Lecture # 11

Last time: Feedback linearization

$$x^{(n)} = f(\underbrace{x, \dot{x}, \dots, x^{(n-1)}}, t) + b(x, \dot{x}, \dots, x^{(n-1)}, t)\, u .$$

reformulate system to have state $\underline{X} = \begin{pmatrix} x \\ \dot{x} \\ \vdots \\ x^{(n-1)} \end{pmatrix}$

If you know f, b then set $f + bu = v$ and solve for u.

This is similar to pole placement in linear systems. P.1.1 (A)
- take a dynamics & replace it w/ something else using feedback (change coeffts)

Suppose you have a magical controller that gives you $x(t) \equiv x_d(t)$ exactly. Then $u(t)$ is the unique solution to

$$\underline{x}_d^{(n)} = f(\underline{x}_d, t) + b(\underline{x}_d, t)\, u$$

Note: If the solution, it is difficult to find it - You need algorithms.

Thus, all the tools we have used before (adaptive control, robust/sliding control) are trying to achieve this ideal control input. But it is difficult because we don't have all the information.

p1.1 (B) Then we said that if we only care about one variable, we use that variable to keep differentiating that variable until 'u' pops out.

Sometimes this leads to left over dynamics if 'u' pops out earlier than the full order of the system.

The whole line of this research is to extend linear systems research.

(A) But of course we donot know δ, b accurately and so we did robust control. (p1.1)

So in the linear case, if you had slightly wrong params, you would be slightly off. But if you very wrong params, you would be way off (poles in RHP).

But in nonlinear case, bad things can happen even for small errors. Thus, we did robust control.

(B) So the whole idea behind feedback linearization is to see how general this approach (change of variables) is.

There are two points of view here:

(1) Retain the sys but find state transformation so that the last eq'n is $x^{(n)} = v$ and remaining eq'ns are $\left(\begin{array}{l} x_1 = x \\ x_2 = \dot{x} \\ \vdots \end{array} \right)$ or we converted the eq'n into a linear sys.

(2) The other way is to start w/ $x(t)$ and differentiate it until you get 'u' in the equation.

And if you have to differentiate exactly 'n' times (to find u), then you would have precisely transformed the system.

If it happens before, then you have leftover dynamics, which you hope will not blow up.

For example,

System I

$\dot{x}_1 = x_2 + u$

$\nearrow$ differentiate once and input pops out.

(If we care only about controlling x_1) $\dot{x}_2 = u$

System II

$\dot{x}_1 = x_2 + u$

$\dot{x}_2 = -u$

Setting $u = -x_2 - x_1$, worked for sys. I

But not for system II

Transfer function

$$\frac{x_1}{u} = \frac{s+1}{s^2}$$

zero in LHP

Inverse sys. is stable.
[If x_1 behaves, u behaves]
[This is called a minimum-phase system.
$\Rightarrow$ An LTI sys. and its inverse is causal and stable]

$$\frac{x_1}{u} = \frac{s-1}{s^2}$$

zero in RHP

Inverse sys. is unstable
[If x_1 behaves, u does not behave]
[Non-minimum phase]

So, can we show this result more generally for linear systems & then look @ non-linear systems?

– take system, differentiate until you get control, and only control that $\Rightarrow$ will this work generally?

(section 6.1.3)

- Given linear system

scalar → $\dfrac{z}{u} = C(sI - A)^{-1}B\,u = \dfrac{b_0 + b_1 s + \dots + b_m s^m}{a_0 + a_1 s + \dots + a_{n-1}s^{n-1} + s^n}\,u$

$C(sI-A)^{-1}B$: transfer function in Laplace form from state-space repr.

n-poles, n = dimension of state
m zeros.
n−m = relative degree/order of sys.

$$= (b_0 + b_1 s + \dots + b_m s^m)\left(\frac{1}{a_0 + a_1 s + \dots + a_{n-1}s^{n-1} + s^n}\right)u$$

$x_3 = \ddot{x}_1$, $x_n = x_1^{(n-1)}$

$\dot{x}_1 = x_2$, $\dot{x}_2 = x_3$

$\dot{x}_{n-1} = x_n$

$\dot{x}_n = -a_0 x_1 - a_1 x_2 - \dots - a_{n-1} x_n + u$

$= -a_0 - a_1\dot{x}_1 - \dots - a_{n-1} x_1^{(n-1)} + u$

$s^n x_1 = -(a_0 + a_1 s - \dots - a_{n-1}s^{n-1})x_1 + u$

$x_1(a_0 + a_1 + \dots + a_{n-1} + s^n) = u$

So How does this look in companion form?

$$\frac{d}{dt}\underline{x} = \begin{bmatrix} \begin{matrix}0\\ \vdots \\ 0\end{matrix} & \begin{bmatrix} I_{n-1} \end{bmatrix} \\ -a_0 \;\; -a_1 & \cdots \;\; -a_{n-1} \end{bmatrix}\underline{x} + \begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix}u$$

$$z = [\,b_0 \;\; b_1 \; \dots \; b_m \;\; \underbrace{0 \;—\; 0}_{n-(m+1)\text{ zeros}}\,]\,\underline{x} = b_0 x_1 + b_1 x_2 + \dots + b_m x_{m+1}$$

$$= b_0 x_1 + b_1 \dot{x}_1 + \dots + b_m x_1^{(m)}$$

$[\,z(s) = b_0 + b_1 s + b_m s^m\,]\,x_1$

z = output of interest.

let's differentiate z until u shows up.

$$\dot{z} = [\,0 \;\; b_0 \;\; b_1 \; \dots \; b_m \; \underbrace{0 \;—\; 0}_{\text{one less zero}}\,]\,\underline{x}$$

shift by one step.

n−m zeros

⋮

$$z^{(n-m-1)} = [\,0 \;——\; 0 \;\; b_0 \;\; b_1 \; \dots \; b_m\,]\,\underline{x}$$

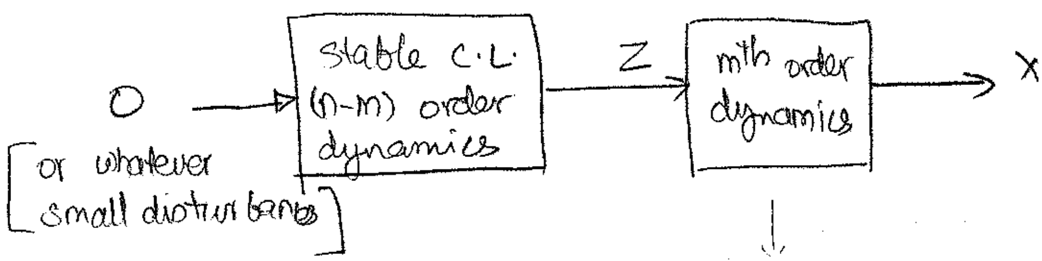
Note that u has not yet shown up.

Differentiate once more and then only 'u' appears.
(from $\dot{x}_n$)

$$z^{(n-m)} = \underbrace{u + \cdots}_{\text{call this } v}$$

$\Rightarrow$ Differentiate $(n-m)$ times for 'u' to appear = relative degree of the transfer function $\frac{z}{u}$

So we will create the following



$$b_0 x_1 + b_1 \dot{x}_1 + b_2 \ddot{x}_1 + \cdots + b_m x_1^{(m)} = z$$

$\Rightarrow$ This transfer function is

$$\frac{x_1}{z} = \frac{1}{b_0 + b_1 s + \cdots + b_m s^m}$$

left-over dynamics

Thus we are seeing that the original system's zeros show up in the left-over dynamics as poles & that is why we were stressing minimum phase as a criteria. and sys. does not blow up.

Note that this is similar to the robust control idea where define a sliding variable s w/ relative degree of 1. and the left-over dynamics has the remaining relative degree ($\dot{s}$ contains u) (definition of s)

So if you know a sys. is minimum phase, then you can ignore left-over dynamics

Can we do this for non-linear systems? (p5)

- The left-over dynamics is more appropriately called zero dynamics

when $b_0 x_1 + \cdots + b_m x^{(m)} = 0$

This corresponds to ~~the~~ setting $z = 0$

In linear case, stable zero dynamics $\Rightarrow$ "stable" $\underbrace{\text{left-over}}_{\text{internal}}$ dynamics

(As $z \to 0$, $x_1 \to 0$)

Linear sys: ✓ we know how to do it.

Nonlinear sys: $\begin{cases} \text{Left-over dynamics corresponding to the} \\ \text{Unique } u(t) \text{ that makes } z \equiv 0 \\ \qquad \text{(or by extension } x = 0) \end{cases}$

- Since $z^{(n-m)} = v$, there is a unique choice of v, and thus u, that makes z stay @ zero.

But in general for nonlinear systems,

"stable" zero dynamics $\not\Rightarrow$ "stable" internal dynamics

$\Leftarrow$ Inverse is true.

(So it is not straightforward to extend the notion of minimum phase to nonlinear sys.)

But if you have nonlinear zero dynamics + then you linearize it, and then you have unstable linearized zero dynamics, then of course the zero dynamics is unstable as well.

This is one use of this method. [show that sys. is unstable]

The other use of this method is to have a simpler left-over - dynamics, for which you can create a Lyapunov controller.

[we may see a technique called Backstepping for this]

section 6.2

Now some more math to work this out

Can we find a $\boxed{\text{smooth invertible}}$ stable state transformation (diffeomorphism)

for a sys. $\dot{x} = f(x) + b(x)u$

$\underline{z} = \underline{z}(\underline{x})$ such that

$$\underline{z} = \begin{bmatrix} z_1 \\ \dot{z}_1 \\ \vdots \\ z_1^{(n-1)} \end{bmatrix} \quad \text{and} \quad \underbrace{z_1^{(n)}}_{\substack{\text{equivalent to saying find variable } 'z_1' \\ \text{relative degree } 'n'}} = v \quad \text{where}$$

$$u = \alpha(\underline{x}) + \beta(\underline{x})v$$

For this, we need a number of tools to do this.

But first let's define a few things.

Suppose we have a scalar function $h(\underline{x})$

$$\nabla h = \text{gradient as a row vector} = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \cdots & \frac{\partial h}{\partial x_n} \end{bmatrix}$$

If we have a vector field. $\underline{f}(x)$,

$$\underbrace{\nabla \underline{f}}_{\text{Jacobian}} = \left(\frac{\partial f_i}{\partial x_j}\right)_{ij} = \begin{bmatrix} \nabla f_1 \\ \nabla f_2 \\ \vdots \\ \nabla f_n \end{bmatrix}$$

Now, Lie derivative of h wrt $\underline{f}$ is the scalar function $L_{\underline{f}} h$

$$L_{\underline{f}} h = \nabla h \cdot \underline{f}$$

= directional derivative of h along f.

recursive definition of Lie derivatives

$$L_{\underline{f}}^0 h = h, \quad L_{\underline{f}}^1 h = L_{\underline{f}} h = \nabla h \cdot \underline{f}$$

$$L_{\underline{f}}^2 h = L_{\underline{f}}(L_{\underline{f}} h) \qquad L_{\underline{f}}^n h = L_{\underline{f}}(L_{\underline{f}}^{n-1} h) = \nabla(L_{\underline{f}}^{n-1} h) \cdot \underline{f}$$

Why is this useful?

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

$y = h(x)$ → you want to differentiate y multiple times to get u.

$$\dot{y} = \frac{\partial h}{\partial x}\dot{x} = \frac{\partial h}{\partial x}f = L_f h$$

$$\ddot{y} = L_f^2 h$$ Tells us what the derivatives look like in terms of f and h.

So if you have Lyapunov function $V(x)$,

$$\dot{V}(x) = L_f V$$

Another operator — Lie bracket on two vectors f and g

$$[f, g] = \text{vector field}$$

$$= \nabla g\, f - \nabla f\, g$$ [Note: sometimes the definition is the other way around]

Suppose we have a set of $\underbrace{\text{linearly independent}}_{\text{no linear combination}}$ vector fields $f_1, \ldots, f_n$.

This set is involutive if $\underbrace{[f_i, f_j]}_{} = \sum_k \alpha_{ijk}(x) f_k(x)$

Lie bracket is still an element of the family space

No new space is explored

Won't take you to new spaces.

Frobenius theorem: relating involutivity + integrability

- will tell us if we can find the transformation talked about on P6 .