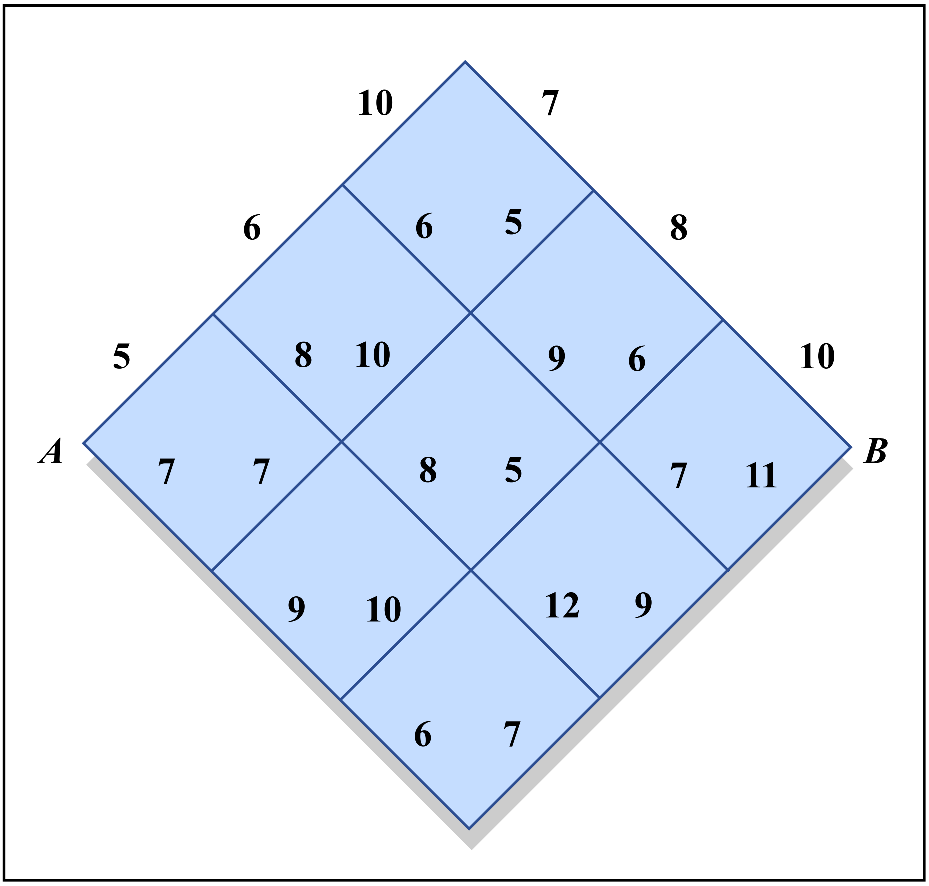
Dynamic Programming

- Principle of Optimality
- Dynamic Programming
- Discrete LQR

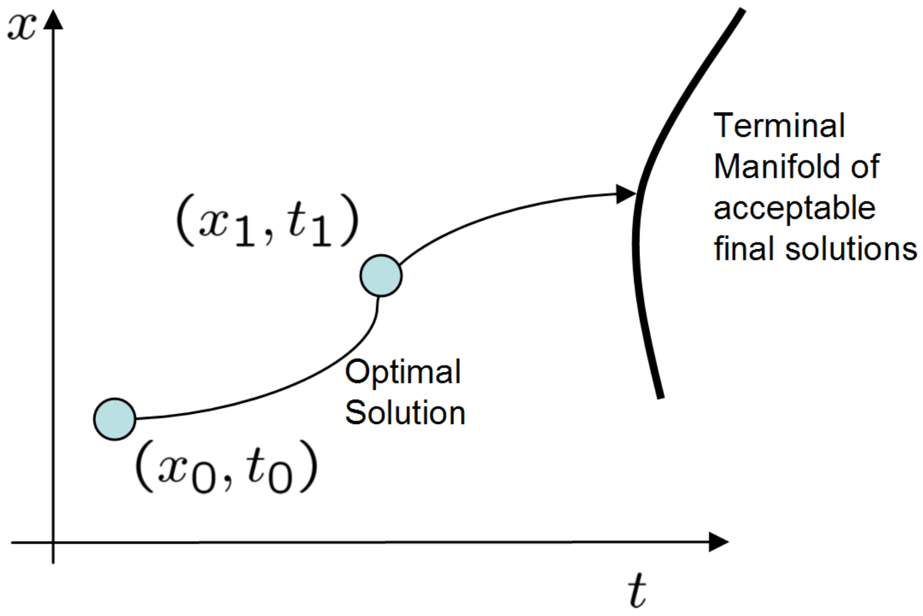


Dynamic Programming

- A central idea of control theory that is based on the

  Principle of Optimality:

  Suppose that 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.

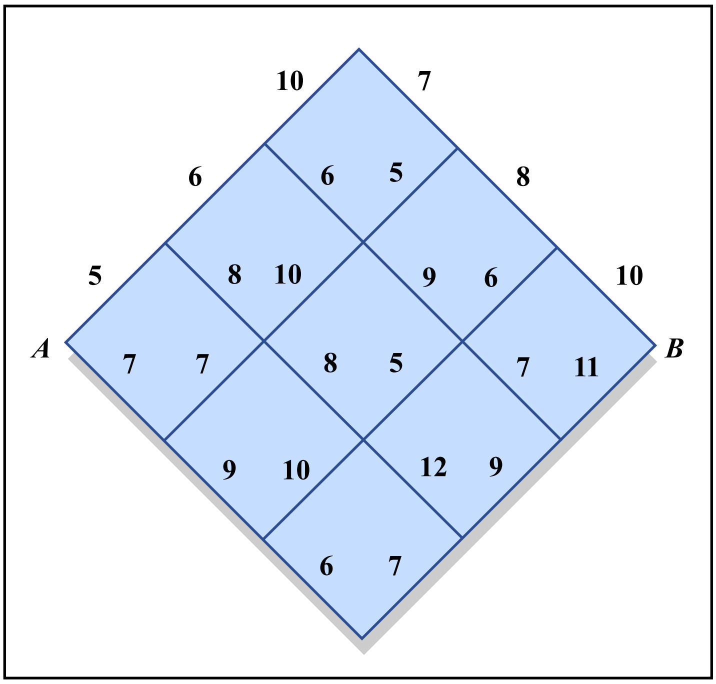


- Proof? What would the implications be if it was false?
- Principle leads to:
  - Numerical solution procedure called **Dynamic Programming** for solving multi-stage decision making problems.
  - Theoretical results on the structure of the resulting control law.

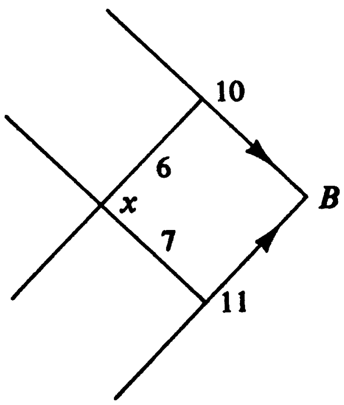
  - Dynamic Programming (Paperback) by Richard Bellman (Dover)
  - Dynamic Programming and Optimal Control (Vol 1 and 2) by D. P. Bertsekas

Classical Examples

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



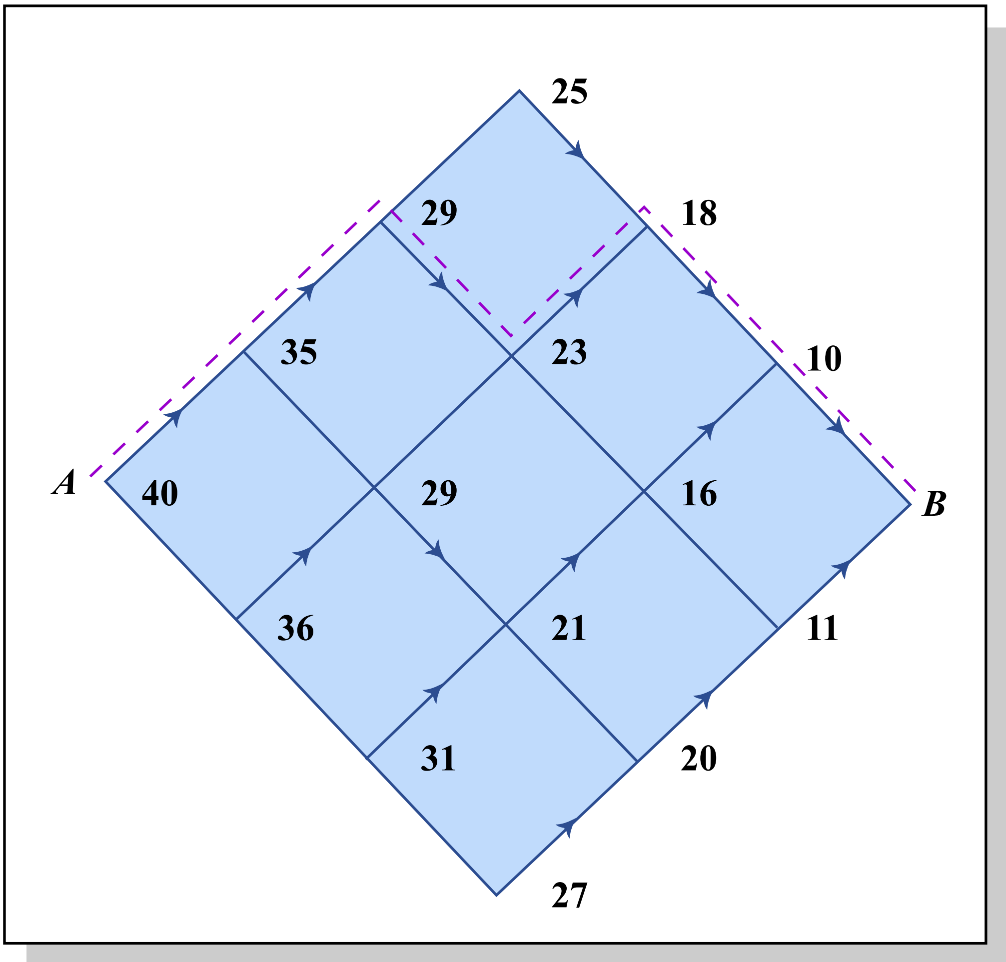
- Goal is to travel from A to B in the shortest time possible
  - Travel times for each leg are shown in the figure
  - There are 20 options to get from A to B – could evaluate each and compute travel time, but that would be pretty tedious
- **Alternative approach:** Start at B and work backwards, invoking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



- Consider the travel time from point x
  - Can go up and then down $6 + 10 = 16$
  - Or can go down and then up $7 + 11 = 18$

- Clear the best option from x is to go up and then down, with a time of 16
- From principle of optimality, this is the best way to get to B for any path that passes through x.

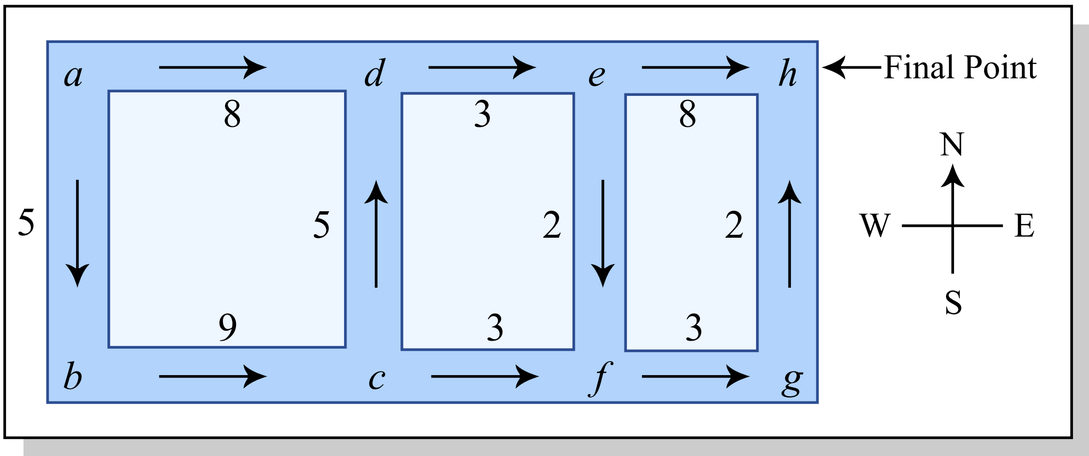
- Repeat the process for all other points, until finally get to initial point $\Rightarrow$ shortest path can be traced by moving in the directions of the arrows.



- **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 in this case, but this scales up immensely for larger problems.

Example 2

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



  – 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 $J^\star_{dh} = 10$

  – 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

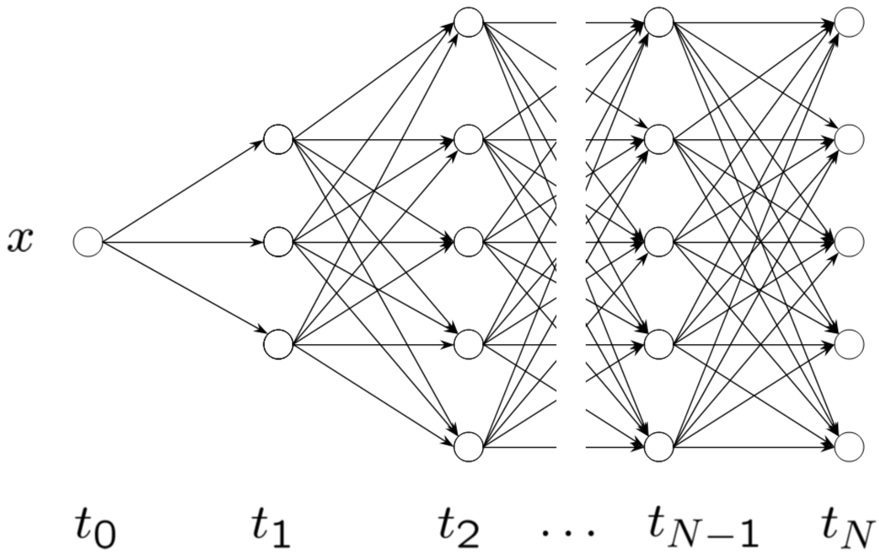


Figure 1: Classic picture of discrete time/quantized space grid with the linkages possible through the control commands. Again, it is hard to evaluate all options moving forward through the grid, but we can work backwards and use the principle of optimality to reduce this load.

Classic Control Problem

- Consider the problem of minimizing:

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

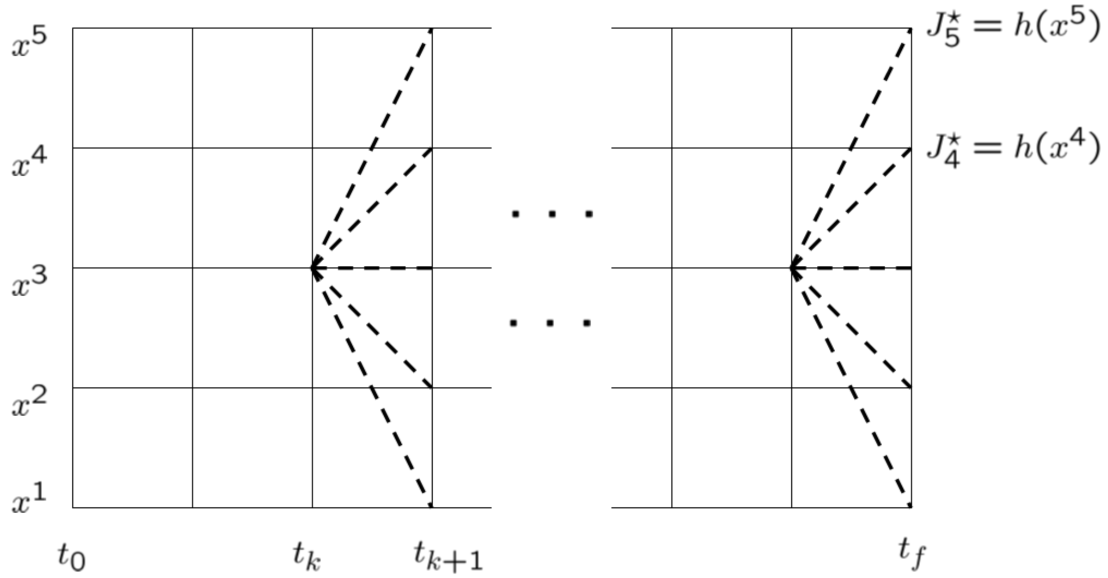
subject to

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

  – 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,. . . ,x^5

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

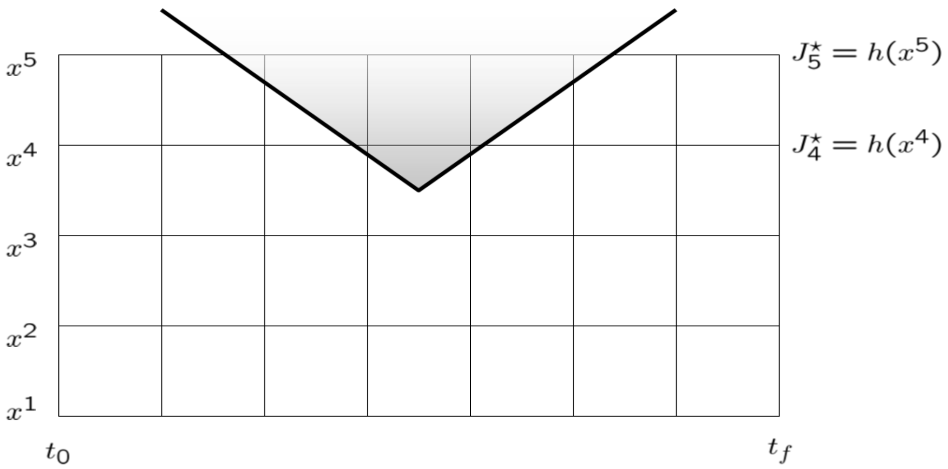


- **Step 2**: back up 1 step in time and consider all possible ways of completing the problem.
  – To evaluate the cost of a control action, must approximate the integral in the cost.

- 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:
$$x_{k+1}^j \approx x_k^i + a(x_k^i, u_k^{ij}, t_k)\Delta t$$
$$\Rightarrow a(x_k^i, u_k^{ij}, t_k) = \frac{x_{k+1}^j - x_k^i}{\Delta t}$$
    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
$$J^\star(x_k^i, t_k) = \min_{x_{k+1}^j} \left[\Delta J(x_k^i, x_{k+1}^j) + J^\star(x_{k+1}^j) \right]$$
  - Output is best x_k^i to pick because it gives the lowest cost and a clear plan to follow to achieve this best cost.
- Then work backwards in time until you reach x_{t_0}, when only 1 value of x is allowed because of the given initial condition.

Other Considerations

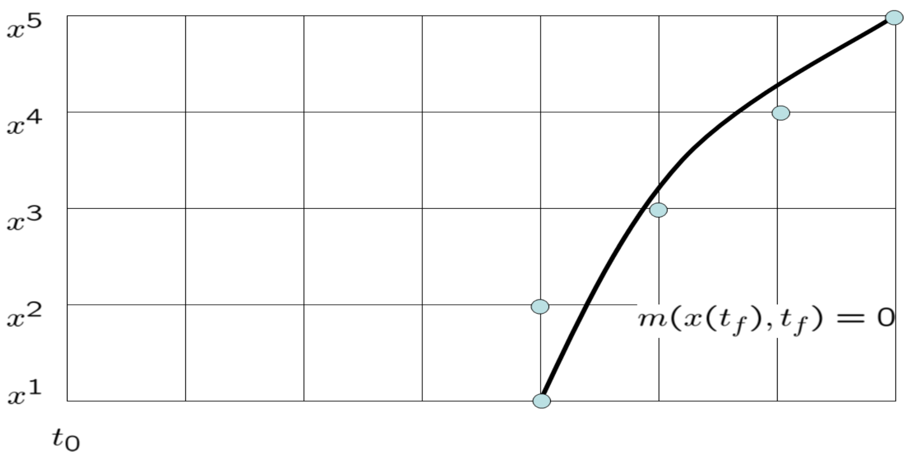
- With bounds on the control, then certain state transitions might not be allowed from 1 time-step to the next.
- With constraints on the state, certain values of $x(t)$ might not be allowed at certain times t.



- 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 $m(x(t_f), t_f) = 0$.

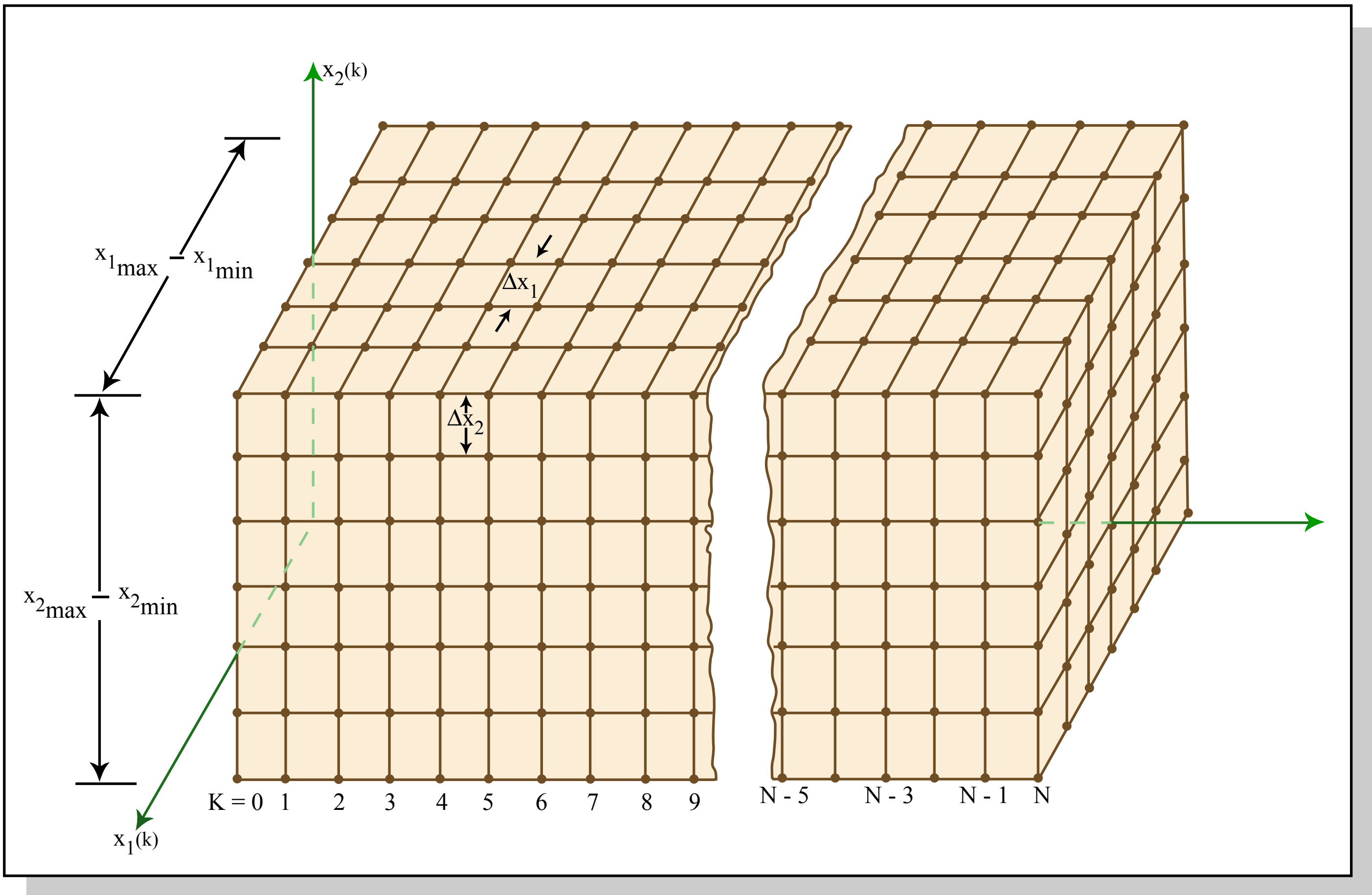


  – Gives group of points that (approximately) satisfy the terminal constraint and can evaluate the cost for each, and work backwards from there.

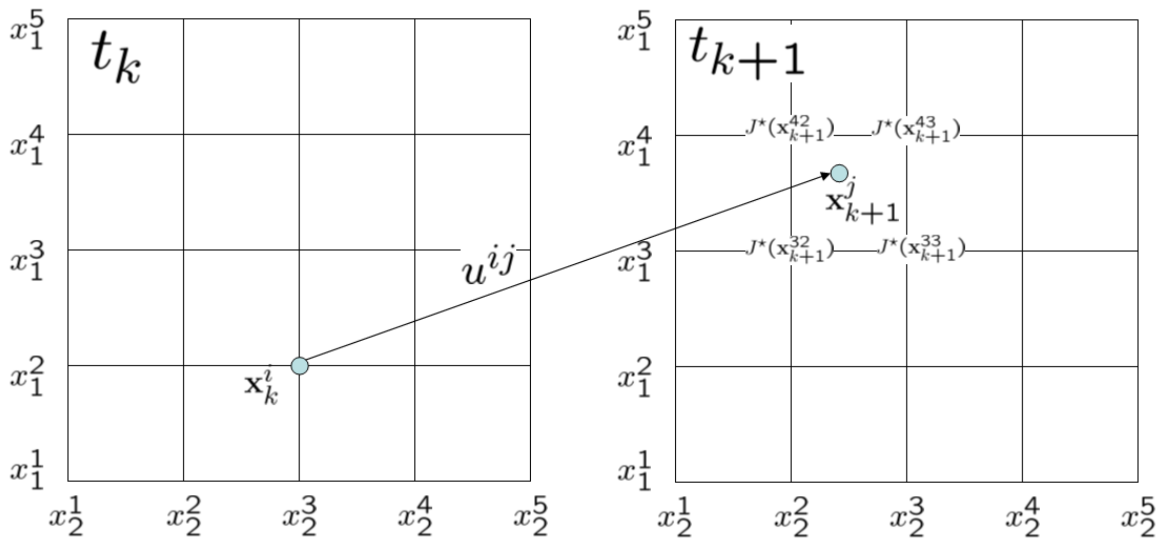
- Previous formulation picked x's and used those to determine the u's.
  - For more general problems, would better off picking the u's and using those to determine the x's:

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

  - To do this, must quantize the control inputs as well.
  - But then likely that the 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 $\Rightarrow$ more on this later.
- This process can be extended 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:



- At any time t_k, have a two dimensional array of grid points.
- Must find the control that moves the state from a point on one grid to a point on another.
  - Alternatively, could just 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 page)



- 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 this approach is how badly it scales.
  - Given N_t points in time and N_x quantized states of dimension n
  - The number of points to consider is $N = N_t N_x^n$
    $\Rightarrow$ "**the curse of dimensionality**" – R. Bellman, Dynamic Programming (1957) now available 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 $0 \leq x \leq 1.5$ and $-1 \leq u \leq 1$

- 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 the exact calculation from the previous section – the 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.
  - Assume that $0 \leq x(k) \leq 1.5$, and x is quantized into four possible values $x_k \in \{0, 0.5, 1.0, 1.5\}$
  - Assume that control is bounded $|u(k)| \leq 1$ and is quantized into five possible values: $u_k \in \{-1, -0.5, 0, 0.5, 1\}$

- Start – evaluate cost associate with all possible terminal states

x_2^j	$J_2^\star = h(x_2^j)$
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	**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

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

- Take transpose and set equal to 0, yields

$$\left[R_{N-1} + B_{N-1}^T H B_{N-1}\right]\mathbf{u}_{N-1} + B_{N-1}^T H A_{N-1}\mathbf{x}_{N-1} = 0$$

- 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 $N-1$:

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

  - Furthermore, can show that

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

    so that the stationary point is a minimum

- With this control decision, take another look at

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

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

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

- 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

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

- 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:

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

  cycle through steps (ii) and (iii) from $N-1$ to 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

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

- 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.

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

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

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

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

  – 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
  - Time invariant problem (LTI) – i.e., A, B, Q, R are constant
  - System (A, B) stabilizable – uncontrollable modes are stable.
  - $H = 0$ (not completely necessary)

- Then as $N \rightarrow \infty$, the recursion for P tends to a constant solution which is bounded and satisfies (set $P_k \equiv P_{k+1}$)

$$P = Q + A^T \left\{ P - PB \left[R + B^T PB \right]^{-1} B^T P \right\} A$$

  with $P \geq 0$ at least.
  - Famous discrete form of the **Algebraic Riccati Equation**
  - Typically hard to solve analytically, but easy to solve numerically.

- Note that if $Q = C^T C$ and (A, C) detectable,
  - Equivalent to having measurements of the form $\mathbf{y} = C\mathbf{x}$ and a state penalty

$$\mathbf{y}^T\mathbf{y} = \mathbf{x}^T C^T C \mathbf{x} = \mathbf{x} Q \mathbf{x}$$

  Then
  - The recursion for P has a **unique** steady state solution that is **positive definite**.
  - Closed-loop system $\mathbf{x}_{k+1} = (A+BF)\mathbf{x}_k$ **asymptotically stable**.

- Reason for this constraint:
  - If a system mode is not penalized in the "state cost", then it would not make sense in the optimal solution to exert effort to control that mode $\Rightarrow$ pole associated with the mode would not be changed, and if it were in the RHP, the closed-loop system would be unstable.

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
$$J = \frac{1}{4}x(5)^2 + \frac{1}{2}\sum_{k=0}^{4}[x_k^2 + u_k^2]$$
so that $Q = R = 1,\ H = 1/2$
- Note that this is a discrete system, and the rules for stability are different – need $|\lambda_i(A + BF)| < 1$.
  – Thus open loop system is marginally stable, and a gain $-1 < F < 0$ will stabilize the system.

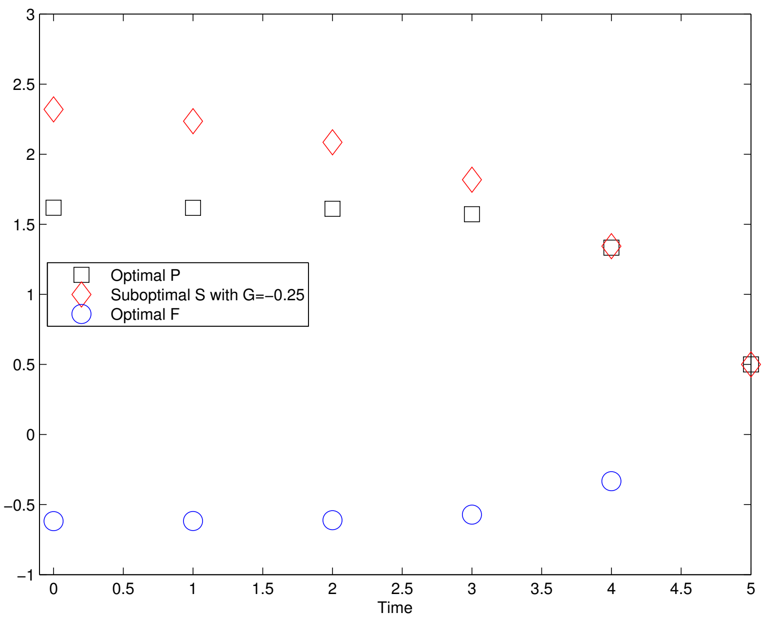


Figure 2: discrete LQR comparison to constant gain, $G = -0.25$

- 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

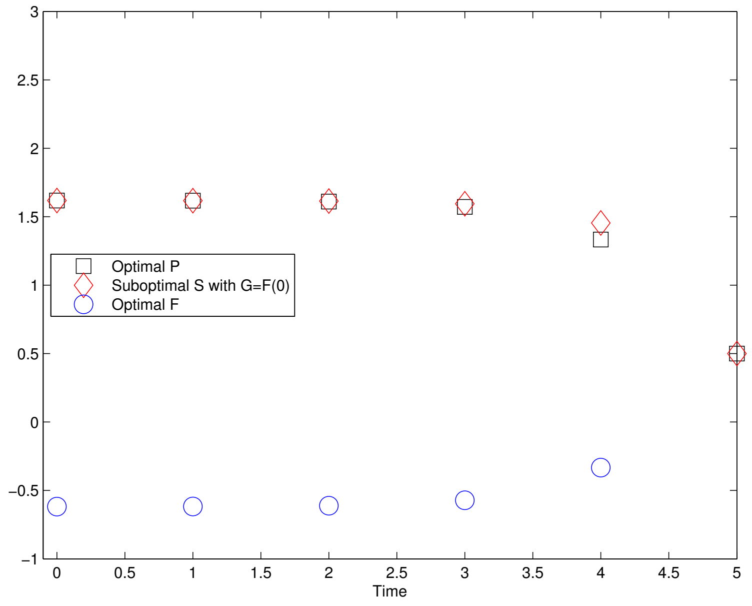


Figure 3: discrete LQR comparison to constant gain, $G = F(0)$

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

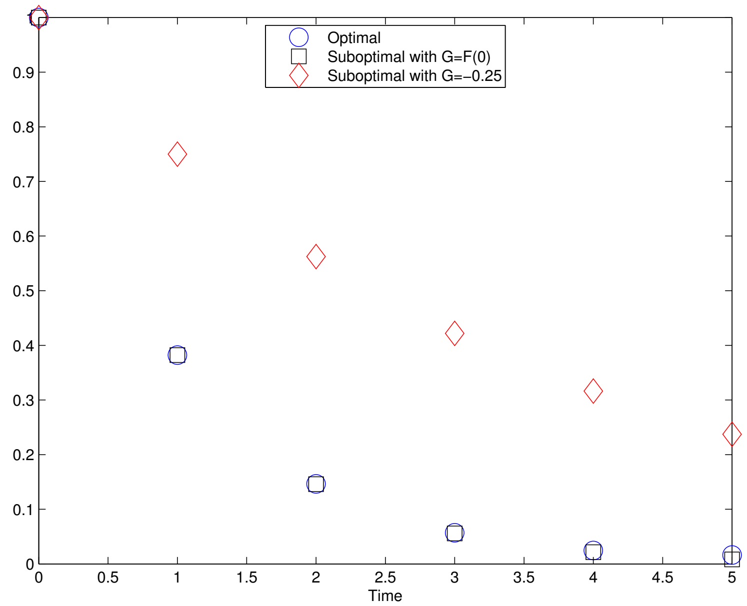


Figure 4: State response comparison

- State response consistent

Discrete scalar LQR

```
% integrator system
clear all
A=1;B=1;
Q=1;R=1;H=0.5;

N=5;
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)
plot(time,P,'ks',time,S,'rd',time(1:N),F,'bo','MarkerSize',12)
legend('Optimal P','Suboptimal S with G=F(0)','Optimal F','Location','West')
xlabel('Time')
axis([-.1 5 -1 3] )
print -depsc integ.eps;jpdf('integ')

% what if we used a fixed gain of F=-0.5, 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)
legend('Optimal P','Suboptimal S with G=-0.25','Optimal F','Location','West')
axis([-.1 5 -1 3] )
xlabel('Time')
print -depsc integ2.eps;jpdf('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);
    xs2(k+1)=(A+B*(-0.25))*xs2(k);
end
figure(3)
plot(time,xo,'bo','MarkerSize',12)
hold on
plot(time,xs1,'ks','MarkerSize',12)
plot(time,xs2,'rd','MarkerSize',12)
hold off
legend('Optimal','Suboptimal with G=F(0)','Suboptimal with G=-0.25','Location','North')
%axis([-.1 5 -1 3] )
xlabel('Time')
print -depsc integ3.eps;jpdf('integ3')
```

Gain Insights

- Note that from Eq. 1 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
$$F_{N-1} = -\frac{B_{N-1}^T P_N A_{N-1}}{R_{N-1} + B_{N-1}^2 P_N}$$

- So if there is a high weighting on the terminal state, $H \to \infty$ and thus P_N is large. Thus
$$F_{N-1} \to -\frac{B_{N-1}^T P_N A_{N-1}}{B_{N-1}^2 P_N} \to \frac{-A}{B}$$
therefore
$$x_N = (A + BF)X_{N-1} = (A + B\frac{-A}{B})X_{N-1} = 0$$
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
$$F_{N-1} \to -\frac{B_{N-1}^T P_N A_{N-1}}{B_{N-1}^2 P_N} \to \frac{-A}{B}$$
and $x_N = 0$ as well.
  - State penalized, but control isn't, so 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 not limits on the control effort.