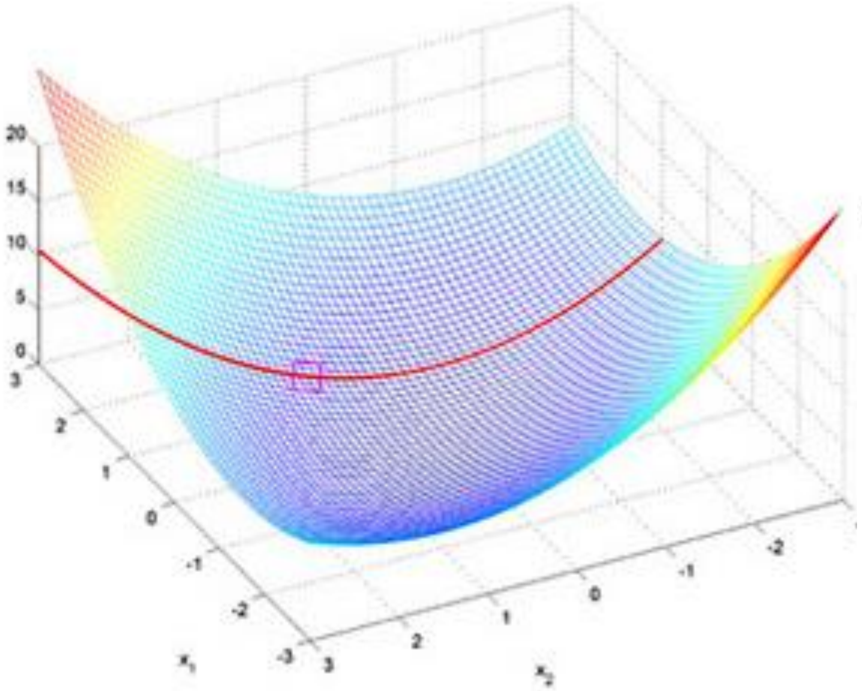
MEGR 7090/8090: Advanced Optimal Control

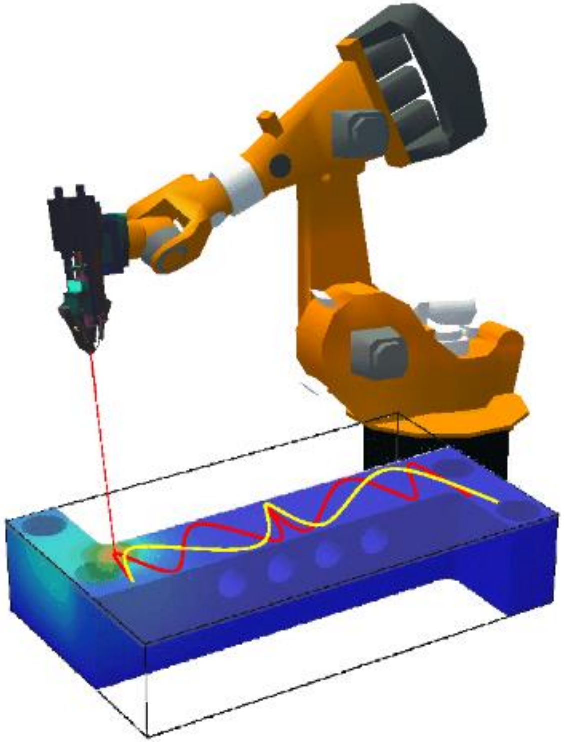
UNC CHARLOTTE



$$\boxed{V_n(\mathbf{x}_n) = \min_{\{\mathbf{u}_n,\mathbf{u}_{n+1},\cdots,\mathbf{u}_{N-1}\}} \left[\frac{1}{2}\sum_{k=n}^{N-1}\left(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k\right) + \frac{1}{2}\mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N\right]}$$

$$\begin{aligned}
V_n(\mathbf{x}_n) &= \min_{\{\mathbf{u}_n,\mathbf{u}_{n+1},\cdots,\mathbf{u}_{N-1}\}} \left[\frac{1}{2}\sum_{k=n}^{N-1}\left(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k\right) + \frac{1}{2}\mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N\right] \\
&= \min_{\mathbf{u}_n}\left[\frac{1}{2}\left(\mathbf{x}_n^T\mathbf{Q}_n\mathbf{x}_n + \mathbf{u}_n^T\mathbf{R}\mathbf{u}_n\right) + \underbrace{\min_{\{\mathbf{u}_{n+1},\cdots,\mathbf{u}_{N-1}\}}\left[\frac{1}{2}\sum_{k=n+1}^{N-1}\left(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k\right) + \frac{1}{2}\mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N\right]}_{V_{n+1}(\mathbf{x}_{n+1})}\right] \\
&= \min_{\mathbf{u}_n}\left[\frac{1}{2}\left(\mathbf{x}_n^T\mathbf{Q}_n\mathbf{x}_n + \mathbf{u}_n^T\mathbf{R}\mathbf{u}_n\right) + V_{n+1}(\mathbf{x}_{n+1})\right]
\end{aligned}$$

$$\boxed{V_n(\mathbf{x}_n) = \min_{\mathbf{u}_n}\left[\frac{1}{2}\left(\mathbf{x}_n^T\mathbf{Q}_n\mathbf{x}_n + \mathbf{u}_n^T\mathbf{R}\mathbf{u}_n\right) + V_{n+1}(\mathbf{x}_{n+1})\right]}$$



Lecture 4

August 31, 2017

Optimal Control – General Discrete-Time Framework

Whether the control trajectory is optimized offline or online, every discrete-time optimal control problem will involve the following general framework:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} J(\mathbf{u};\mathbf{x}(0)) \quad \text{where} \quad \underbrace{J(\mathbf{u};\mathbf{x}(0))}_{\text{Total cost}} = \sum_{i=0}^{N-1} \underbrace{g(\mathbf{x}(i),\mathbf{u}(i))}_{\text{Stage cost}} + \underbrace{h(\mathbf{x}(N))}_{\text{Terminal cost}}$$

Subject to:

$$\mathbf{x}(i+1) = f(\mathbf{x}(i),\mathbf{u}(i))$$
$$\mathbf{u}(i) \in U, i = 0 \ldots N-1$$
$$\mathbf{x}(i) \in X, i = 0 \ldots N-1$$
$$\mathbf{x}(N) \in X_f$$

Note: $\mathbf{u} = [\mathbf{u}(0) \quad \ldots \quad \mathbf{u}(N-1)]^T$…Now, what is the dimension of $\mathbf{u}$?

Key benefit of discrete-time representations for optimal control: The optimization problem is reduced to a finite-dimensional optimization!

Design Optimization vs. Optimal Control

UNC CHARLOTTE

Design optimization framework:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} J(\mathbf{p})$$ where $\mathbf{p}$ = vector of design parameters and J is a **static function** of those design parameters

Subject to: $\mathbf{p} \in P$

Control optimization framework:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} J(\mathbf{u};\mathbf{x}(0))$$ where $$J\big(\mathbf{u};\mathbf{x}(0)\big) = \sum_{i=0}^{N-1} g(\mathbf{x}(i),\mathbf{u}(i)) + h(\mathbf{x}(N))$$

Subject to:

$$\begin{aligned}
&\mathbf{x}(i+1) = f(\mathbf{x}(i),\mathbf{u}(i))\\
&\mathbf{u}(i) \in U, i = 0 \ldots N-1\\
&\mathbf{x}(i) \in X, i = 0 \ldots N-1\\
&\mathbf{x}(N) \in X_f
\end{aligned}$$

Key point: The Papalambros textbook ***only*** addresses design optimization – however, the discrete-time, finite horizon (finite N) control optimization can be shown to be ***equivalent*** to the design optimization problem

Design Optimization vs. Optimal Control

UNC CHARLOTTE

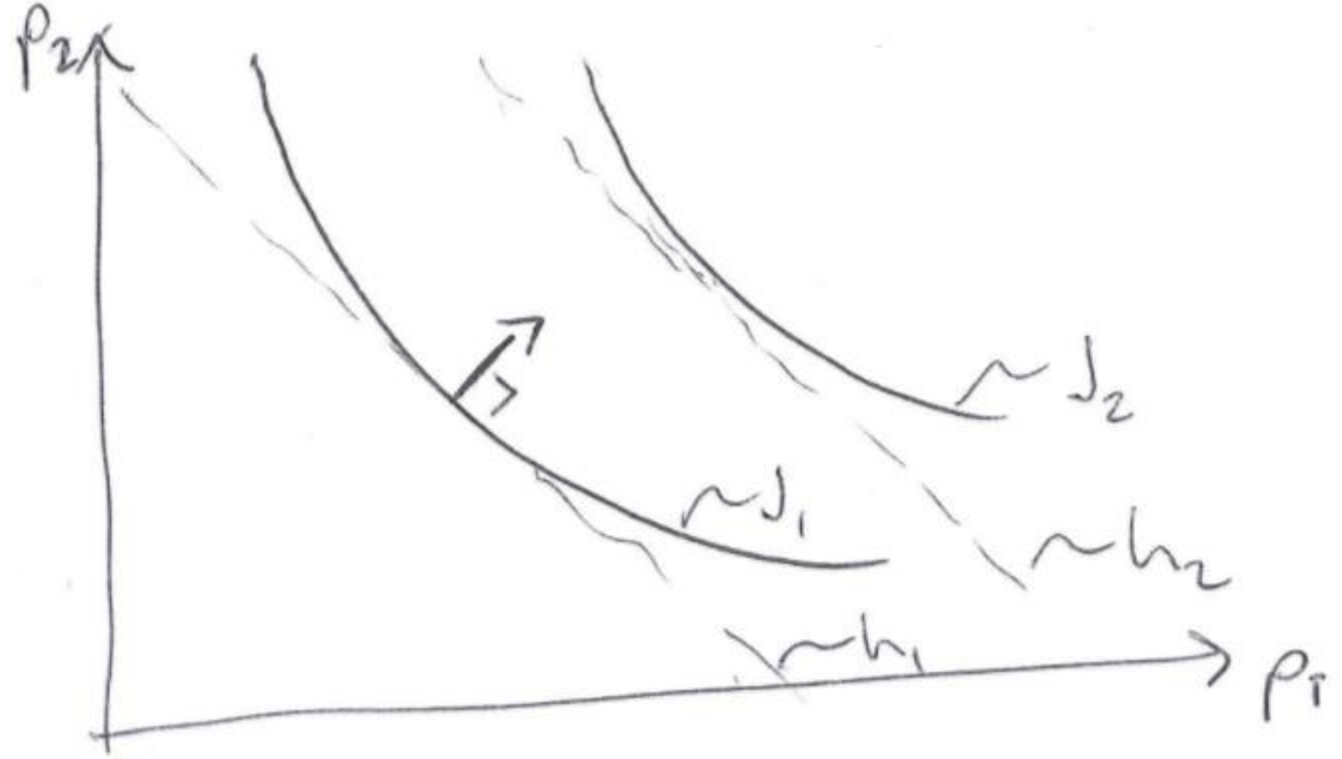
In order to use design optimization tools to optimize discrete-time control sequences, we need to "get rid of":

1) The dynamic model
2) All state constraints

Note: "Get rid of" is in quotes above, since we don't ***really*** get rid of the dynamics – after all, that would change the whole problem. Instead, we use the dynamics to write the states at step 1, 2, 3, … in terms of the control sequence and initial state ***only***. In that respect, we are rolling the dynamics into the objective function in order to eliminate dependence of the objective function on the state sequence (other than the initial state). Same idea for the state constraints.

Design Optimization Example – Conceptual Sketch

At the optimal design, the gradient of the objective function must be **aligned** with the gradient of the constraint function. This means that the level surfaces (curves in the 2D diagram at the right) must be **tangent** at the optimum. This is the basis for the theory behind **Lagrange multipliers** (which we'll learn much more about in later lectures).



Design Optimization Example – A Simple Box

Consider a box with length, width, and depth given by L, W, and D, respectively. Choose L, W, and D such that the volume is maximized, subject to the constraint that the total surface area is equal to 6 m^2.

Mathematical formulation: $\mathbf{p} = [L \quad W \quad D]^T$

$$J(\mathbf{p}) = -LWD$$

Constraint: $2(LW + LD + WD) = 6$ i.e., $P = \{\mathbf{p}: LW + LD + WD = 3\}$

Exercise in class: Using basic multivariable calculus and Lagrange multipliers, compute the optimal values of L, W, and D

Design Optimization Example – A Simple Box

$$p = [L \quad W \quad D]^T \qquad J(p) = -LWD$$

Constraint: $\underbrace{LW + LD + WD}_{h(p)} = 3$

At optimum, $\nabla J + \lambda \nabla h = \underline{0}$

$$\nabla J = [-WD \quad -LD \quad -LW] \qquad \nabla h = [W+D \quad L+D \quad L+W]$$

$$\left.\begin{aligned} WD &= (W+D)\lambda \\ LD &= (L+D)\lambda \\ LW &= (L+W)\lambda \end{aligned}\right\} \quad \left.\begin{aligned} \frac{WD}{\lambda(W+D)} &= \frac{LD}{\lambda(L+D)} \Rightarrow L = W \\ \frac{WD}{\lambda(W+D)} &= \frac{LW}{\lambda(L+W)} \Rightarrow L = D \end{aligned}\right\} \begin{aligned} L &= W \\ &= D \end{aligned}$$

$$LW + LD + WD = 3$$

$$\Rightarrow 3L^2 = 3 \Rightarrow L = 1 = W = D$$

$$\lambda = 1/2$$

A Simple Control Optimization Example

Consider the following discrete-time system model:

$$x(k+1) = x(k) + u(k)$$

Objective: Minimize $J(\mathbf{u}; x(0)\ ) = \sum_{i=0}^{2} [x(i)^2 + u(i)^2]$

Subject to: $x(3) = 0$

Given: $x(0) = 10$

Cast the above problem in the form of the aforementioned ***design optimization*** problem:

- Write $J\big(\mathbf{u}; x(0)\big)$ explicitly as a function of the decision variable ($\mathbf{u}$)
- Convert the terminal constraint ($x(3)$) into a constraint on the decision variable, which should have the form $u \in U$…come up with a mathematical description of U

A Simple Control Optimization Example

Optimal control problem (reminder):

Minimize $$J(\mathbf{u}; x(0)) = \sum_{i=0}^{2} [x(i)^2 + u(i)^2]$$

Subject to: $x(3) = 0$

$x(k+1) = x(k) + u(k)$

Given: $x(0) = 10$

Writing the optimal control problem in a form that is analogous to the design optimization:

- The decision variable is $\mathbf{u}$, where $\mathbf{u} = [u(0) \quad u(1) \quad u(2)]^T$
- To get an explicit function for $J(\mathbf{u}; x(0))$ (which doesn't contain a summation or intermediate values of x), we write out a recursion:

$$\left.\begin{aligned} x(1) &= x(0) + u(0) \\ x(2) &= x(1) + u(1) = x(0) + u(0) + u(1) \\ x(3) &= x(2) + u(2) = x(0) + u(0) + u(1) + u(2) \end{aligned}\right\}$$

$$\begin{gathered} J(\mathbf{u}; x(0)) = 100 + (10 + u(0))^2 \\ +(10 + u(0) + u(1))^2 \\ +u(0)^2 + u(1)^2 + u(2)^2 \end{gathered}$$

A Simple Control Optimization Example

Optimal control problem (reminder):

Minimize $$J(\mathbf{u}; x(0)) = \sum_{i=0}^{2} [x(i)^2 + u(i)^2]$$

Subject to: $x(3) = 0$

$x(k+1) = x(k) + u(k)$

Given: $x(0) = 10$

Constraint set analogy:

- Remember: $x(3) = x(2) + u(2) = x(0) + u(0) + u(1) + u(2)$
- Therefore, the constraint can be specified entirely in terms of the decision variable as:

$$u(0) + u(1) + u(2) = -10$$

- This can be expressed in set form as:

$$\mathbf{u} \in U \text{ where } U = \{\mathbf{u}: u(0) + u(1) + u(2) = -10\}$$

A Simple Control Optimization Example

Model: $x(k+1) = x(k) + u(k)$

Obj : $J(\underline{u}; x(0)) = \sum_{i=0}^{2} \left[x(i)^2 + u(i)^2 \right]$

Constraints : $x(3) = 0$

Given : $x(0) = 10$

$$J(\underline{u}; x(0)) = x(0)^2 + u(0)^2 + x(1)^2 + u(1)^2 + x(2)^2 + u(2)^2$$

$$= 100 + u(0)^2 + [\underset{\underset{10}{\shortparallel}}{x(0)} + u(0)]^2 + u(1)^2 + [\underbrace{x(1) + u(1)}_{\underset{\underset{10}{\shortparallel}}{x(0)} + u(0) + u(1)}]^2 + u(2)^2$$

A Simple Control Optimization Example

$$= 100 + u(0)^2 + [10 + u(0)]^2 + u(1)^2 + [10 + u(0) + u(1)]^2 + u(2)^2$$

$$\begin{aligned} x(3) &= x(2) + u(2) \\ &= x(1) + u(1) + u(2) \\ &= x(0) + u(0) + u(1) + u(2) \\ &= 10 + u(0) + u(1) + u(2) \overset{\text{need}}{=} 0 \end{aligned}$$

Analogies Between Design Optimization and Discrete-Time Optimal Control

Based on our examples, the following table of analogies can be drawn between the design optimization problem and optimal control problem:

Feature	Design Optimization	Optimal Control
Decision variable	$\mathbf{p} = [L \quad W \quad D]^T$	$\mathbf{u} = [u(0) \quad u(1) \quad u(2)]^T$
Objective (cost) function	$J(\mathbf{p}) = -LWD$	$J(\mathbf{u}; x(0)) = x(0)^2 + (x(0) + u(0))^2$ $+(x(0) + u(0) + u(1))^2 + u(0)^2 + u(1)^2 + u(2)^2$
Constraints	$\mathbf{p} \in P$ where $P = \{\mathbf{p}: LW + DL +$	$\mathbf{u} \in U$ where $U = \{\mathbf{u}: u(1) + u(2) + u(3) =$
Given parameters	none	$x(0) = 10$

Writing an Objective Function Entirely in Terms of the Decision Variable – General Linear Case

Key point: The ability to draw analogies between design optimization and discrete-time optimal control ***hinges on the ability to express*** $J(\mathbf{u};\mathbf{x}(0)) = \sum_{i=0}^{N} f(\mathbf{x}(i),\mathbf{u}(i))$ ***entirely in terms of*** $\mathbf{u}$ and $x(0)$…this is ***always possible!*** …We just need to eliminate the $\mathbf{x}(i)$ terms in the summation.

For linear systems (dynamics given by $\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$), the following recursive relationship applies:

$$\mathbf{x}(1) = A\mathbf{x}(0) + B\mathbf{u}(0)$$

$$\mathbf{x}(2) = A\mathbf{x}(1) + B\mathbf{u}(1) = A^2\mathbf{x}(0) + AB\mathbf{u}(0) + B\mathbf{u}(1)$$

$$\mathbf{x}(3) = A\mathbf{x}(2) + B\mathbf{u}(2) = A^3\mathbf{x}(0) + A^2B\mathbf{u}(0) + AB\mathbf{u}(1) + B\mathbf{u}(2)$$

…

$$\Rightarrow \mathbf{x}(n) = A^n\mathbf{x}(0) + \sum_{i=0}^{n-1} A^{n-1-i}B\mathbf{u}(i)$$

Writing an Objective Function Entirely in Terms of the Decision Variable – General Nonlinear Case

Key point: The ability to draw analogies between design optimization and discrete-time optimal control ***hinges on the ability to express*** $J\big(\mathbf{u};\mathbf{x}(0)\big) = \sum_{i=0}^{N} f(\mathbf{x}(i), \mathbf{u}(i))$ ***entirely in terms of*** $\mathbf{u}$ and $\mathbf{x}(0)$…this is ***always possible!*** …We just need to eliminate the $\mathbf{x}(i)$ terms in the summation**.**

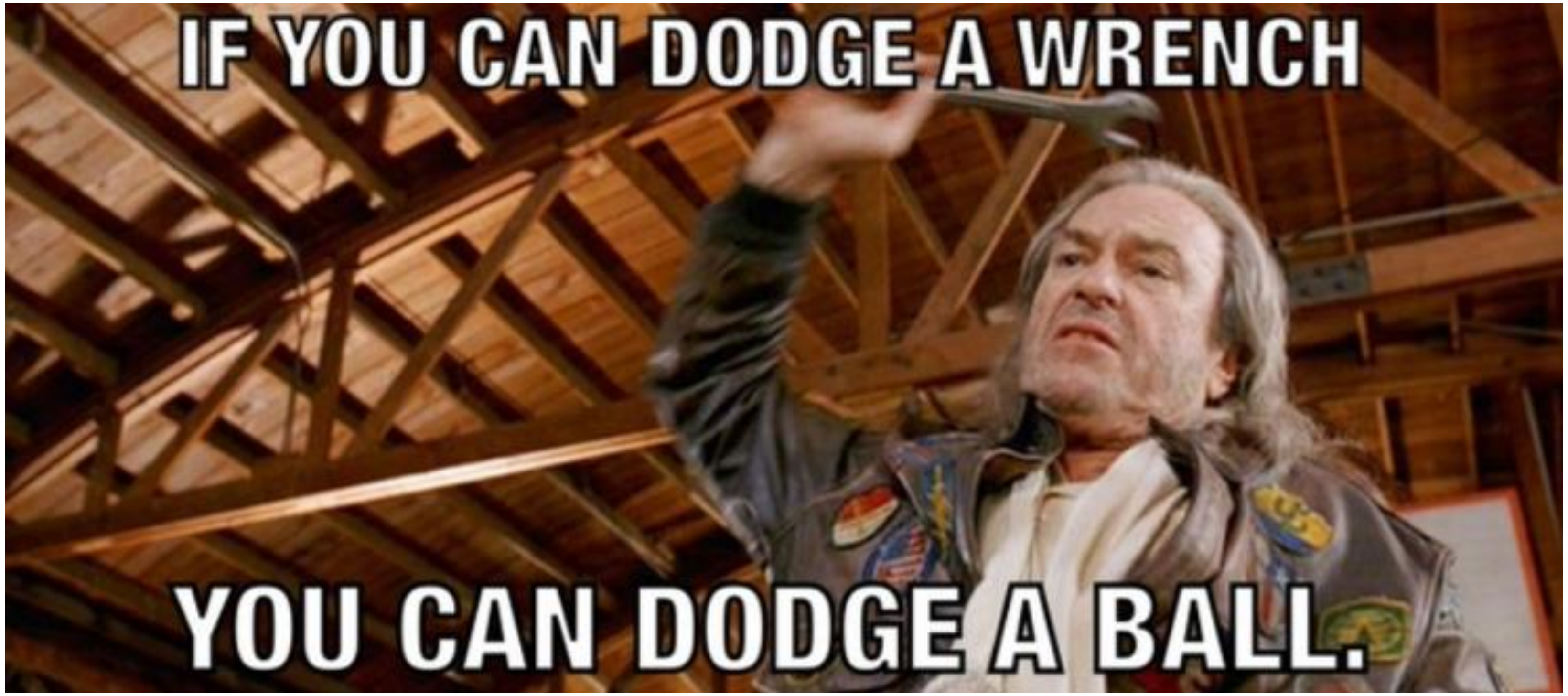
For nonlinear systems (dynamics given by $\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k))$, the following more general recursive relationship applies:

$$\left.\begin{aligned} \mathbf{x}(1) &= f\big(\mathbf{x}(0), \mathbf{u}(0)\big) \\ \mathbf{x}(2) &= f\Big(f\big(\mathbf{x}(0), \mathbf{u}(0)\big), \mathbf{u}(1)\Big) \\ \mathbf{x}(3) &= f\bigg(f\Big(f\big(\mathbf{x}(0), \mathbf{u}(0)\big), \mathbf{u}(1)\Big), \mathbf{u}(2)\bigg) \\ &\dots \end{aligned}\right\}$$

No nice matrix form for the recursion exists (as it did for the linear case), but you can still see that at each step, all instances of x, except for the initial condition, have been eliminated.

Summary

Discrete-time, finite horizon optimal control problems have been shown to be analogous to **finite-dimensional design optimization problems**…over the next few weeks, we will explore how finite-dimensional design optimization tools can be used to solve optimal control problems



…and if you can do finite-dimensional design optimization, you can do discrete-time, finite horizon optimal control

Preview of next lecture (and beyond)

Topics for lecture 5:

- Conditions for local and global extrema in ***unconstrained*** optimization problems
- Convex functions and their implications

Beyond next lecture, we will examine several different techniques for discrete time optimal control (optimizing a finite sequence of control signals):

- Convex optimization techniques (gradient approaches, Newton's method, sequential quadratic programming (SQP)) for constrained and unconstrained optimization problems
- Dynamic programming for global optimization