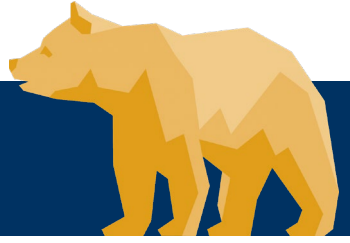
# EECS 106B/206B
# Robotic Manipulation and Interaction

**Valmik Prabhu**



# Goals of this lecture

Introduce three common nonlinear control techniques:

- Control Lyapunov Functions
- Feedback Linearization
  - Computed Torque Control
- Model Predictive Control

# INTRO TO LYAPUNOV STABILITY

Content draws from MLS 4.4 and personal notes from Koushil Sreenath's EE 222

# Nonlinear Control Systems

General Nonlinear System:

$$\dot{x} = F(x, u)$$

Control-Affine Nonlinear System:

$$\dot{x} = f(x) + g(x)u$$

Autonomous Nonlinear System:

$$\dot{x} = f(x)$$

# Lyapunov's Direct Method

We define an "energy function" V (a Lyapunov function) and consider its derivative

We define a set D around the origin (which we assume is an equilibrium point)

$$
\begin{aligned}
& V : \mathcal{D} \to \mathbb{R} \\
& V > 0, && \forall x \in \mathcal{D}, x \neq 0 \\
& \dot{V} \leq 0, && \forall x \in \mathcal{D} && \implies \text{SISL} \\
& \dot{V} < 0, && \forall x \in \mathcal{D}, x \neq 0 && \implies \text{AS} \\
& \dot{V} < -\gamma V, && \forall x \in \mathcal{D}, x \neq 0 && \implies \text{ES}
\end{aligned}
$$

# Lyapunov Stability for Driven Systems

We can define our Lyapunov function the same way, but its derivative will now depend on our input u

$$\dot{V} = L_f V + L_g V u$$

Usually we do this by defining a controller / Lyapunov function pair, and checking if they fulfill the autonomous system criteria.

$$\dot{V} = L_f V + L_g V u(x) = L_F V,$$
$$F(x) = f(x) + g(x)u(x)$$

# Linearization

The easiest way to control a nonlinear system is to linearize it about an equilibrium point and design a linear controller.

$$
\begin{aligned}
\dot{x} &= F(x,u) \\
&= F(x_{eq},u_{eq}) + \underbrace{\frac{\partial F}{\partial x}(x_{eq},u_{eq})}_{\equiv A}\cdot(x-x_{eq}) + \underbrace{\frac{\partial F}{\partial u}(x_{eq},u_{eq})}_{\equiv B}\cdot(u-u_{eq}) + h.o.t. \\
&\approx Ax + Bu,
\end{aligned}
$$

By Lyapunov's Indirect Method, a system that stabilizes the linearized system will locally stabilize the nonlinear system.

# Lyapunov Stability for Driven Systems

What if we don't want to guess and check with new controllers, but simply want to prove if one exists? If we can prove that

$$\forall x \in \mathcal{D}, \exists u \ s.t. \dot{V} \leq 0 \implies \text{SISL}$$

Then we can prove that such a controller exists.

What do we need to show to prove this?

$$L_g V = 0 \implies L_f V \leq 0$$

# Control Lyapunov Functions

If we have a Lyapunov function that fulfills the above condition, we don't actually need to define a controller. $\dot{V} = L_f V + L_g V u$ is a linear function with respect to u. Thus we can define

$$u = \text{argmin } u^T R u$$
$$s.t. \;\; L_f V(x) + L_g V(x) u \leq -\gamma V(x)$$

If we can prove that such an input exists for every x, we can solve this quadratic program at every timestep, thus producing an exponentially stable controller.

# Feedback Linearization; the Idea

We have a system

$$\dot{x} = f(x) + g(x)u$$

What if we could take this system and *invert it* to cancel out the nonlinear terms

$$\dot{x} = f(x) + g(x)u \longrightarrow Ax + Bu$$

If this were the case, we could control x as if it were a linear system.

# Input-Output Linearization of SISO Systems

In order to feedback linearize our system, we pick an output

$$y = h(x)$$

The derivative of this output is

$$\dot{y} = L_f h + L_g h u$$

If $L_g h = 0$, ie if the input doesn't affect the derivative of y, the second derivative is

$$\ddot{y} = L_f L_f h + L_g L_f h u = L_f^2 h + L_g L_f h u$$

# Relative Degree

The number of times we can take the derivative of of y before the input shows up is called the strict *relative degree*, $\gamma$, of y.

$$L_g L_f^i h(x) \equiv 0,\ \forall x \in \mathcal{D}, i = 0, \cdots, \gamma - 2$$
$$L_g L_f^{\gamma-1} h(x) \neq 0$$

If our nonlinear system has n states, the relative degree  if it is defined will be at most n. This is not obvious but needs proof (see EECS 222 textbook Sastry “Nonlinear Systems: Analysis, Stability and Control”, Springer Verlag, 1999)

# I/O Linearization of SISO Systems

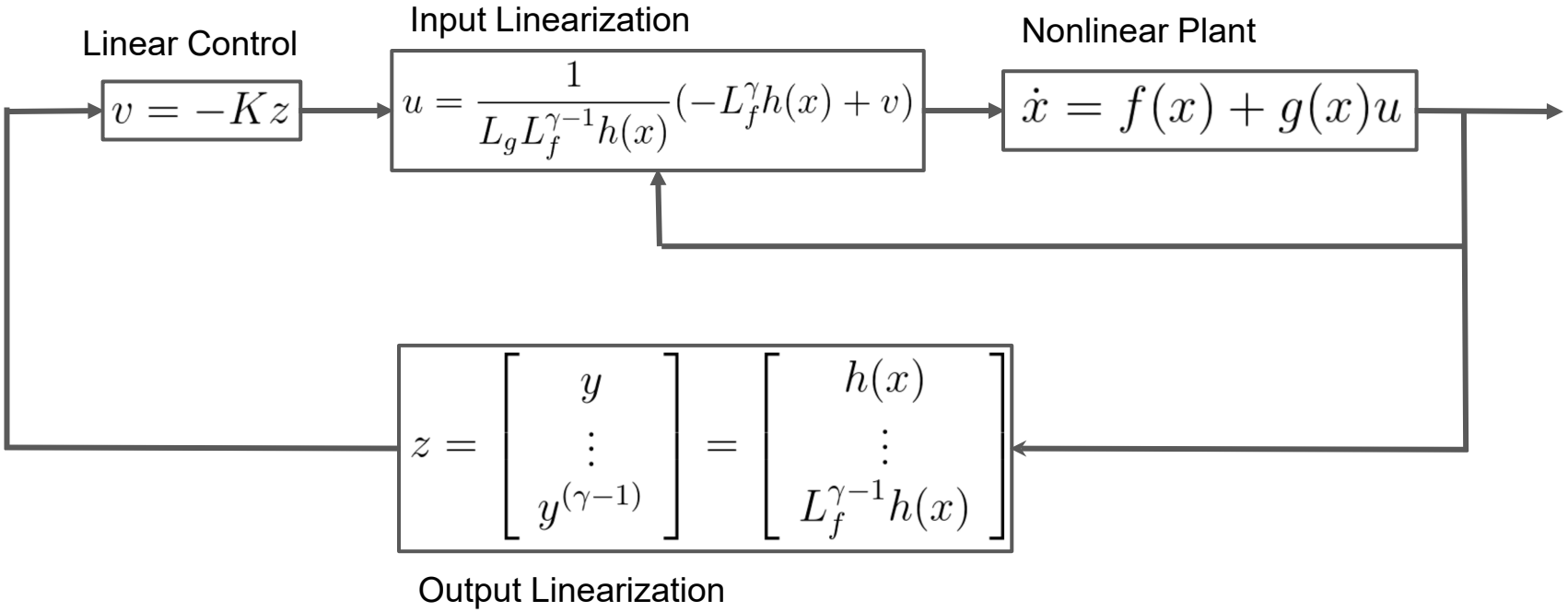
If our output y has relative degree $\gamma$, we can define a new system with $\gamma$ states

$$z = \begin{bmatrix} y \\ \vdots \\ y^{(\gamma-1)} \end{bmatrix} = \begin{bmatrix} h(x) \\ \vdots \\ L_f^{\gamma-1} h(x) \end{bmatrix} \qquad \dot{z} = \begin{bmatrix} L_f h \\ \vdots \\ L_f^{\gamma} h \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ L_g L_f^{\gamma-1} h \end{bmatrix} u$$

This resembles a linear system. If we set $u = \frac{1}{L_g L_f^{\gamma-1} h(x)} (-L_f^{\gamma} h(x) + v)$ we get

$$\dot{z} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} z + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} v$$

# IO Linearization



# Results of I/O Linearization

Using this method we can take a nonlinear system and transform it into a linear system (at least locally)

- If $\gamma$ = n, everything is good. We can fully control our nonlinear system as if it were linear
- If $\gamma$ < n, we only have a *partial* coordinate transform which cannot capture the entire dynamics of our system. This leaves what are called *zero dynamics*, which is rendered unobservable: feedback linearization makes all the nonlinearity unobservable from the I/O standpoint. If the zero dynamics are unstable, our system will destabilize even with a stable feedback linearizing controller.

# Conditions for existence of a Feedback Linearization

Until now, we've assumed that we've had a "good" output y that allows us to properly generate this feedback linearization. For such an output to even exist, our nonlinear system needs to have the following properties

- The system must be locally controllable
- The vector fields f, g  must be locally involutive

These conditions are outside of the scope of this lecture, but are intimately related to the  geometry of nonlinear systems. We will get to these in our non-holonomy lectures.

# I/O Linearization for MIMO Systems

You can do the same thing for systems with multiple inputs, but the math is much more involved . If you have  p inputs, you should also have p outputs

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_p(x) \end{bmatrix}$$

The *vector relative degree* is a vector of the relative degrees of each of the outputs. If the sum of this vector is n, there are no dynamics rendered unobservable (no zero dynamics)

# Feedback Linearization for Lagrangian Systems

Forward Dynamics:

$$\ddot{q} = M^{-1}(\tau - C\dot{q} - G)$$

Inverse Dynamics:

$$\tau = M\ddot{q} + C\dot{q} + G$$

Let’s pick our outputs

$$y_i = q_i$$

# Feedback Linearization for Lagrangian Systems

The first derivative of our outputs is

$$\dot{y}_i = \dot{q}_i$$

The second derivative of our outputs is

$$\ddot{y}_i = \ddot{q}_i = M^{-1}(\tau - C\dot{q} - G)$$

Thus the vector relative degree of our system is uniformly two. We want

$$M^{-1}(\tau - C\dot{q} - G) = v$$

Inverting this yields

$$\tau = Mv + C\dot{q} + G$$

# Feedback Linearization for Lagrangian Systems

Here we've shown that the inverse dynamics are a feedback linearizing controller for *all fully actuated Lagrangian systems*, including many of the robots and multifingered hands we will work with in this course. Underactuated systems are much more challenging to work with, because finding good controllers is much more challenging

# Computed Torque Control

This is the easiest controller to use on an open-chain manipulator

$$\tau = M(\ddot{q}_d - K_p q - K_d \dot{q}) + C\dot{q} + G$$

The dynamics become

$$\ddot{q} = \ddot{q}_d - K_p q - K_d \dot{q}$$

# Problems with Feedback Linearization

Feedback linearization **assumes** that we know the exact dynamics of the system. If we don’t have it exactly right, our controller will not cancel out the nonlinear dynamics, and our system will potentially become unstable

# Model Predictive Control

The overall goal of model predictive control is to turn our control problem into an optimization problem.

At the first timestep, we plan N steps in the future (from t = 0 to N), then execute the first step in the plan

At the next timestep, we again plan N steps into the future (from t = 1 to N+1), then again execute the first step in the plan.

This is also called *receding horizon control*, because your time horizon moves forward at every timestep.

# Discrete Time Nonlinear systems

Most model predictive control requires that we *discretize* our system.

$$\dot{x} = f(x) + g(x)u$$
$$x_{k+1} = x_k + f(x)\delta t + g(x)\delta t u$$

This is called *Euler discretization*, and works if delta t is very small.

Discrete time systems are easier to implement on computers than nonlinear systems and have a couple computational advantages, but continuous systems are easier to deal with in many cases.

# Model Predictive Control

Consider the nonlinear time-invariant system

$$x(t+1) = g(x(t), u(t)),$$

subject to the constraints

$$h(x(t), u(t)) \leq 0, \ \forall t \geq 0$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ the state and input vectors. ***Assume that*** $g(0,0) = 0$, $h(0,0) \leq 0$.

# Model Predictive Control

Consider the following *objective* or *cost* function

$$J_{0\to N}(x_0, U_{0\to N}) := p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$$

where

- $N$ is the time *horizon*,
- $x_{k+1} = g(x_k, u_k),\ k = 1, \ldots, N-1$ and $x_0 = x(0)$,
- $U_{0\to N} := [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^s,\ s := mN$,
- $q(x_k, u_k)$ and $p(x_N)$ are the *stage cost* and *terminal cost*, respectively.

# Model Predictive Control

Consider the **C**onstrained **F**inite **T**ime **O**ptimal **C**ontrol (CFTOC) problem.

$$
\begin{aligned}
J^*_{0\to N}(x_0) = \quad & \min_{U_{0\to N}} \quad && J_{0\to N}(x_0, U_{0\to N}) \\
& \text{subj. to} && x_{k+1} = g(x_k, u_k),\ k = 0, \dots, N-1 \\
& && h(x_k, u_k) \leq 0,\ k = 0, \dots, N-1 \\
& && x_N \in \mathcal{X}_f \\
& && x_0 = x(0)
\end{aligned}
$$

- $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a *terminal region*,
- $\mathcal{X}_{0\to N} \subseteq \mathbb{R}^n$ to is the set of feasible initial conditions $x(0)$
- the optimal cost $J^*_{0\to N}(x_0)$ is called *value function*,
- assume that there exists a minimum
- denote by $U^*_{0\to N}$ one of the minima

# Sources

MLS Chapter 4.5

Sastry Nonlinear Systems: Analysis, Stability, and Control, Springer Verlag, 1999 Chapter 9.