16B Prof. Anant Sahai

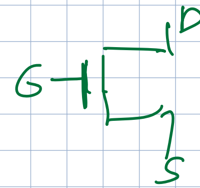
Today: Linearization in General
Linearizing for control systems
Solving nonlinear equations

Announcement: $\boxed{\text{Final Fri Dec 17 8AM}}$
We're slowing down and triaging course scope: DFT material will be dropped this term to permit less HW problems per assignment & more review at the end of the term.

Step Way Back: All of our 16AB models have been linear

except switches / comparators ← nonlinear.

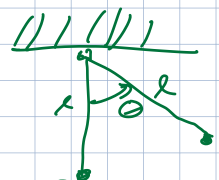
Real World is largely nonlinear.



← Actually smoothly nonlinear.

Almost all mechanical systems are nonlinear.

Consider a pendulum:



e.g. Get sines & cosines in equations governing motion

Need a tool & Philosophy for attacking nonlinear problems.

Tool: Local Approximation: Just Pretend things are linear.

Taylor Series:
$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2} + \dots$$

When x is close to x_0, the higher-order terms contribute very little.

$\Longrightarrow$ lump all dropped terms into the disturbance

Consider: $f(x, u) = 32 + x^3 u^2$

expand around (x_0, u_0)

$$f(x_0 + \delta x, u_0 + \delta u) = 32 + (x_0 + \delta x)^3 (u_0 + \delta u)^2$$

$$= 32 + \left(x_0^3 + 3x_0^2\,\delta x + 3x_0(\delta x)^2 + \delta x^3\right) \cdot \left(u_0^2 + 2u_0\,\delta u + (\delta u)^2\right)$$

If δx and δu are both tiny then $(\delta x)(\delta u)$, $(\delta x)^2$, etc.

All higher powers are extremely small.

$$f(x_0+\delta x, u_0+\delta u) \approx \underbrace{32 + x_0^3 u_0^2}_{f(x_0,u_0)} + \boxed{3x_0^2 u_0^2}\,\delta x + \boxed{2x_0^3 u_0}\,\delta u + \text{super tiny}$$

← lump this into disturbance

This gives an approximate "linear" model in the neighborhood of (x_0, u_0).

↖ Actually "affine" since there is the constant term in front

What is $3x_0^2 u_0^2$? ← Observe: This is the derivative of $f(x,u)$ with respect to x treating u as a constant, evaluated at (x_0, u_0)

This is called the partial derivative $\frac{\partial f}{\partial x}$.

↑ Because you treat other symbols in f as constants.

Similarly $\frac{\partial f}{\partial u} = 2x^3 u$ evaluated at (x_0, u_0) to get $2x_0^3 u_0$

Apply idea to control problems. Nonlinear control system example.

Suppose $\frac{d}{dt}x(t) = f(x,u) = 32 + x^3(t)\,u^2(t)$

Suppose the goal was to keep $x(t) = -2$.

Can we get $x_0 = -2$ to be a "steady state solution"? i.e. Can it stay there?

First Necessary Condition: Does there exist an input u_0 so that $f(-2, u_0) = 0$.

$$32 + (-2)^3 u^2 = 0$$

$$32 - 8u^2 = 0$$

$$8u^2 = 32 \implies u_0 \text{ could be } -2 \text{ or } +2$$

Pick $u_0 = +2$. ∴ Our equilibrium point / operating point is $(x_0, u_0) = (-2, +2)$

$$\frac{d}{dt}x(t) \approx \underset{\substack{\uparrow \\ f(x_0, u_0)}}{0} + 48\,\delta x(t) - 32\,\delta u(t)$$

$$\left.\frac{\partial f}{\partial x}\right|_{x_0, u_0} = 48 \qquad \left.\frac{\partial f}{\partial u}\right|_{x_0, u_0} = -32$$

What are $\delta x(t)$ & $\delta u(t)$ in terms of $x(t)$, $u(t)$?

$$x_0 + \delta x(t) = x(t) \implies \delta x(t) = x(t) - x_0 = x(t) + 2$$

similarly $\delta u(t) = u(t) - u_0 = u(t) - 2$

$$\frac{d}{dt}x(t) \approx 0 + 48(x(t) + 2) - 32(u(t) - 2)$$

Let

$$\tilde{x}(t) = x(t) + 2$$

$$\tilde{u}(t) = u(t) - 2$$

$$\frac{d}{dt}\tilde{x}(t) = 48\,\tilde{x}(t) - 32\,\tilde{u}(t) + w(t)$$

disturbance includes the approximation error

Is this stable?? No! Because $48 > 0$.

Apply continuous-time feedback:

e.g. $\tilde{u}(t) = 2 \cdot \tilde{x}(t) \implies$ So $u(t) = 2 + 2(x(t) + 2)$

$$\implies \frac{d}{dt}\tilde{x}(t) = (48 - 64)\,\tilde{x}(t) + w(t)$$

$$= \underbrace{-16}_{\text{stable since } -16 < 0.}\,\tilde{x}(t) + w(t)$$

Yes! We can practically hold the system state x near -2.

Recipe: Have a goal: a state x_0 we want to hold at.

We are "reducing" to the linear case we know how to solve!

1) Is this even possible?
   Find a u_0 s.t. $f(x_0, u_0) = 0$

2) Linearize the model around (x_0, u_0) treating all approximation error as disturbance.

3) Do control as though the linear model from previous step were true.
   And it is true since we have the disturbance term!

Power & Generality of this recipe makes us want to do this for vector $\vec{x}(t)$ & vector $\vec{u}(t)$.

Consider: $\frac{d}{dt}\vec{x}(t) = \vec{f}(\vec{x}(t), \vec{u}(t))$ ← Nominal Nonlinear Model.

Where $\vec{f}(\vec{x},\vec{u}) = n\left\{\begin{bmatrix} f_1(\vec{x},\vec{u}) \\ f_2(\vec{x},\vec{u}) \\ \vdots \\ f_n(\vec{x},\vec{u}) \end{bmatrix}\right.$ where $\vec{x} \in \mathbb{R}^n$ ← n-dim, $\vec{u} \in \mathbb{R}^m$ ← m-dim.

Recall System-ID thinking: Do things row by row.

Linearize row by row.

$$\underbrace{f_i(\vec{x},\vec{u})}_{\text{Just } n+m \text{ scalar variables.}} = f_i(\vec{x}_0,\vec{u}_0) + \left.\frac{\partial f_i}{\partial \vec{x}}\right|_{\vec{x}_0,\vec{u}_0} \cdot (\vec{x}-\vec{x}_0) + \left.\frac{\partial f_i}{\partial \vec{u}}\right|_{\vec{x}_0,\vec{u}_0}(\vec{u}-\vec{u}_0) + w_i$$

where $\frac{\partial f_i}{\partial \vec{x}} = \underbrace{\begin{bmatrix} \frac{\partial f_i}{\partial x_1} & \frac{\partial f_i}{\partial x_2} & \cdots & \frac{\partial f_i}{\partial x_n} \end{bmatrix}}_{\text{Length } n}$ ← row.

Why? Because $\partial f_i/\partial \vec{x} \cdot (\vec{x} - x_0)$ is scalar and $\dfrac{\partial f_i}{\partial \vec{u}} = \underbrace{\begin{bmatrix} \dfrac{\partial f_i}{\partial u_1} & \dfrac{\partial f_i}{\partial u_2} & \cdots & \dfrac{\partial f_i}{\partial u_m} \end{bmatrix}}_{\text{length } m.}$

Put rows together: $\vec{f}(\vec{x},\vec{u}) \approx \vec{f}(\vec{x}_0,\vec{u}_0)$

$$+ \left.\frac{\partial \vec{f}}{\partial \vec{x}}\right]_{\vec{x}_0,\vec{u}_0} (\vec{x}-\vec{x}_0)$$

$$+ \left.\frac{\partial \vec{f}}{\partial \vec{u}}\right]_{\vec{x}_0,\vec{u}_0} (\vec{u}-\vec{u}_0)$$

Where $\dfrac{\partial \vec{f}}{\partial \vec{x}} = \left.\underbrace{\begin{bmatrix} \dfrac{\partial f_1}{\partial \vec{x}} \\ \dfrac{\partial f_2}{\partial \vec{x}} \\ \vdots \\ \dfrac{\partial f_n}{\partial \vec{x}} \end{bmatrix}}_{n} \right\} n$ ← $n \times n$ square matrix

So $\left.\dfrac{\partial \vec{f}}{\partial \vec{x}}\right)_{\vec{x}_0,\vec{u}_0}$ plays the role of the "A" matrix

$\dfrac{\partial \vec{f}}{\partial \vec{u}} = \left.\underbrace{\begin{bmatrix} \dfrac{\partial f_1}{\partial \vec{u}} \\ \vdots \\ \dfrac{\partial f_n}{\partial \vec{u}} \end{bmatrix}}_{m} \right\} n$ ← Possibly rectangular matrix since n & m could be different.

& $\left.\dfrac{\partial \vec{f}}{\partial \vec{u}}\right]_{\vec{x}_0,\vec{u}_0}$ plays the role of the "B" matrix

$$\frac{d}{dt}\vec{x}(t) = \underbrace{\boxed{\vec{f}(\vec{x}_0,\vec{u}_0)}}_{\text{Annoying term.}} + A(\vec{x}(t)-x_0) + B(\vec{u}(t)-\vec{u}_0) + \vec{w}(t)$$

← disturbance + approximation error.

Recall that an equilibrium point has $\vec{f}(\vec{x}_0, \vec{u}_0) = \vec{0}$.

So if you want to hold the system at $\vec{x}_0$, need to find $\vec{u}_0$ s.t. $\vec{f}(\vec{x}_0, \vec{u}_0) = \vec{0}$.

Aside: Can we always find such a $\vec{u}_0$?

Sadly No.

Consider $f(x,u) = 32 + x^3 u^2$

If $x_0 > 0$ then $0 = 32 + (x_0)^3 u^2$ has no real solutions

---

How can we find a $\vec{u}_0$ s.t. $\vec{f}(\vec{x}_0, \vec{u}_0) = \vec{0}$?

i.e. How can I solve a system of nonlinear equations?

Key Idea: Iteration: Newton's Method.

To simplify notation: Let $\vec{g}(\vec{u}) = \vec{f}(\underset{\substack{\uparrow \\ \text{fixed}}}{\vec{x}_0}, \vec{u})$ ← what I want to find

Want to solve $\vec{g}(\vec{u}) = \vec{0}$

Use linearization. $\vec{g}(\vec{u}) \approx \vec{g}(\vec{u}_0) + \left.\frac{\partial \vec{g}}{\partial \vec{u}}\right|_{\vec{u}_0} (\vec{u} - \vec{u}_0)$

Can I solve this?? Let $A_0 = \left.\frac{\partial \vec{g}}{\partial \vec{u}}\right|_{\vec{u}_0}$

Want: $A_0(\vec{u} - \vec{u}_0) \approx -\vec{g}(\vec{u}_0)$

Call $\vec{\tilde{u}} = \vec{u} - \vec{u}_0$ $\qquad A_0 \vec{\tilde{u}} \approx -\vec{g}(\vec{u}_0)$

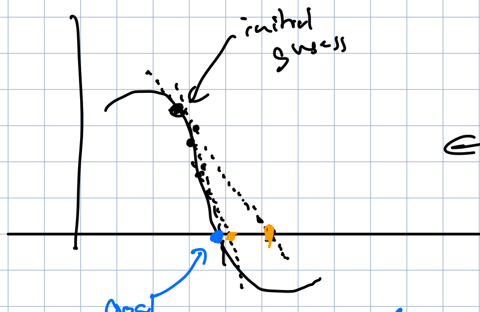
Can solve using least-squares or min-norm approaches depending on whether A_0 is tall or wide.

Suppose it were like least-squares. So A_0 is tall or square.

After I solve this, I'll get some $\vec{\tilde{u}}_1$.

I'll actually set $\vec{u}_1 = \vec{u}_0 + \alpha \cdot \vec{\tilde{u}}_1$.

Move a little bit in this direction
since approximation is local.



to find a zero of the function.

Iterate....
taking little steps

Can combine with the "transpose heuristic" ← From the Homework to iteratively solve least-squares.

$$\vec{u}_{i+1} = \vec{u}_i - \alpha A_i^T g(\vec{u}_i)$$

$$\text{where} \quad A_i = \left.\frac{\partial \vec{g}}{\partial \vec{u}}\right|_{\vec{u}_i}.$$

"Treat A^T as a cheap proxy for A^{-1}"

Often works to get you to a solution in the neighborhood of initial guess.

<u>Next time:</u> Where does this "transpose heuristic" come from?

If we have a way of solving systems of nonlinear equations, then we can find local minima & maxima.

Find places where no matter which direction you move in, nothing gets better.

Gives rise to a system of equations.