A

EECS C106B / 206B Robotic Manipulation and Interaction **Spring 2022**

Lecture 2A: Introduction to Control - Stability, Lyapunov Methods

Scribes: Maxime Defauw, Amit Palekar, Enya Xing

2.1 Control

The goal of control of a system is that even with small deviations, they will be dampened.

2.1.1 Control of a First Order System

Problem
We start with state x and input u, where u is a velocity. We construct the kinematic model $\dot{x} = u$ Our goal here is to follow the trajectory $x^{des}(t)$

General approach
We define error $e(t) = x^{des}(t) - x(t)$. Our goal is to have our error converge exponentially to 0.

Strategy
Our goal here is to find u such that $\dot{e} + K_p e = 0$. If we have a $K_p > 0$ then $e(t) = exp(-K_p(t - t_0)e(t_0)$. Here, K_p helps determine how fast the error converges to 0 as proportional feedback. We define $u(t) = \dot{x}^{des}(t) + K_p e(t)$, where $K_p e(t)$ is the feedback term.

2.1.2 Control of a Second Order System

Problem
We start with state x and input u. We construct the kinematic model $\ddot{u} = u$, where u is an acceleration. We want to follow the trajectory $x^{des}(t)$.

General approach
We define our error term $e(t) = x^{des}(t) - x(t)$. Again, we want our error to converge exponentially to 0.

Strategy
As we only have control over the 2nd derivative, we want to find a u such that $u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e = 0$. $\ddot{x}^{des}(t)$ is the feed-forward term, $K_d\dot{e}(t)$ is the derivative term, and $K_p e$ is the proportional term. $K_d, K_p > 0$

2.1.3 Control for Trajectory Tracking

PD Control
$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e = 0$. $\ddot{x}^{des}(t)$. The K_p, or proportional term, has a capacitance response and K_d, or derivative term, has a resistance response.

PID Control
$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e = 0 + K_I \int_0^t e(\tau)d\tau$. Here, the integral term K_I helps atone for any model imperfections to make the steady state error to go to 0.

2.1.4 Control Gains

Stereotyped 2nd Order Response

- $\ddot{e} + K_d\dot{e} + K_p e = 0$
- $\ddot{e} + 2\zeta w_n \dot{e} + w_n^2 e = 0$
- $\lambda = -w_n(\zeta \pm i\sqrt{1-\zeta^2})$

Here ζ is the dampening ratio. There are 2 closed loop eigenvalues. If $\zeta > 1$, both λ are real. If $\zeta < 1$, the response is more lively.

A high gain K_p value results in a more bouncy and lively response. A low K_p value results in a more staged and steady response. Through manual tuning of the K_p, K_d, K_I terms, it is possible to ijmpact rise time, overshot, settling time, and steady-state error, as general guidelines. An example of PID gain tuning is the *Ziegler-Nichols Method* which relies on increasing K_p until the ultimate gain K_u is as big as possible.

Model-Based Control

The model-based control law is written as $u(t) = m(\ddot{x}(t) + K_d\dot{e}(t) + K_p e(t)) + b\dot{x}(t) + kx(t)$. The servo-based component, or $(\ddot{x}(t) + K_d\dot{e}(t) + K_p e(t))$ is used to drive the error term to zero. The model-based component, m and $+b\dot{x}(t) + kx(t)$ is model-specific.

2.2 Written Example

2.2.1 Quadrotor example

The equilibrium, in the hover configuration is

$$\mathbf{q_e} = \begin{bmatrix} y_0 \\ z_0 \\ 0 \end{bmatrix}, \mathbf{x_e} = \begin{bmatrix} q_e \\ 0 \end{bmatrix} \text{ with } y_0, z_0\phi_0 = 0 u_{1,0} = mg, u_{2,0} = 0$$

To make it look like a control system — with a desired y and z — you need to build a second order controller for y and z, while considering ϕ an input. Then construct another controller to get the desired ϕ. Each of these controllers will make for a first order system.

The inner loop in this system should be fast. This is needed for example in aircraft control systems. The first thing test engineers make sure is to have the ϕ control in place before moving to control other variables.

In the quadrotor case, the equations are as follows

Lateral dynamics
$\ddot{y} = -g\phi$
$\ddot{\phi} = \frac{u_2}{I_{xx}}$

Vertical dynamics
$\ddot{z} = \frac{u_1}{m}$

Desired attitude
$\phi_{des} = -\frac{\ddot{y}_c}{g}$
$\dot{\phi}_{des} = 0$
$\ddot{\phi}_{des} = 0$

Z-position controller
$u_1 = m(\ddot{z}_c)$

Attitude controller
$u_2 = I_{xx}\ddot{\phi}_c$

Looking at the vertical dynamics first. Since $\ddot{z}$ is a controlled variable, you figure out what $m\ddot{z}_c$ is. Next, $u_2 = I_{xx}\ddot{\phi}_c$, which is a function of some controlled variable $\ddot{\phi}_c$. To get ϕ_c, you take y_{des} and pretend that phi is like an input.

Putting all this together we get the following equations.

Control equations
$u_1 = m(\ddot{z}_c + k_{d,z}(\dot{z}_{des} - \dot{z}) + k_{p,z}(z_{des} - z))$
$u_2 = I_{xx}(\ddot{\phi}_{des} + k_{d,\phi}(\dot{\phi}_{des} - \dot{phi}) + k_{p,\phi}(\phi_{des} - \phi))$
$\phi_{des} = -\frac{1}{g}(\ddot{z}_{des} + k_{d,y}(\dot{y}_{des} - \dot{y}) + k_{p,y}(y_{des} - y))$

We first start with ϕ_{des} and do a PD control in the y direction. Tweak the $k_{d,y}$ and k_{py}. u_2 has to be a fast variable since it's the inner attitude loop. Therefore, a high $k_{p,y}$ is needed so that the desired value of ϕ follows from treating it as an input variable. The you tweak the $k_{d,z}$ and $k_{p,z}$ for u_1. In total, you tweak three sets of PD gains.

This corresponds to the mantra of test flight engineers: *aviate, navigate, communicate.* Aviate means staying alive — controlling thrust and roll. Navigate means you go where you want to go — controlling the position. And lastly, communicate means telling other aircrafts where you are.

2.2.2 A quick introduction to non-linear control

In non-linear control, you're interested in controlling non-linear affine systems (like the quadrotor).

In the SISO (single input, single output case: input u and output y with state x) these are the default parameters:

$\dot{x} = f(x) + g(x)u, x \in R$
$y = h(x), y \in R$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, f(x) = \begin{bmatrix} f_1(x)) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix}, g(x) = \begin{bmatrix} g_1(x)) \\ g_2(x) \\ \vdots \\ g_n(x) \end{bmatrix}$$

A lot of systems are non-linear, which it does not need to be. If for example, we want to control y (let if follow a $y_{des}(t)$ and we have sensors measuring all the x's and an input u. The approach is to differentiate y w.r.t. t. $y(x)$ is a function of time because $x(t)$. By convention, a derivative is always a row vector as shown.

$y(t) = h(x(t))$
$\frac{\mathrm{d}}{\mathrm{d}t}y(t) = \frac{\mathrm{d}}{\mathrm{d}t}h(x(t)) = \frac{\mathrm{d}h(x)}{\mathrm{d}x}\dot{x} = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} & \cdots & \frac{\partial h}{\partial x_n} \end{bmatrix} \dot{x}$

Replacing x_{dot} with its definition gives us the end expression for the derivative of the output y along trajectories of the control system.

$\frac{\mathrm{d}}{\mathrm{d}t}y(t) = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} & \cdots & \frac{\partial h}{\partial x_n} \end{bmatrix} \dot{x} = \begin{bmatrix} \frac{\mathrm{d}h(x)}{\mathrm{d}x} \end{bmatrix} \begin{bmatrix} f(x) + g(x)u \end{bmatrix}$
$\dot{y} = \frac{\mathrm{d}h(x)}{\mathrm{d}x}f(x) + \frac{\mathrm{d}h(x)}{\mathrm{d}x}g(x)u$

This terms in y_{dot} have names. The first term is called the *Lie derivative* of h w.r.t. f or how much h

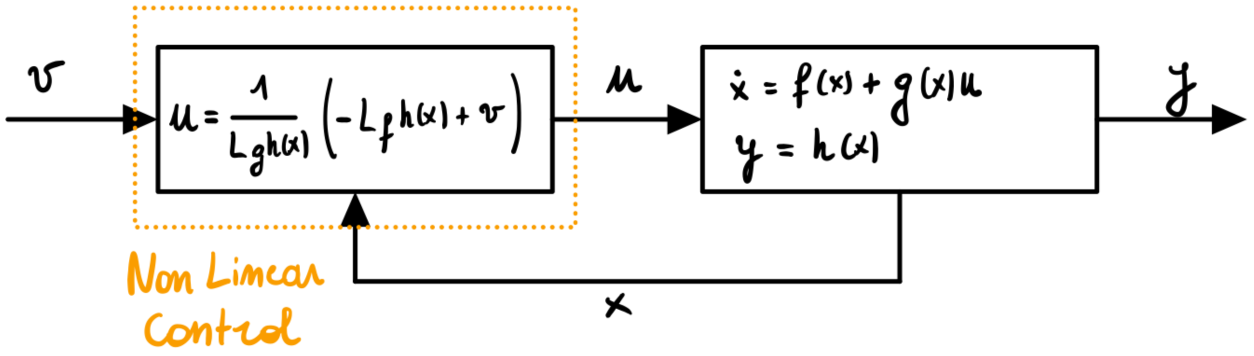

Figure 2.1: The model with non-linear control

changes in the f direction. The second term is the *Lie derivative* of h w.r.t. g or how much h changes in the g direction. A shorthand notation is $L_f h(x)$ and $L_g h(x)$

If in this equation $L_g h(x) \neq 0$, then one should choose a control law u:

$u = \frac{1}{L_g h(x)}(-L_f h(x) + v)$

Putting this into the expression for $\dot{y}$, (non-linear) terms cancel out:

$y = L_f h(x) + L_g h(x) \frac{1}{L_g h(x)}(-L_f h(x) + v)$

With v being a new auxiliary input.

This gives the following expression with every Lie derivative cancelling out until $y_{dot} = v$.

In summary, what you have done here can be shown the following diagram. u is the input, y is the output. You can measure all x'es, which is fed into a computed non-linear control law from above. This control law asks for a new auxiliary input v. The result is fed into the plant as u.

This is a non-linear control system with a new input v. There is now a transfer function from v to y. This is therefore called an input/output linearization of this first order system.

To tack $y_{des}(t)$, we set v to be:

$v = \dot{y}_{des}(t) + k_1(y_{des} - y)$

This guarantees that the error goes to 0 exponentially.

$e = y_{des} - y$
$\dot{e} + k_1 e = 0, e \to 0$ exponentially

To do this, we set u in the composite controller to the following:

$u = \frac{1}{L_g h(x))}\left[-L_f h(x) + \dot{y}_{des}(t) + k_1(y_{des} - y)\right]$

2.2.3 Non-linear control (continued)

What if $L_g h(x) = 0$?

$\dot{y} = L_f h(x) + L_g h(x) u = L_f h(x)$
$\ddot{y} = \frac{d}{dt}[\dot{y}] = \frac{d}{dt}[L_f h(x)]$

$\ddot{y} = \frac{d}{dx}\left[L_f h(x)\right] \cdot \left[f(x) + g(x)u\right]$
$\ddot{y} = \frac{d}{dx}\left[(L_f h(x))f(x)\right] + \frac{d}{dx}\left[(L_f h(x))g(x)u\right]$
$\ddot{y} = L_f(L_f(h(x)) + L_g(L_f h(x))u$
$\ddot{y} = L_f^2 h(x) + L_g(L_f h(x))u$

If $L_g L_f h(x)! = 0$,
$u = \frac{1}{L_g L_f h(x)}[-L_f^2 h(x) + v]$
$\ddot{y} = v$

Here we managed turn our nonlinear control into a 2nd order linear system.

If we want y $\rightarrow y_{des}$:

$v = \ddot{y_{des}} + k_1(\dot{y_{des}} - \dot{y}) + k_2(y_{des} - y)$
$= \ddot{y_{des}} + k_1(\dot{y_{des}} - L_f h(x)) + k_2(y_{des} - h(x))$

Now we would set u:

$u = \frac{1}{L_g L_f h(x)}[-L_f^2 h(x) + \ddot{y_{des}}(t) + k_1[$

Terminology: r is the relative degree which is the number of times you need to differentiate y to get the input u to show up.