, B_{\text{w}}, C_{\text{z}})$ is stabilizable/detectable (essential)
3. $D_{\text{zu}}^T [\, C_{\text{z}} \;\; D_{\text{zu}} \,] = [\, 0 \;\; I \,]$ (simplify/essential)
4. $\begin{bmatrix} B_{\text{w}} \\ D_{\text{yw}} \end{bmatrix} D_{\text{yw}}^T = \begin{bmatrix} 0 \\ I \end{bmatrix}$ (simplify/essential)

- Note that we will not cover all the details of the solution to this problem – it is well covered in the texts.

[30] SP367

- There exists a stabilizing $G_c(s)$ such that $\|F_l(P, G_c)\|_\infty < \gamma$ iff

  (1) $\exists X \geq 0$ that solves the ARE

  $$A^T X + XA + C_{\mathrm{z}}^T C_{\mathrm{z}} + X(\gamma^{-2} B_{\mathrm{w}} B_{\mathrm{w}}^T - B_{\mathrm{u}} B_{\mathrm{u}}^T)X = 0$$

  and $\mathcal{R}\lambda_i \left[A + (\gamma^{-2} B_{\mathrm{w}} B_{\mathrm{w}}^T - B_{\mathrm{u}} B_{\mathrm{u}}^T)X\right] < 0 \ \ \forall\, i$

  (2) $\exists Y \geq 0$ that solves the ARE

  $$AY + YA^T + B_{\mathrm{w}}^T B_{\mathrm{w}} + Y(\gamma^{-2} C_{\mathrm{z}}^T C_{\mathrm{z}} - C_{\mathrm{y}}^T C_{\mathrm{y}})Y = 0$$

  and $\mathcal{R}\lambda_i \left[A + Y(\gamma^{-2} C_{\mathrm{z}}^T C_{\mathrm{z}} - C_{\mathrm{y}}^T C_{\mathrm{y}})\right] < 0 \ \ \forall\, i$

  (3) $\rho(XY) < \gamma^2$

  ρ is the **spectral radius** ($\rho(A) = \max_i |\lambda_i(A)|$).

- Given these solutions, the **central $\mathcal{H}_\infty$ controller** is given by

$$G_c(s) := \left[\begin{array}{c|c} A + (\gamma^{-2} B_{\mathrm{w}} B_{\mathrm{w}}^T - B_{\mathrm{u}} B_{\mathrm{u}}^T)X - ZYC_{\mathrm{y}}^T C_{\mathrm{y}} & ZYC_{\mathrm{y}}^T \\ \hline -B_{\mathrm{u}}^T X & 0 \end{array}\right]$$

  where $Z = (I - \gamma^{-2} YX)^{-1}$

  – Central controller has as many states as the generalized plant.

- Note that this design does not decouple as well as the regulator/estimator for LQG

Observations

- Basic assumptions:

(A1) $(A, B_{\rm u}, C_{\rm y})$ is stabilizable/detectable
(A2) $(A, B_{\rm w}, C_{\rm z})$ is stabilizable/detectable
(A3) $D_{\rm zu}^T[\ C_{\rm z}\ \ D_{\rm zu}\ ] = [\ 0\ \ I\ ]$ (scaling and no cross-coupling)
(A4) $\begin{bmatrix} B_{\rm w} \\ D_{\rm yw} \end{bmatrix} D_{\rm yw}^T = \begin{bmatrix} 0 \\ I \end{bmatrix}$ (scaling and no cross-coupling)

- The restrictions that $D_{\rm zw} = 0$ and $D_{\rm yu} = 0$ are weak, and can easily be removed (the codes handle the more general D case).

- (A1) is required to ensure that it is even possible to get a stabilizing controller.

- Need $D_{\rm zu}$ and $D_{\rm yw}$ to have full rank to ensure that we penalize control effort (A3) and include sensor noise (A4)
  $\Rightarrow$ Avoids singular case with infinite bandwidth controllers.
  $\Rightarrow$ Often where you will have the most difficulties initially.

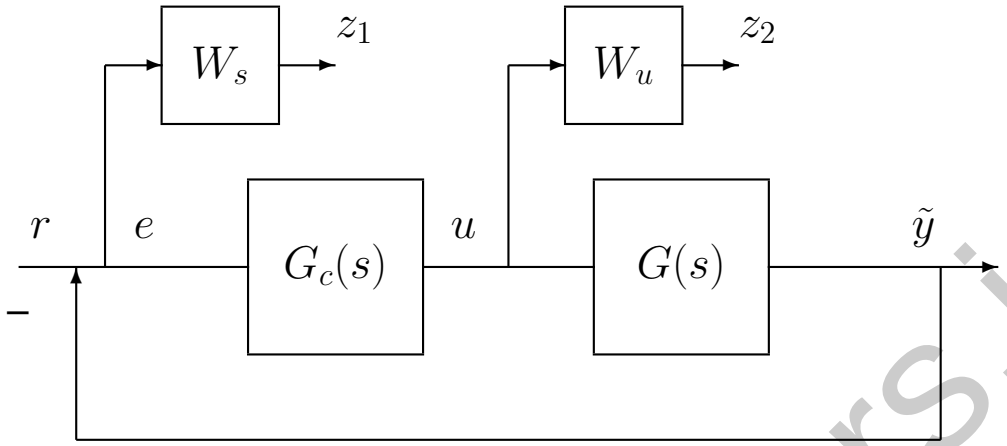
- Typically will see two of the assumptions written as:

(Ai) $\begin{bmatrix} A - \mathbf{j}\omega I & B_{\rm u} \\ C_{\rm z} & D_{\rm zu} \end{bmatrix}$ has full column rank $\forall\ \ \omega$

(Aii) $\begin{bmatrix} A - \mathbf{j}\omega I & B_{\rm w} \\ C_{\rm y} & D_{\rm yw} \end{bmatrix}$ has full row rank $\forall\ \ \omega$

  - These ensure that there are **no** $\mathbf{j}\omega$-axis zeros in the $P_{\rm zu}$ or $P_{\rm yw}$ TF's – cannot have the controller canceling these, because that design would not internally stabilize the closed-loop system.
  - But with assumptions (A3) and (A4) given above, can show that A(i) and A(ii) are equivalent to our assumption (A2).

Simple Design Example



where

$$G = \frac{200}{(0.05s+1)^2(10s+1)}$$

- Note that we have 1 input (r) and two performance outputs - one that penalizes the sensitivity $S(s)$ of the system, and the other that penalizes the control effort used.

- Easy to show (see next page) that the closed-loop is:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} W_s S \\ W_u G_c S \end{bmatrix} r$$

  where, in this case, the input r acts as the "disturbance input" w to the generalized system.

- To achieve good low frequency tracking and a crossover frequency of about 10 rad/sec, pick

$$W_s = \frac{s/1.5 + 10}{s + (10)\cdot(0.0001)} \qquad W_u = 1$$

- Generalized system in this case:

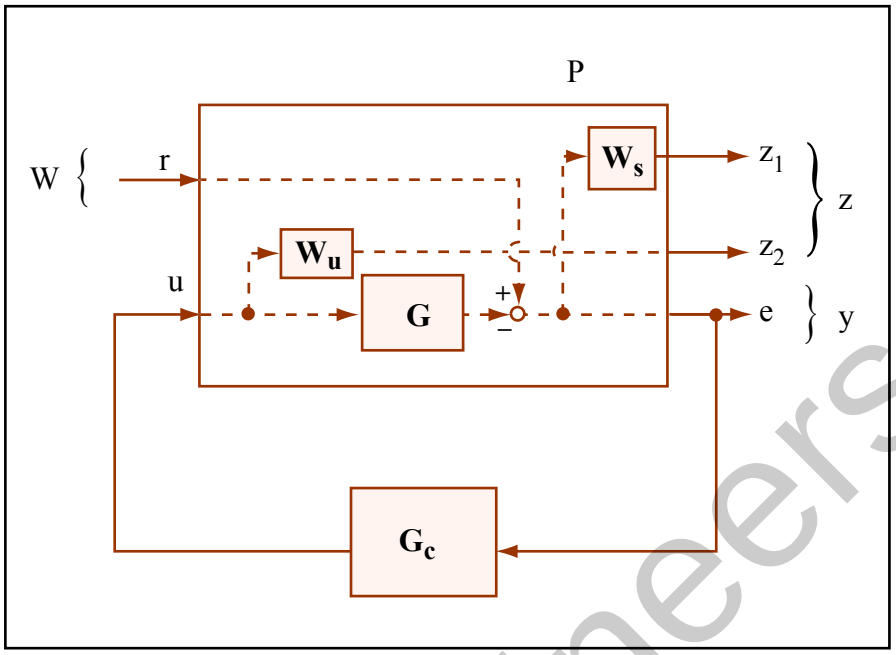


Figure by MIT OpenCourseWare.

Figure 15.2: Rearrangement of original picture in the generalized plant format.

- Derive $P(s)$ as

$$\begin{aligned} z_1 &= W_s(s)(r - Gu) \\ z_2 &= W_u u \\ e &= r - Gu \\ u &= G_c e \end{aligned}$$

$$\begin{aligned} P(s) &= \left[\begin{array}{c|c} W_s(s) & -W_s(s)G(s) \\ 0 & W_u(s) \\ \hline 1 & -G(s) \end{array}\right] \\ &= \left[\begin{array}{cc} P_{\rm zw}(s) & P_{\rm zu}(s) \\ P_{\rm yw}(s) & P_{\rm yu}(s) \end{array}\right] \end{aligned}$$

$$\begin{aligned} P_{CL} &= F_l(P, G_c) \\ &= \left[\begin{array}{c} W_s \\ 0 \end{array}\right] + \left[\begin{array}{c} -W_s G \\ W_u \end{array}\right] G_c(I + GG_c)^{-1} 1 \\ &= \left[\begin{array}{c} W_s - W_s G G_c S \\ W_u G_c S \end{array}\right] = \left[\begin{array}{c} W_s S \\ W_u G_c S \end{array}\right] \end{aligned}$$

- In state space form, let

$$G(s) := \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right] \quad W_s(s) := \left[\begin{array}{c|c} A_w & B_{\rm w} \\ \hline C_w & D_w \end{array}\right] \quad W_u = 1$$

$$\begin{aligned} \dot{x} &= Ax + Bu \\ \dot{x}_w &= A_w x_w + B_{\rm w} e = A_w x_w + B_{\rm w} r - B_{\rm w} C x \\ z_1 &= C_w x_w + D_w e = C_w x_w + D_w r - D_w C x \\ z_2 &= W_u u \\ e &= r - Cx \end{aligned}$$

$$P(s) := \left[\begin{array}{cc|c|c} A & 0 & 0 & B \\ -B_{\rm w}C & A_w & B_{\rm w} & 0 \\ \hline -D_w C & C_w & D_w & 0 \\ \hline 0 & 0 & 0 & W_u \\ \hline -C & 0 & 1 & 0 \end{array}\right]$$

- Now use the mu-tools code to solve for the controller. (Could also have used the robust control toolbox code).

```
A=[Ag zeros(n1,n2);-Bsw*Cg Asw];
Bw=[zeros(n1,1);Bsw];
Bu=[Bg;zeros(n2,1)];
Cz=[-Dsw*Cg Csw;zeros(1,n1+n2)];
Cy=[-Cg zeros(1,n2)];
Dzw=[Dsw;0];
Dzu=[0;1];
Dyw=[1];
Dyu=0;
P=pck(A,[Bw Bu],[Cz;Cy],[Dzw Dzu;Dyw Dyu]);
% call hinf to find Gc (mu toolbox)
[Gc,G,gamma]=hinfsyn(P,1,1,0.1,20,.001);
```

- Results from the γ-iteration showing whether we pass or fail the various X, Y, $\rho(XY)$ tests as we keep searching over γ, starting at the initial bound of 20.

```
Resetting value of Gamma min based on D_11, D_12, D_21 terms

Test bounds:      0.6667 <  gamma  <=     20.0000

    gamma    hamx_eig  xinf_eig  hamy_eig   yinf_eig   nrho_xy   p/f
     20.000  9.6e+000  6.2e-008  1.0e-003  0.0e+000    0.0000    p
     10.333  9.6e+000  6.3e-008  1.0e-003  0.0e+000    0.0000    p
      5.500  9.5e+000  6.3e-008  1.0e-003  0.0e+000    0.0000    p
      3.083  9.5e+000  6.5e-008  1.0e-003  0.0e+000    0.0000    p
      1.875  9.4e+000  6.9e-008  1.0e-003  0.0e+000    0.0000    p
>>    1.271  9.1e+000 -1.2e+004# 1.0e-003   -4.5e-010    0.0000    f
      1.573  9.3e+000  7.3e-008  1.0e-003  0.0e+000    0.0000    p
      1.422  9.2e+000  7.6e-008  1.0e-003  0.0e+000    0.0000    p
>>    1.346  9.2e+000 -6.4e+004# 1.0e-003   0.0e+000    0.0000    f
      1.384  9.2e+000  7.7e-008  1.0e-003  0.0e+000    0.0000    p
>>    1.365  9.2e+000 -1.9e+006# 1.0e-003   0.0e+000    0.0000    f
      1.375  9.2e+000  7.7e-008  1.0e-003   -4.5e-010   0.0000    p
      1.370  9.2e+000  7.7e-008  1.0e-003  0.0e+000    0.0000    p
      1.368  9.2e+000  7.7e-008  1.0e-003  0.0e+000    0.0000    p
      1.366  9.2e+000  7.7e-008  1.0e-003  0.0e+000    0.0000    p
>>    1.366  9.2e+000 -1.3e+007# 1.0e-003   0.0e+000    0.0000    f

 Gamma value achieved:     1.3664
```

- Since $\gamma_{\min} = 1.3664$, this indicates that we **did not** meet the desired goal of $|S| < 1/|W_s|$ (can only say that $|S| < 1.3664/|W_s|$).
  - Confirmed by the plot, which shows that we just fail the test (blue line passes above magenta)

- But note that, even though this design fails the sensitivity weight - we still get pretty good performance
  - For performance problems, can think of the objective of getting $\gamma_{\min} < 1$ as a "design goal" $\rightsquigarrow$ it is "not crucial"
  - Use W_u to tune the control design

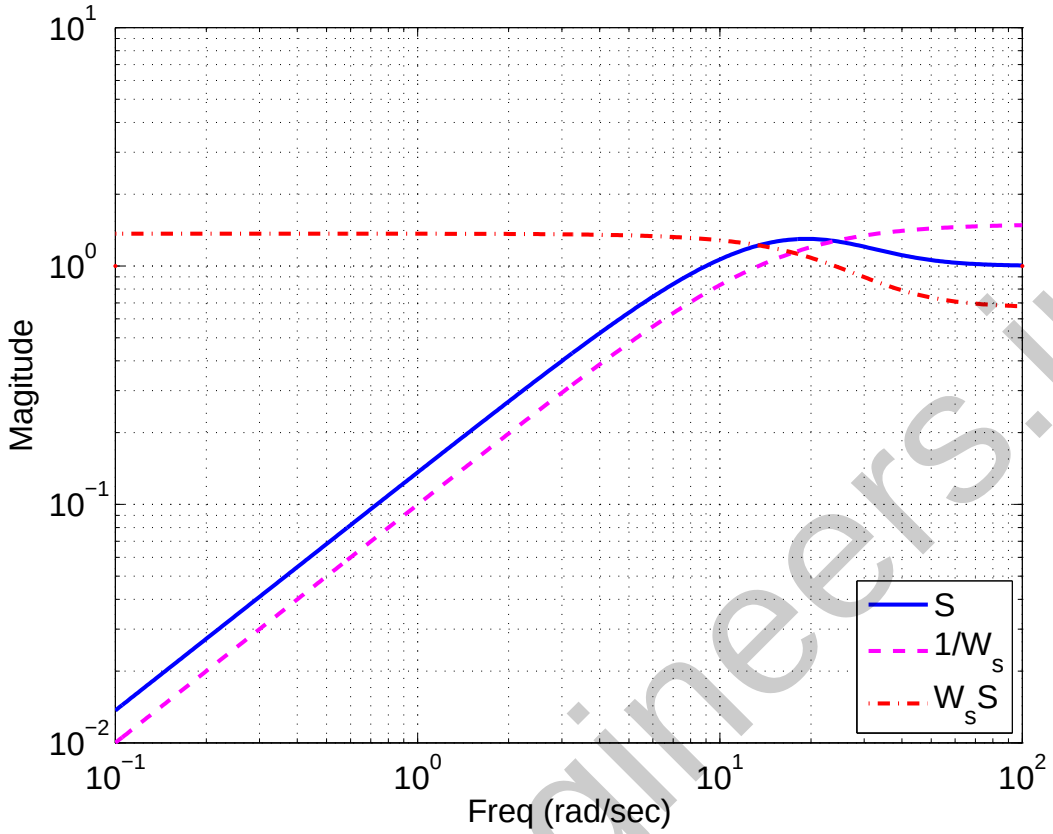


Figure 15.3: Visualization of the weighted sensitivity tests.

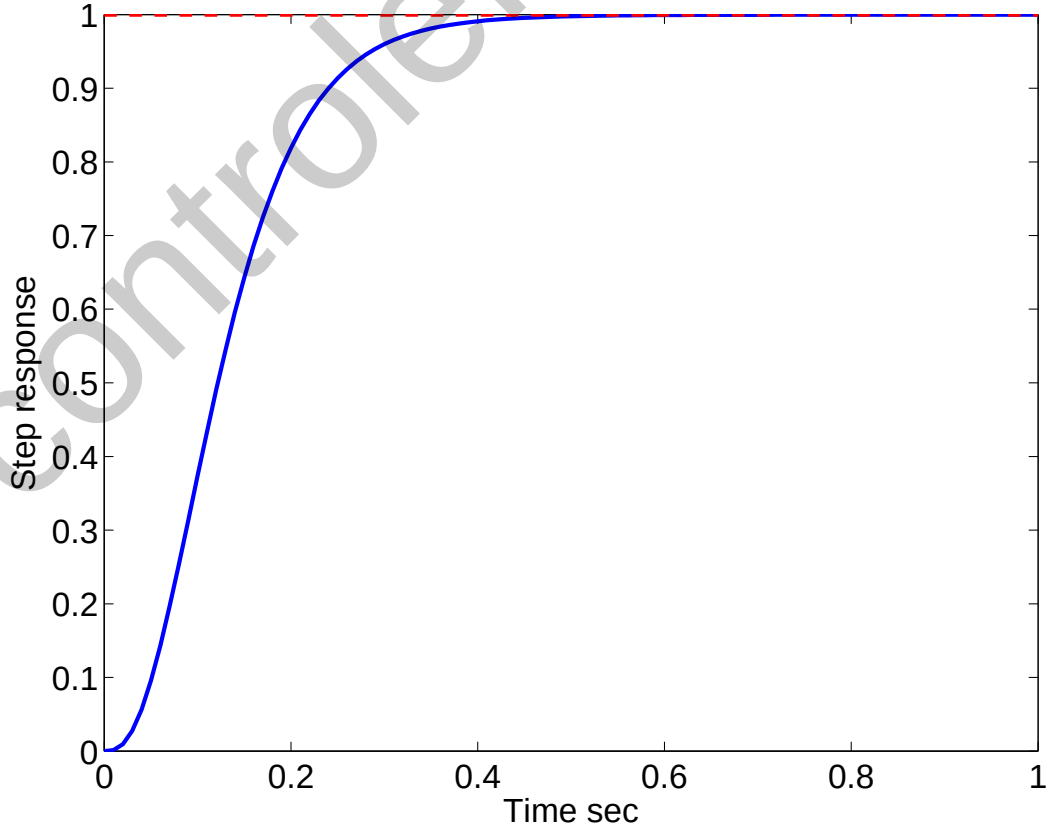


Figure 15.4: Time response of controller that yields $\gamma_{\min} = 1.3664$.

General LQG Problem

- Can also put LQG ($\mathcal{H}_2$) design into this generalized framework [31].
- Define the dynamics

$$\begin{aligned} \dot{x} &= Ax + Bu + w_d \\ y &= Cx + w_n \end{aligned}$$

where

$$E\left\{ \begin{bmatrix} w_d(t) \\ w_n(t) \end{bmatrix} \begin{bmatrix} w_d^T(\tau) & w_n^T(\tau) \end{bmatrix} \right\} = \begin{bmatrix} W & 0 \\ 0 & V \end{bmatrix} \delta(t-\tau)$$

- LQG problem is to find controller $u = G_c(s)y$ that minimizes

$$J = E\left\{ \lim_{T\to\infty} \frac{1}{T} \int_0^T (x^T R_{xx} x + u^T R_{uu} u) dt \right\}$$

- To put this problem in the general framework, define

$$z = \begin{bmatrix} R_{xx}^{1/2} & 0 \\ 0 & R_{uu}^{1/2} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} w_d \\ w_n \end{bmatrix} = \begin{bmatrix} W^{1/2} & 0 \\ 0 & V^{1/2} \end{bmatrix} w$$

where w is a unit intensity white noise process.

- With $z = F_l(P, G_c)w$, the LQG cost function can be rewritten as

$$J = E\left\{ \lim_{T\to\infty} \frac{1}{T} \int_0^T z^T(t) z(t) dt \right\} = \|F_l(P, G_c)\|_2^2$$

- In this case the generalized plant matrix is

$$P(s) := \left[\begin{array}{c|cc|c} A & W^{1/2} & 0 & B \\ \hline R_{xx}^{1/2} & 0 & 0 & 0 \\ 0 & 0 & 0 & R_{uu}^{1/2} \\ \hline C & 0 & V^{1/2} & 0 \end{array}\right]$$

[31] SP365

Controller Interpretations

- Given these solutions, the **central $\mathcal{H}_\infty$ controller** is given by

$$G_c(s) := \left[\begin{array}{c|c} A + (\gamma^{-2} B_{\rm w} B_{\rm w}^T - B_{\rm u} B_{\rm u}^T) X - Z Y C_{\rm y}^T C_{\rm y} & Z Y C_{\rm y}^T \\ \hline -B_{\rm u}^T X & 0 \end{array}\right]$$

  where $Z = (I - \gamma^{-2} Y X)^{-1}$

- Can develop a further interpretation of this controller if we rewrite the dynamics as:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + \gamma^{-2} B_{\rm w} B_{\rm w}^T X \hat{x} - B_{\rm u} B_{\rm u}^T X \hat{x} - Z Y C_{\rm y}^T C_{\rm y} \hat{x} + Z Y C_{\rm y}^T y \\ u &= -B_{\rm u}^T X \hat{x} \end{aligned}$$

$$\Rightarrow \dot{\hat{x}} = A\hat{x} + B_{\rm w}\left[\gamma^{-2} B_{\rm w}^T X \hat{x}\right] + B_{\rm u}\left[-B_{\rm u}^T X \hat{x}\right] + Z Y C_{\rm y}^T \left[y - C_{\rm y}\hat{x}\right]$$

$$\Rightarrow \dot{\hat{x}} = A\hat{x} + B_{\rm w}\left[\gamma^{-2} B_{\rm w}^T X \hat{x}\right] + B_{\rm u} u + L\left[y - C_{\rm y}\hat{x}\right]$$

  looks very similar to Kalman Filter developed for LQG controller.

- The difference is that we have an additional input $\hat{w}_{\rm worst} = \gamma^{-2} B_{\rm w}^T X \hat{x}$ that enters through $B_{\rm w}$.
  – $w_{\rm worst}$ is an estimate of **worst-case** disturbance to the system.

- Finally, note that a separation rule does exist for the $\mathcal{H}_\infty$ controller. But we will not discuss it.

Code: $\mathcal{H}_\infty$ Synthesis

```
% Hinf example
% 16.323 MIT Spring 2007
% Jon How
%
set(0,'DefaultAxesFontName','arial')
set(0,'DefaultAxesFontSize',16)
set(0,'DefaultTextFontName','arial')
set(0,'DefaultTextFontSize',20)

clear all
if ~exist('yprev')
   yprev=[1 1]';
   tprev=[0 1]';
   Sensprev=[1 1];
   fprev=[.1 100];
end

%Wu=1/1e9;
Wu=1;
% define plant
[Ag,Bg,Cg,Dg]=tf2ss(200,conv(conv([0.05 1],[0.05 1]),[10 1]));
Gol=ss(Ag,Bg,Cg,Dg);
% define sensitivity weight
M=1.5;wB=10;A=1e-4;
[Asw,Bsw,Csw,Dsw]=tf2ss([1/M wB],[1 wB*A]);
Ws=ss(Asw,Bsw,Csw,Dsw);
% form augmented P dynamics
n1=size(Ag,1);
n2=size(Asw,1);
A=[Ag zeros(n1,n2);-Bsw*Cg Asw];
Bw=[zeros(n1,1);Bsw];
Bu=[Bg;zeros(n2,1)];
Cz=[-Dsw*Cg Csw;zeros(1,n1+n2)];
Cy=[-Cg zeros(1,n2)];
Dzw=[Dsw;0];
Dzu=[0;Wu];
Dyw=[1];
Dyu=0;
P=pck(A,[Bw Bu],[Cz;Cy],[Dzw Dzu;Dyw Dyu]);

% call hinf to find Gc (mu toolbox)
diary hinf1_diary
[Gc,G,gamma]=hinfsyn(P,1,1,0.1,20,.001);
diary off

[ac,bc,cc,dc]=unpck(Gc);
ev=max(real(eig(ac)/2/pi))

PP=ss(A,[Bw Bu],[Cz;Cy],[Dzw Dzu;Dyw Dyu]);
GGc=ss(ac,bc,cc,dc);
CLsys = feedback(PP,GGc,[2],[3],1);
[acl,bcl,ccl,dcl]=ssdata(CLsys);
% reduce closed-loop system so that it only has
% 1 input and 2 outputs
bcl=bcl(:,1);ccl=ccl([1 2],:);dcl=dcl([1 2],1);
CLsys=ss(acl,bcl,ccl,dcl);

f=logspace(-1,2,400);
Pcl=freqresp(CLsys,f);
CLWS=squeeze(Pcl(1,1,:)); % closed loop weighted sens
WS=freqresp(Ws,f); % sens weight
SensW=squeeze(WS(1,1,:));
Sens=CLWS./SensW; % divide out weight to get closed-loop sens
figure(1);clf
loglog(f,abs(Sens),'b-','LineWidth',2)
hold on
loglog(f,abs(1./SensW),'m--','LineWidth',2)
```

```
68  loglog(f,abs(CLWS),'r-.','LineWidth',2)
69  loglog(fprev,abs(Sensprev),'r.')
70  legend('S','1/W_s','W_sS','Location','SouthEast')
71  hold off
72  xlabel('Freq (rad/sec)')
73  ylabel('Magitude')
74  grid
75
76  print -depsc hinf1.eps;jpdf('hinf1')
77
78  na=size(Ag,1);
79  nac=size(ac,1);
80  Acl=[Ag Bg*cc;-bc*Cg ac];Bcl=[zeros(na,1);bc];Ccl=[Cg zeros(1,nac)];Dcl=0;
81  Gcl=ss(Acl,Bcl,Ccl,Dcl);
82  [y,t]=step(Gcl,1);
83
84  figure(2);clf
85  plot(t,y,'LineWidth',2)
86  hold on;plot(tprev,yprev,'r--','LineWidth',2);hold off
87  xlabel('Time sec')
88  ylabel('Step response')
89
90  print -depsc hinf12.eps;jpdf('hinf12')
91
92  yprev=y;
93  tprev=t;
94  Sensprev=Sens;
95  fprev=f;
```

16.323 Lecture 16

Model Predictive Control

- Allgower, F., and A. Zheng, Nonlinear Model Predictive Control, Springer-Verlag, 2000.
- Camacho, E., and C. Bordons, Model Predictive Control, Springer-Verlag, 1999.
- Kouvaritakis, B., and M. Cannon, Non-Linear Predictive Control: Theory & Practice, IEE Publishing, 2001.
- Maciejowski, J., Predictive Control with Constraints, Pearson Education POD, 2002.
- Rossiter, J. A., Model-Based Predictive Control: A Practical Approach, CRC Press, 2003.

MPC

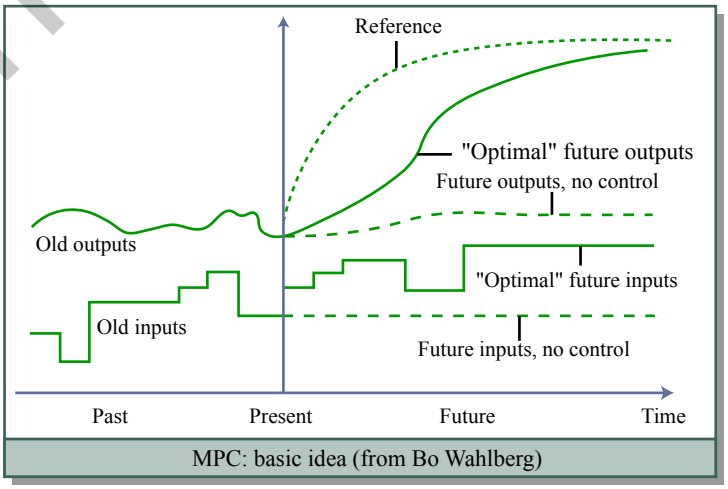
- Planning in Lecture 8 was effectively "open-loop"
  - Designed the control input sequence $\mathbf{u}(t)$ using an assumed model and set of constraints.
  - Issue is that with modeling error and/or disturbances, these inputs will not necessarily generate the desired system response.

- Need a "closed-loop" strategy to compensate for these errors.
  - Approach called **Model Predictive Control**
  - Also known as **receding horizon control**

- Basic strategy:
  - At time k, use knowledge of the system model to design an input sequence

$$\mathbf{u}(k|k), \mathbf{u}(k+1|k), \mathbf{u}(k+2|k), \mathbf{u}(k+3|k), \ldots, \mathbf{u}(k+N|k)$$

    over a finite horizon N from the current state $\mathbf{x}(k)$
  - Implement a fraction of that input sequence, usually just first step.
  - Repeat for time $k+1$ at state $\mathbf{x}(k+1)$



MPC: basic idea (from Bo Wahlberg)

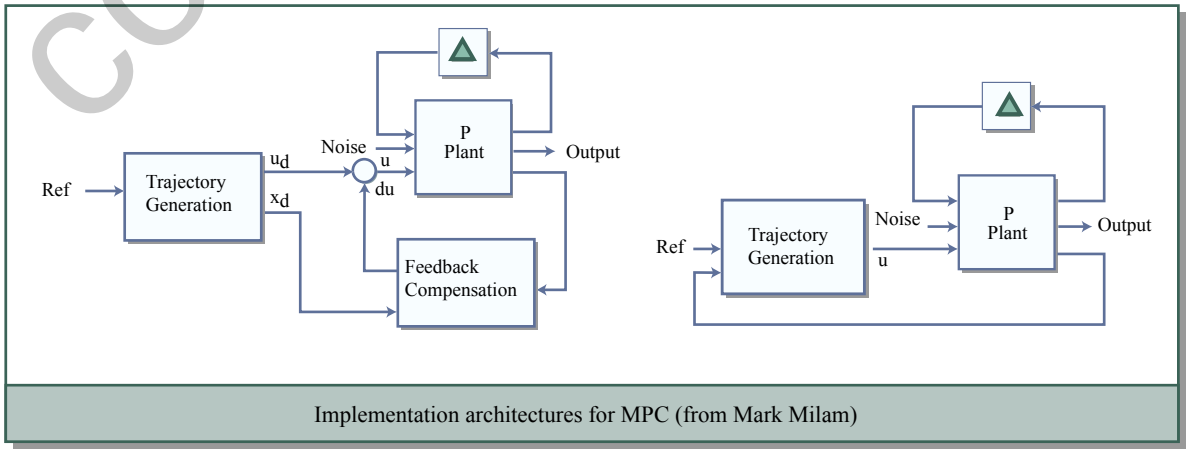
Figure by MIT OpenCourseWare.

- Note that the control algorithm is based on numerically solving an optimization problem at each step
  - Typically a constrained optimization

- Main advantage of MPC:
  - Explicitly accounts for system constraints.
    - ◇ Doesn't just design a controller to keep the system away from them.
  - Can easily handle nonlinear and time-varying plant dynamics, since the controller is explicitly a function of the model that can be modified in real-time (and plan time)

- Many commercial applications that date back to the early 1970's, see http://www.che.utexas.edu/~qin/cpcv/cpcv14.html
  - Much of this work was in process control - very nonlinear dynamics, but not particularly fast.

- As computer speed has increased, there has been renewed interest in applying this approach to applications with faster time-scale: trajectory design for aerospace systems.



Implementation architectures for MPC (from Mark Milam)

Figure by MIT OpenCourseWare.

Basic Formulation

- Given a set of plant dynamics (assume linear for now)

$$\begin{aligned}\mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k)\\ \mathbf{z}(k) &= C\mathbf{x}(k)\end{aligned}$$

  and a cost function

$$J = \sum_{j=0}^{N} \{\|\mathbf{z}(k+j|k)\|_{R_{zz}} + \|\mathbf{u}(k+j|k)\|_{R_{uu}}\} + F(\mathbf{x}(k+N|k))$$

  – $\|\mathbf{z}(k+j|k)\|_{R_{xx}}$ is just a short hand for a weighted norm of the state, and to be consistent with earlier work, would take

$$\|\mathbf{z}(k+j|k)\|_{R_{zz}} = \mathbf{z}(k+j|k)^T R_{zz} \mathbf{z}(k+j|k)$$

  – $F(\mathbf{x}(k+N|k))$ is a terminal cost function

- Note that if $N \to \infty$, and there are no additional constraints on $\mathbf{z}$ or $\mathbf{u}$, then this is just the discrete LQR problem solved on page 3–14.
  – Note that the original LQR result could have been written as just an input control sequence (**feedforward**), but we choose to write it as a linear state **feedback**.
  – In the nominal case, there is no difference between these two implementation approaches (feedforward and feedback)
  – But with modeling errors and disturbances, the state feedback form is much less sensitive.

  $\Rightarrow$ This is the main reason for using feedback.

- **Issue:** When limits on $\mathbf{x}$ and $\mathbf{u}$ are added, we can no longer find the general solution in analytic form $\Rightarrow$ must solve it numerically.

- However, solving for a very long input sequence:
  - Does not make sense if one expects that the model is wrong and/or there are disturbances, because it is unlikely that the end of the plan will be implemented (a new one will be made by then)
  - Longer plans have more degrees of freedom and take much longer to compute.

- Typically design using a small $N \Rightarrow$ short plan that does not necessarily achieve all of the goals.
  - Classical hard question is how large should N be?
  - If plan doesn't reach the goal, then must develop an estimate of the remaining **cost-to-go**

- Typical problem statement: for finite N ($F = 0$)

$$\begin{aligned} \min_u J &= \sum_{j=0}^{N} \{\|\mathbf{z}(k+j|k)\|_{R_{zz}} + \|\mathbf{u}(k+j|k)\|_{R_{uu}}\} \\ \text{s.t.} \quad \mathbf{x}(k+j+1|k) &= A\mathbf{x}(k+j|k) + B\mathbf{u}(k+j|k) \\ \mathbf{x}(k|k) &\equiv \mathbf{x}(k) \\ \mathbf{z}(k+j|k) &= C\mathbf{x}(k+j|k) \\ \text{and} \quad |\mathbf{u}(k+j|k)| &\leq u_m \end{aligned}$$

- Consider converting this into a more standard optimization problem.

$$\begin{aligned}
\mathbf{z}(k|k) &= C\mathbf{x}(k|k) \\
\mathbf{z}(k+1|k) &= C\mathbf{x}(k+1|k) = C(A\mathbf{x}(k|k) + B\mathbf{u}(k|k)) \\
&= CA\mathbf{x}(k|k) + CB\mathbf{u}(k|k) \\
\mathbf{z}(k+2|k) &= C\mathbf{x}(k+2|k) \\
&= C(A\mathbf{x}(k+1|k) + B\mathbf{u}(k+1|k)) \\
&= CA(A\mathbf{x}(k|k) + B\mathbf{u}(k|k)) + CB\mathbf{u}(k+1|k) \\
&= CA^2\mathbf{x}(k|k) + CAB\mathbf{u}(k|k) + CB\mathbf{u}(k+1|k) \\
&\vdots \\
\mathbf{z}(k+N|k) &= CA^N\mathbf{x}(k|k) + CA^{N-1}B\mathbf{u}(k|k) + \cdots \\
&\quad + CB\mathbf{u}(k+(N-1)|k)
\end{aligned}$$

- Combine these equations into the following:

$$\begin{bmatrix} \mathbf{z}(k|k) \\ \mathbf{z}(k+1|k) \\ \mathbf{z}(k+2|k) \\ \vdots \\ \mathbf{z}(k+N|k) \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} \mathbf{x}(k|k)$$

$$+ \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ CB & 0 & 0 & & 0 \\ CAB & CB & 0 & & 0 \\ \vdots & & & & \\ CA^{N-1}B & CA^{N-2}B & CA^{N-3}B & \cdots & CB \end{bmatrix} \begin{bmatrix} \mathbf{u}(k|k) \\ \mathbf{u}(k+1|k) \\ \vdots \\ \mathbf{u}(k+N-1|k) \end{bmatrix}$$

- Now define

$$Z(k) \equiv \begin{bmatrix} \mathbf{z}(k|k) \\ \vdots \\ \mathbf{z}(k+N|k) \end{bmatrix} \quad U(k) \equiv \begin{bmatrix} \mathbf{u}(k|k) \\ \vdots \\ \mathbf{u}(k+N-1|k) \end{bmatrix}$$

then, with $\mathbf{x}(k|k) = \mathbf{x}(k)$

$$Z(k) = G\mathbf{x}(k) + HU(k)$$

- Note that

$$\sum_{j=0}^{N} \mathbf{z}(k+j|k)^T R_{zz} \mathbf{z}(k+j|k) = Z(k)^T W_1 Z(k)$$

with an obvious definition of the weighting matrix W_1

- Thus

$$\begin{aligned} Z(k)^T W_1 Z(k) + U(k)^T W_2 U(k) \\ &= (G\mathbf{x}(k) + HU(k))^T W_1 (G\mathbf{x}(k) + HU(k)) + U(k)^T W_2 U(k) \\ &= \mathbf{x}(k)^T H_1 \mathbf{x}(k) + H_2^T U(k) + \frac{1}{2} U(k)^T H_3 U(k) \end{aligned}$$

where

$$H_1 = G^T W_1 G, \quad H_2 = 2(\mathbf{x}(k)^T G^T W_1 H), \quad H_3 = 2(H^T W_1 H + W_2)$$

- Then the MPC problem can be written as:

$$\begin{aligned} \min_{U(k)} \tilde{J} &= H_2^T U(k) + \frac{1}{2} U(k)^T H_3 U(k) \\ \text{s.t.} \quad & \begin{bmatrix} I_N \\ -I_N \end{bmatrix} U(k) \leq u_m \end{aligned}$$

- **Key point:** the MPC problem is now in the form of a standard **quadratic program** for which standard and efficient codes exist.

```
QUADPROG Quadratic programming. %
X=QUADPROG(H,f,A,b) attempts to solve the %
quadratic programming problem:

min 0.5*x'*H*x + f'*x   subject to:  A*x <= b
 x

X=QUADPROG(H,f,A,b,Aeq,beq) solves the problem %
above while additionally satisfying the equality%
constraints Aeq*x = beq.
```

- Several Matlab toolboxes exist for testing these ideas
  - MPC toolbox by Morari and Ricker – extensive analysis and design tools.
  - MPCtools [32] enables some MPC simulation and is free www.control.lth.se/user/johan.akesson/mpctools/

[32] Johan Akesson: "MPCtools 1.0 - Reference Manual". Technical report ISRN LUTFD2/TFRT–7613–SE, Department of Automatic Control, Lund Institute of Technology, Sweden, January 2006.

MPC Observations

- Current form assumes that full state is available - can hookup with an estimator

- Current form assumes that we can sense and apply corresponding control immediately
  - With most control systems, that is usually a reasonably safe assumption
  - Given that we must re-run the optimization, probably need to account for this computational delay - different form of the discrete model - see F&P (chapter 2)

- If the constraints are not active, then the solution to the QP is that

$$U(K) = -H_3^{-1} H_2$$

which can be written as:

$$\begin{aligned} u(k|k) &= -\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} (H^T W_1 H + W_2)^{-1} H^T W_1 G \mathbf{x}(k) \\ &= -K\mathbf{x}(k) \end{aligned}$$

which is just a state feedback controller.

  - Can apply this gain to the system and check the eigenvalues.

- What can we say about the stability of MPC when the constraints are active? [33]
  - Depends a lot on the terminal cost and the terminal constraints.[34]

- Classic result:[35] Consider a MPC algorithm for a linear system with constraints. Assume that there are terminal constraints:
  - $\mathbf{x}(k+N|k) = 0$ for predicted state $\mathbf{x}$
  - $\mathbf{u}(k+N|k) = 0$ for computed future control u

  Then if the optimization problem is feasible at time k, $\mathbf{x} = 0$ is stable.

  Proof: Can use the performance index J as a Lyapunov function.
  - Assume there exists a feasible solution at time k and cost J_k
  - Can use that solution to develop a feasible candidate at time $k+1$, by simply adding $\mathbf{u}(k+N+1) = 0$ and $\mathbf{x}(k+N+1) = 0$.
  - **Key point:** can estimate the candidate controller performance

$$\begin{aligned}\tilde{J}_{k+1} &= J_k - \{\|\mathbf{z}(k|k)\|_{R_{zz}} + \|\mathbf{u}(k|k)\|_{R_{uu}}\} \\ &\leq J_k - \{\|\mathbf{z}(k|k)\|_{R_{zz}}\}\end{aligned}$$

  - This candidate is suboptimal for the MPC algorithm, hence J decreases even faster $J_{k+1} \leq \tilde{J}_{k+1}$
  - Which says that J decreases if the state cost is non-zero (observability assumptions) $\Rightarrow$ but J is lower bounded by zero.

- Mayne *et al.* [2000] provides excellent review of other strategies for proving stability – different terminal cost and constraint sets

[33] "Tutorial: model predictive control technology," Rawlings, J.B. American Control Conference, 1999. pp. 662-676

[34] Mayne, D.Q., J.B. Rawlings, C.V. Rao and P.O.M. Scokaert, "Constrained Model Predictive Control: Stability and Optimality," Automatica, 36, 789-814 (2000).

[35] A. Bemporad, L. Chisci, E. Mosca: "On the stabilizing property of SIORHC", Automatica, vol. 30, n. 12, pp. 2013-2015, 1994.

Example: Helicopter

- Consider a system similar to the Quansar helicopter[36]

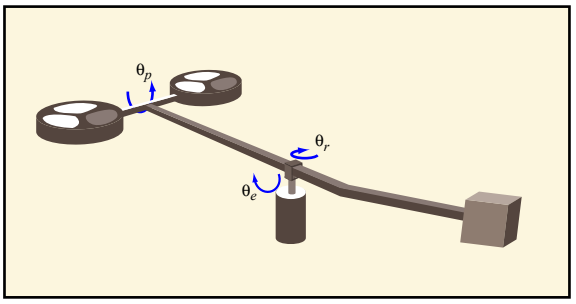


Figure by MIT OpenCourseWare.

- There are 2 control inputs – voltage to each fan V_f, V_b
- A simple dynamics model is that:

$$\begin{aligned} \ddot{\theta}_e &= K_1(V_f + V_b) - T_g/J_e \\ \ddot{\theta}_r &= -K_2 \sin(\theta_p) \\ \ddot{\theta}_p &= K_3(V_f - V_b) \end{aligned}$$

and there are physical limits on the elevation and pitch:

$$-0.5 \leq \theta_e \leq 0.6 \qquad -1 \leq \theta_p \leq 1$$

- Model can be linearized and then discretized $T_s = 0.2$sec.

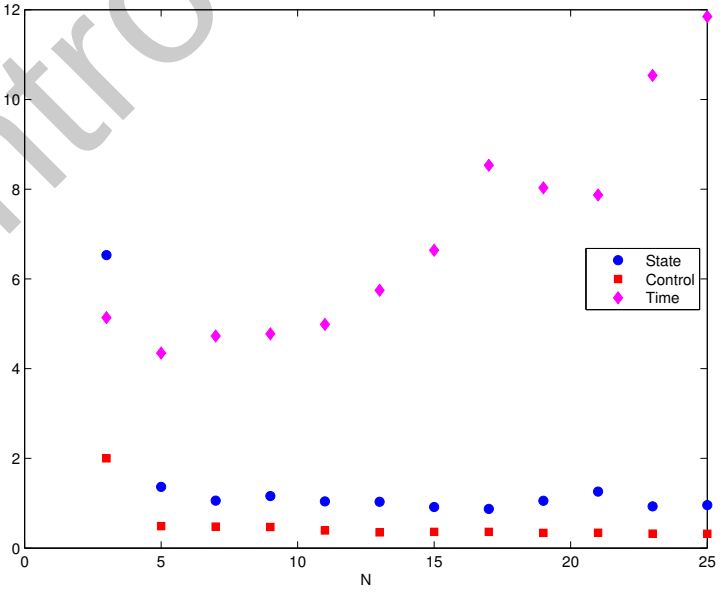


Figure 16.3: Response Summary

[36] ISSN 02805316 ISRN LUTFD2/TFRT- -7613- -SE MPCtools 1.0 Reference Manual Johan Akesson Department of Automatic Control Lund Institute of Technology January 2006

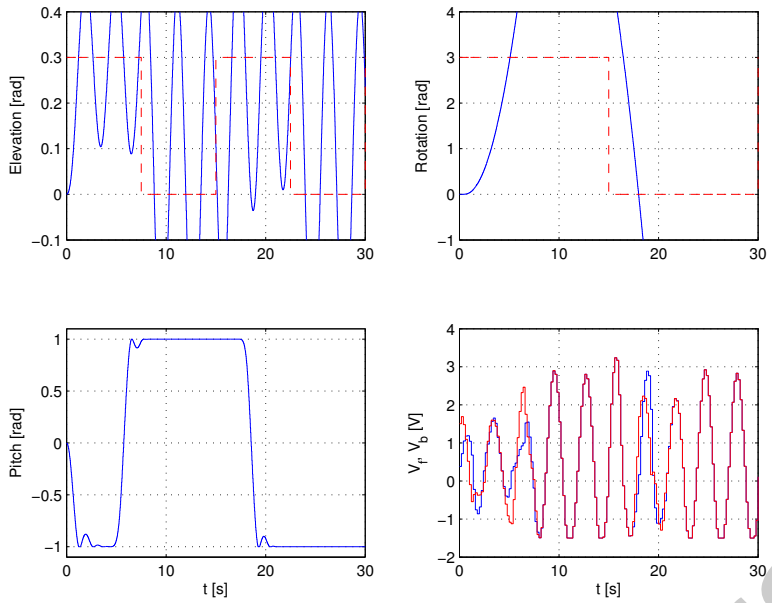


Figure 16.4: Response with $N = 3$

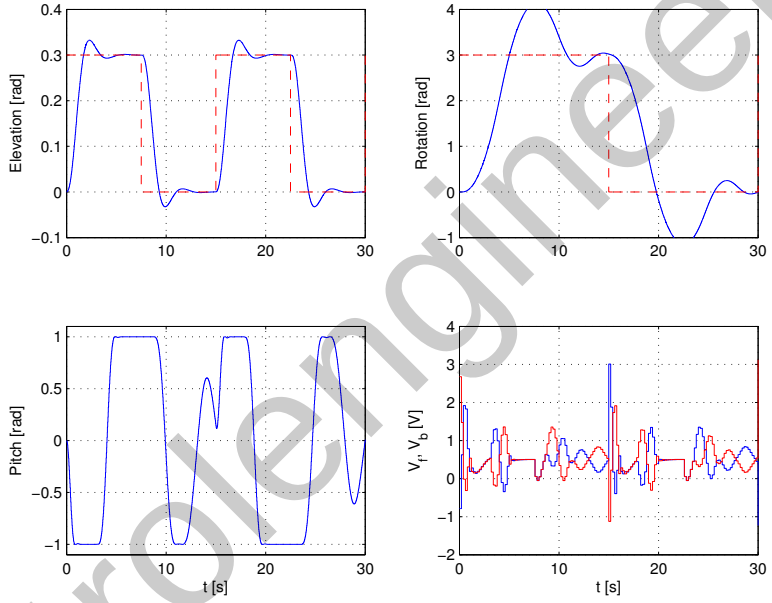


Figure 16.5: Response with $N = 10$

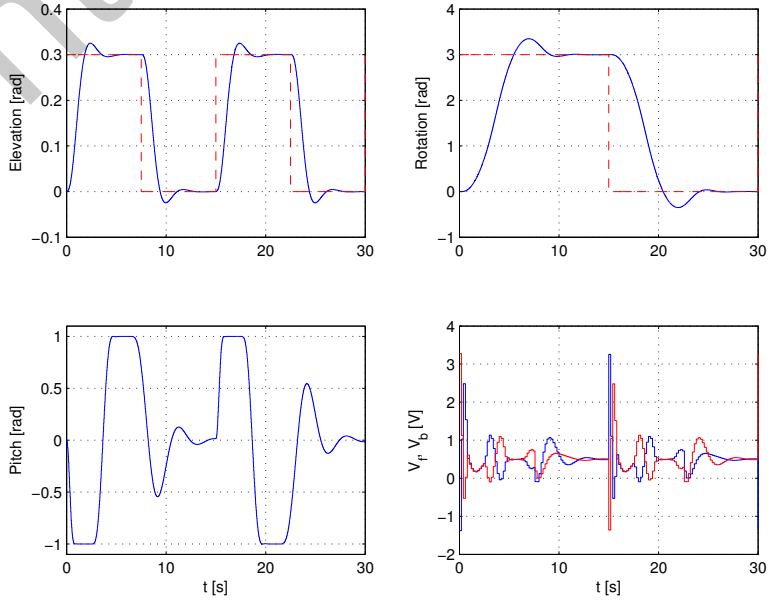


Figure 16.6: Response with $N = 25$