> Welcome to 16.90 iSession ...

- Instructor: Turn on Webex ,
  and distribute MuddyCards ...

> Students: Please LOG OUT from your

Facebook

Twitter

Google+

Foursquare

Email

Messenger

...etc...

...etc...

...etc...

Reading recap

- Stiffness
  - Orders of magnitude difference in timescales
  - Eigenvalues determines time scale $\frac{dy}{dt} = \lambda u$
- Newton Raphson
  - Implicit schemes benefit stiff ODEs.
  - Each step requires solving a nonlinear algebraic equation
  - Newton Raphson solves such nonlinear algebraic equations

$$\frac{dy}{dt} = \lambda u$$

$$V^{n+1} = V^{n-1} + 2\Delta t \left(\lambda V^{n} \right)$$

$$\begin{cases} V^{n+1} = V^{0} z^{n+1} \\ V^{n} = V^{0} z^{n} \\ V^{n-1} = V^{0} z^{n-1} \end{cases}$$

$$|z| = \sqrt{Re(z)^2 + Im(z)^2}$$

A transient thermal problem

- **An aluminum plate and a copper plate initially at room temperature.**
- **One plate Heated by hot air at time t=t0.**
- **Predict the measured temperature as a function of time.**
- **Is it a stiff problem? Why or why not?**

$$\dot{Q}_{cond} = kA\frac{\Delta T}{\Delta x} \qquad \Delta T = T_A - T_c$$

$$m_A C_A \frac{\partial T_A}{\partial t} = \dot{Q}_{conv} - \dot{Q}_{cond}$$

A

$$\frac{\partial T_A}{\partial t} = \frac{kA}{m_A C_A}\frac{(T_A - T_c)}{\Delta x} + \frac{hA(T_{air} - T_A)}{m_A C_A}$$

$$\frac{\partial T_c}{\partial t} = \frac{kA}{m_e C_c}\frac{T_A - T_c}{\Delta x} - \frac{hA(T_c - T_{can})}{m_c C_c}$$

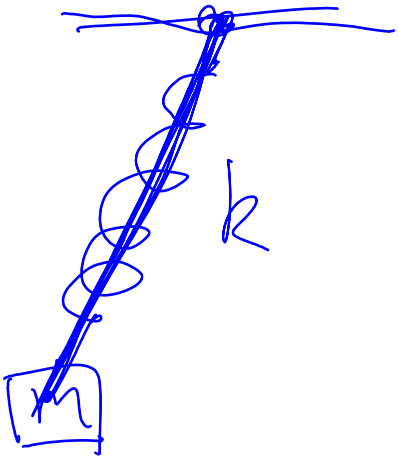
$$\frac{\partial}{\partial t}\begin{pmatrix} T_A \\ T_c \end{pmatrix} = \begin{pmatrix} -L-S & +L \\ L & -L-S \end{pmatrix}\begin{pmatrix} T_A \\ T_c \end{pmatrix}$$

$$\text{eig}\begin{pmatrix} -L & L \\ L & -L \end{pmatrix}$$

$$L \cdot \text{eig}\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} = 0, -2L$$

slow fast

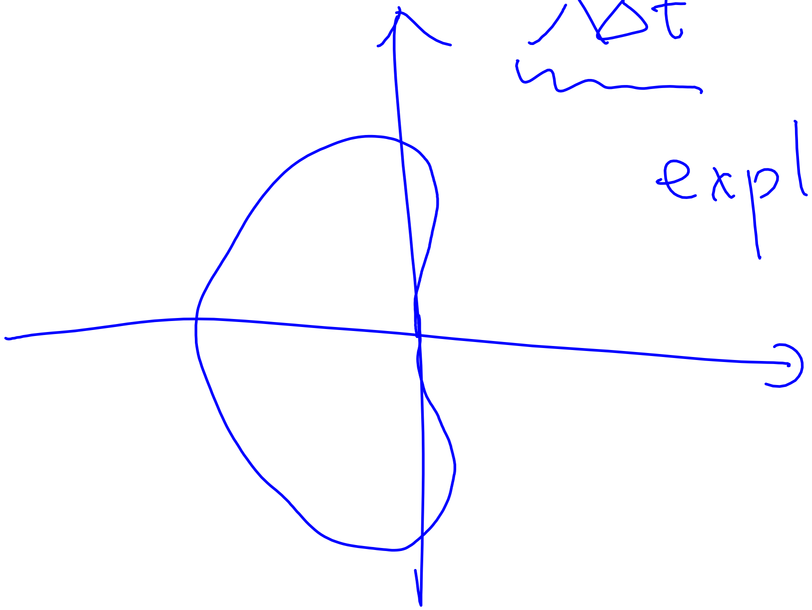
$$\text{eig}\begin{pmatrix} -L-S & L \\ L & -L-S \end{pmatrix} = -S, -2L-S$$

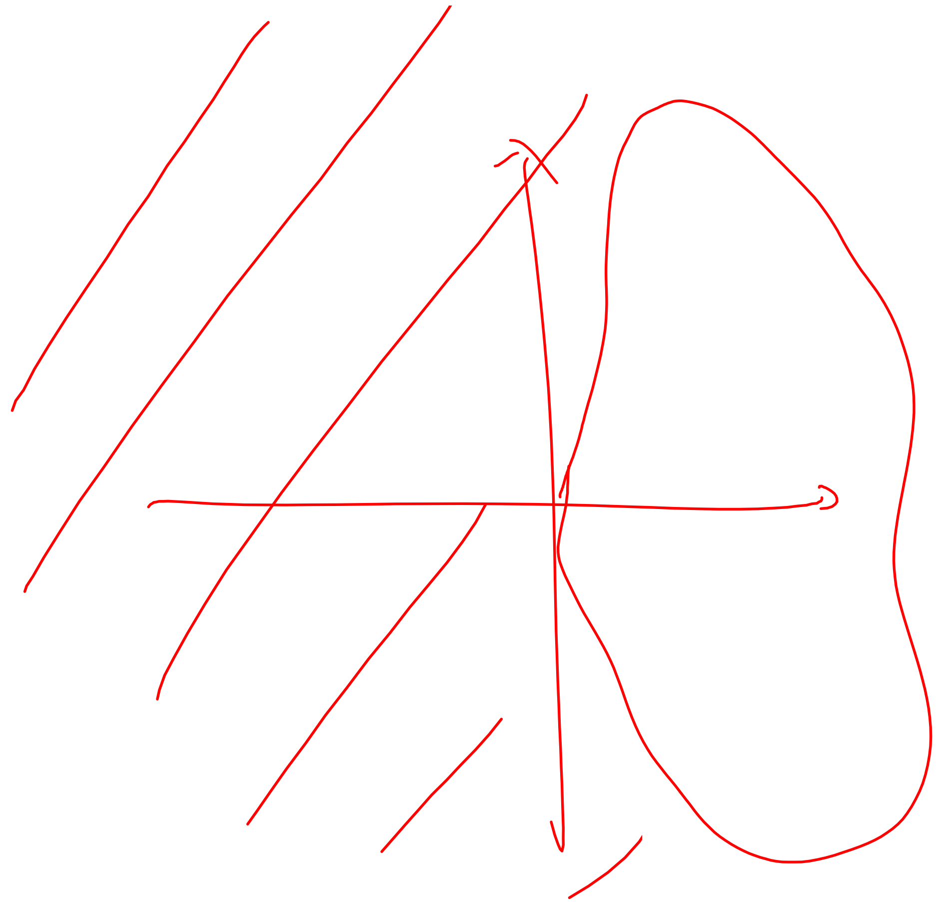

k
m

Newton Raphson for implicit scheme

$\lambda \Delta t$

explicit





$$\frac{\underbrace{V^{n+1}} - V^{n}}{\Delta t} = f(\underbrace{V^{n+1}})$$

Linear: $f(V^{n+1}) = A V^{n+1}$

$$\left(\frac{I}{\Delta t} - A\right) V^{n+1} = \frac{V^{n}}{\Delta t}$$

$\delta V = V^{n+1} - V^{n}$

Nonlinear $\quad f(V^{n+1}) \approx \underbrace{\cancel{f(V^{n})}}_{\text{known}} + \underbrace{\left(\frac{\partial f}{\partial V}\Big|_{V^{n}}\right)}_{\text{known}} \cdot \underbrace{\delta V}_{\text{unknown}}$

$\text{lt} \# 1$

$$\frac{\Delta V}{\Delta t} = f(V^n) + \left(\frac{\partial f}{\partial V}\bigg|_{V^n} \right) \Delta V$$

$$\left(\frac{1}{\Delta t} - \left(\frac{\partial f}{\partial V} \right) \right) \Delta V = f(V^n)$$

$$\tilde{V}^{n+1} = V^n + \Delta V$$

$$\frac{\tilde{V}^{n+1} - V^n}{\Delta t} \stackrel{?}{=} f(\tilde{V}^{n+1})$$ if not good enough

$$V^{n+1} = \tilde{V}^{n+1} + \Delta V \leftarrow \text{new } \delta V$$

It #2

$$f(V^{n+1}) = f(\hat{V}^{n+1}) + \left(\frac{\partial f}{\partial V}\bigg|_{\hat{V}^{n+1}}\right) \cdot \Delta V$$

$$\frac{\hat{V}^{n+1} - V^{n} + \Delta V}{\Delta t} = f(\hat{V}^{n+1}) + \left(\quad\right) \Delta V$$

$$\left(\frac{I}{\Delta t} - \left(\frac{\partial f}{\partial V}\right)\right) \Delta V = \underline{f(\hat{V}^{n+1}) - \frac{\hat{V}^{n+1} - V^{n}}{\Delta t}}$$

$$\hat{\hat{V}}^{n+1} = \hat{V}^{n+1} + \Delta V$$

$$A \cdot \Delta V = R \qquad \Delta V = A^{-1} R$$

$$\delta V = \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{pmatrix} \qquad R = \begin{pmatrix} R_1 \\ \vdots \\ R_n \end{pmatrix}$$

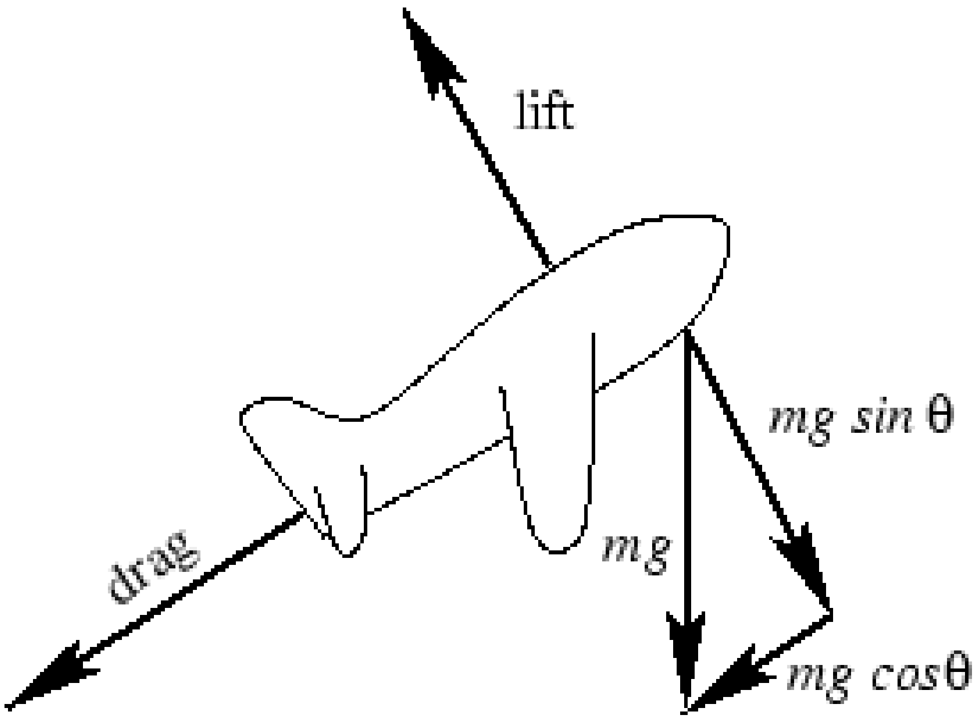
$$A = \begin{pmatrix} \frac{\partial R_1}{\partial V_1} & \cdots & \frac{\partial R_1}{\partial V_n} \\ \vdots & & \\ \frac{\partial R_n}{\partial V_1} & \cdots & \frac{\partial R_n}{\partial V_n} \end{pmatrix}$$

$$\delta V = \underline{A^{-1} R} \qquad A \backslash R \equiv A^{-1} R$$

A real example of stiff system

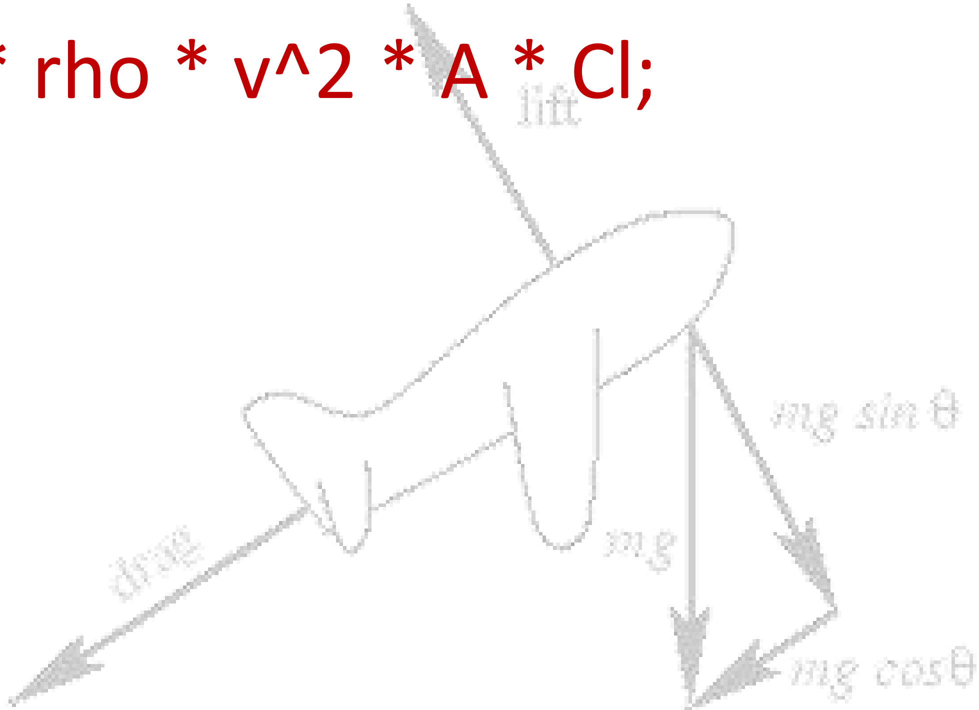
Let’s build a **Matlab flight simulator**

- Cl = 2 * pi * alpha; % angle of attack



Let's build a **Matlab flight simulator**

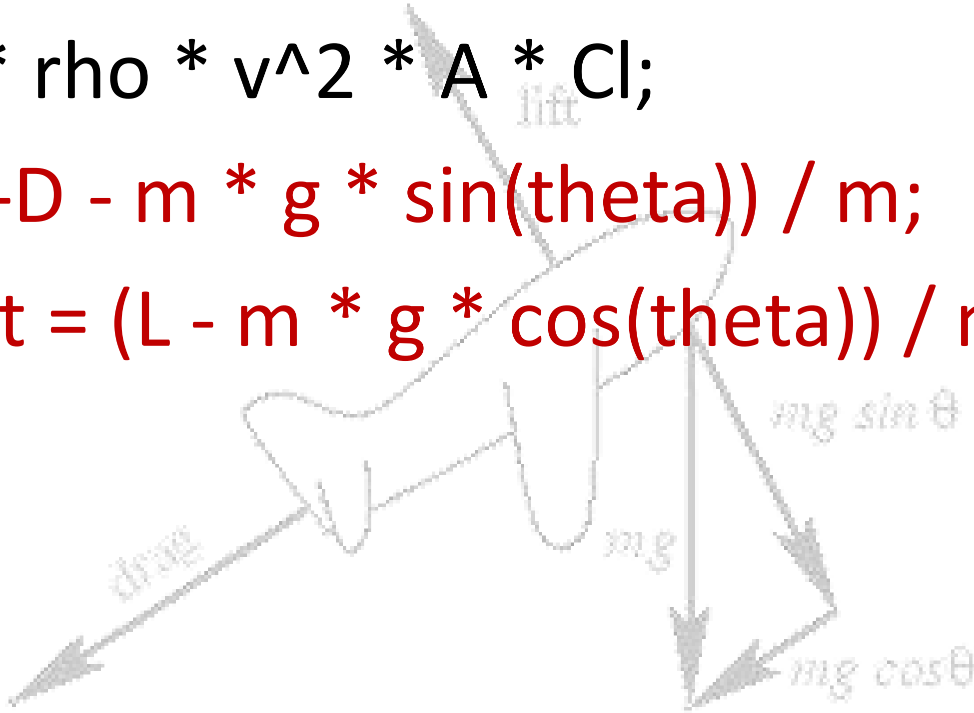
- Cl = 2 * pi * alpha; % angle of attack
- D = 0.5 * rho * v^2 * A * Cd;
- L = 0.5 * rho * v^2 * A * Cl;



Let’s build a **Matlab flight simulator**

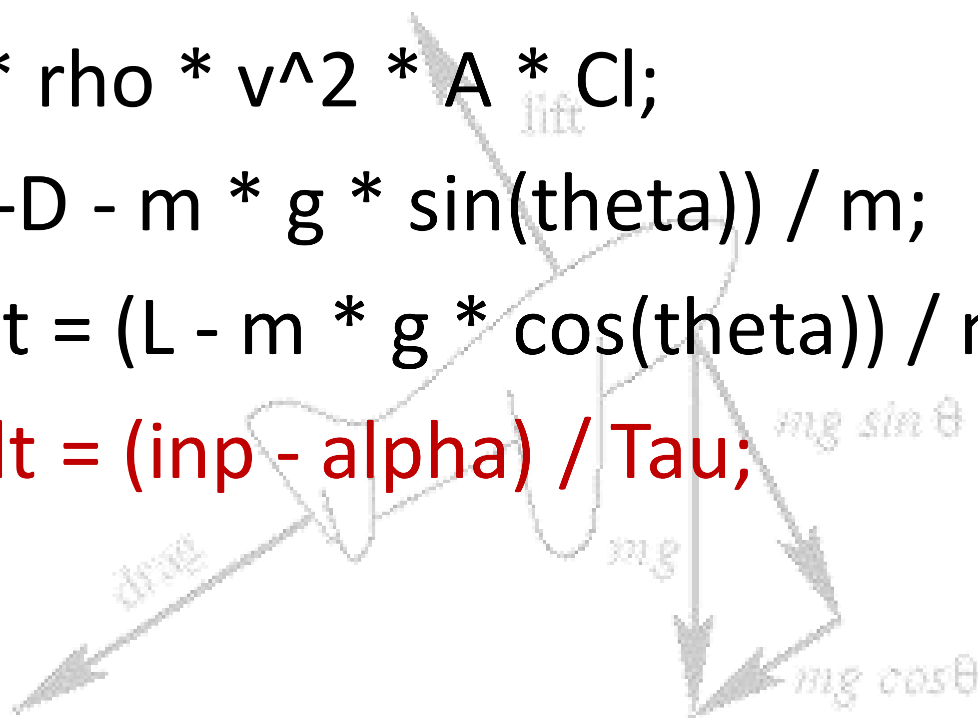
Dynamics

- Cl = 2 * pi * alpha;   % angle of attack
- D = 0.5 * rho * v^2 * A * Cd;
- L = 0.5 * rho * v^2 * A * Cl;
- dvdt = (-D - m * g * sin(theta)) / m;
- dthetadt = (L - m * g * cos(theta)) / m / v;



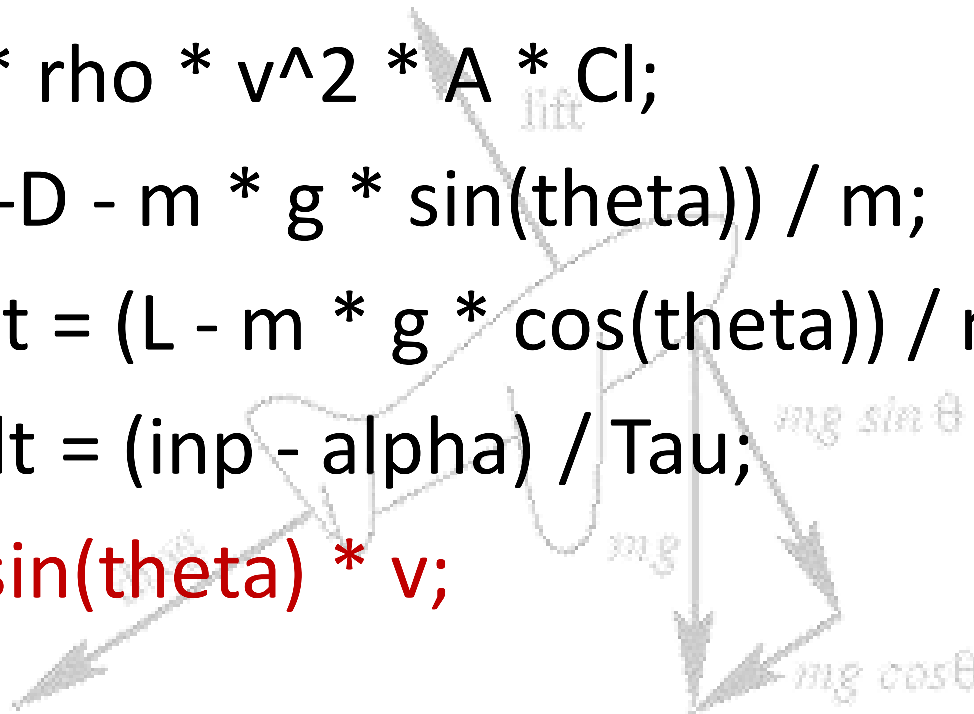
Let’s build a Matlab flight simulator
Elevator trim

- Cl = 2 * pi * alpha;   % angle of attack
- D = 0.5 * rho * v^2 * A * Cd;
- L = 0.5 * rho * v^2 * A * Cl;
- dvdt = (-D - m * g * sin(theta)) / m;
- dthetadt = (L - m * g * cos(theta)) / m / v;
- dalphadt = (inp - alpha) / Tau;



Let's build a Matlab flight simulator
Altitude

- Cl = 2 * pi * alpha;   % angle of attack
- D = 0.5 * rho * v^2 * A * Cd;
- L = 0.5 * rho * v^2 * A * Cl;
- dvdt = (-D - m * g * sin(theta)) / m;
- dthetadt = (L - m * g * cos(theta)) / m / v;
- dalphadt = (inp - alpha) / Tau;
- dhdt = sin(theta) * v;



Eigenvalues of the linearized system

16.90 Computational Methods in Aerospace Engineering
Spring 2014