Overview of Rotorcraft Modeling & Simulation

Modeling and Simulation Outline

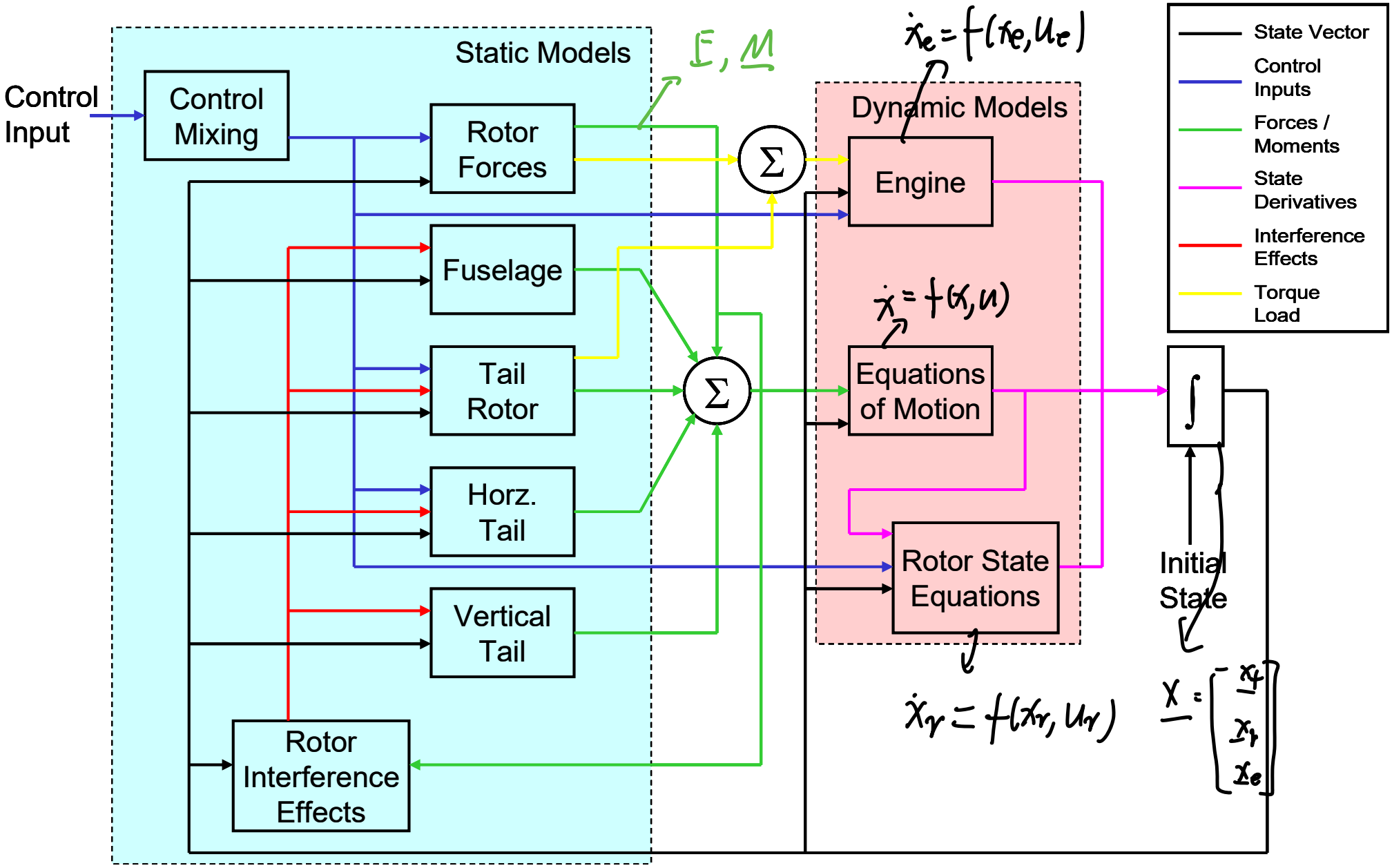
I. Review of Kinematics, Dynamics, and 6-DOF Equations of Motion
   - A. Reference Frames / Notation
   - B. Useful kinematical equations
   - C. Translational and Rotational Dynamics
   - D. Euler Angles / Kinematics

II. Overview of Rotorcraft Modeling & Simulation
   - **A. Modular Model Structure**
   - **B. Simple Helicopter Model**
   - **C. Numerical Integration**
   - **D. Numerical Linearization**
   - **E. Trim**

III. Rotor Dynamic Model
   - A. Flapping Equations of Motion
   - B. Multi-blade Coordinates
   - C. Rotor Forces and Moments
   - D. Dynamic Inflow

IV. Complete Modular Rotorcraft Simulation Model
   - A. Tail Rotor
   - B. Fuselage and Empennage Aerodynamics
   - C. Engine Dynamics
   - D. Flight Controls
   - E. Integrated Simulation Model
   - F. Other Configurations (tandem rotors, tilt-rotors, compounds, multi-copters)

HeloSim Model Structure

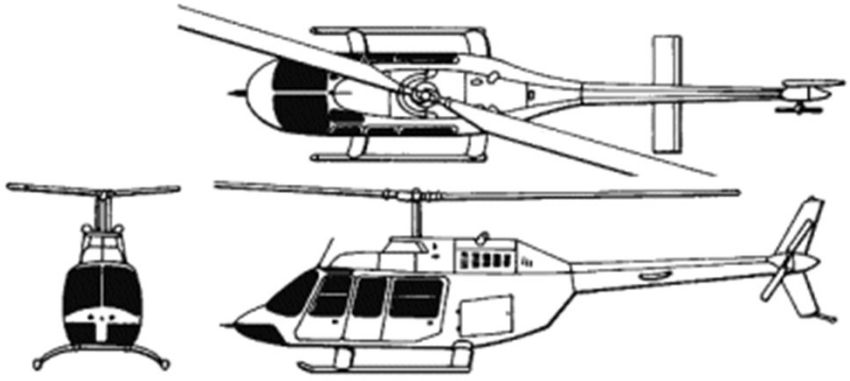


Simple Helicopter Model

- Before developing the HeloSim model, I will present basic simulation methods and principles with a highly simplified rotorcraft model
  - 6 DOF and 12 State model, no rotor or inflow dynamics
  - Aerodynamic forces and moments are static function of aircraft state and controls using simple models
- Demonstrate basic simulation functionality
  - Numerical integration
  - Trim
  - Linearization
- The same functions will be extended to more complex HeloSim model later in the course

Simple Helicopter Model

- Based loosely on the Bell 206L3
- Properties:
  - GW = 4000 lbs
  - Two-blade teetering rotor system
  - Main rotor radius = 18 ft
  - Tail rotor radius = 2.7 ft



Simple Helo Model Structure

- As with all simulations implemented in this class, model is set up in time invariant state variable form:

- State vector $\mathbf{x} = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta & \psi & x & y & z \end{bmatrix}^T$

- Control vector $\mathbf{u} = \begin{bmatrix} \theta_{1c} & \theta_{1s} & \theta_0 & \theta_{0_T} \end{bmatrix}^T$ ~ deg

  θ_{1c}: lateral cyclic; θ_{1s}: long. cyclic; θ_0: MR collective; θ_{0_T}: tail rotor collective.

- MATLAB functional call of state equations:

  [xdot,y] = SimpleHelo(x,u,xdot,constants);

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

  $y = g(x,u)$

  x, u, y, xdot are state, control, output, and state derivative vectors respectively

  output can be anything you wanna to be (sensors, usually)

  Data structure with rotorcraft constants

- Note we use implicit state space form, but SimpleHelo.m is an explicit state space model. *It does not use xdot in the input.*
- We keep this format for consistency with more complex models used later in the class

Simple Helo Model Structure

- SimpleHelo.m calls two functions:
  - SimpleAero.m – Calculates aerodynamic forces and moments
  - EqnMot.m – Calculates 6 DOF equations of motion
- EqnMot.m function:

```
x_dot=eqnmot(x,FM,constants);
```

Returns state derivative

Inputs are states, force and moment vector, and vehicle constants (it needs mass properties)

- SimpleAero.m function:

```
[FM,y]=SimpleAero(x,u,constants);
```

Returns total forces and moments at CG, and output vector of users choice (I just set to return specific forces).

Inputs are states, control inputs, and vehicle constants

6-DOF Rigid Body Equations of Motion (in EqnMot.m)

$$\dot{u} = \frac{X}{m} - g\sin\theta - qw + rv$$

$$\dot{v} = \frac{Y}{m} + g\cos\theta\sin\phi - ru + pw$$

$$\dot{w} = \frac{Z}{m} + g\cos\theta\cos\phi - pv + qu$$

$$\begin{bmatrix} X \\ Y \\ Z \\ L \\ M \\ N \end{bmatrix} = \text{static function of} \ldots ?$$

$$\dot{p} = \frac{1}{I_x I_z - I_{xz}^2}\left(I_z L + I_{xz} N + I_{xz}\left(I_x - I_y + I_z\right)pq - \left(I_z^2 - I_z I_y + I_{xz}^2\right)qr\right)$$

$$\dot{q} = \frac{1}{I_y}\left(M - \left(I_x - I_z\right)rp - I_{xz}\left(p^2 - r^2\right)\right)$$

$$\dot{r} = \frac{1}{I_x I_z - I_{xz}^2}\left(I_x N + I_{xz} L - I_{xz}\left(I_x - I_y + I_z\right)qr + \left(I_x^2 - I_x I_y + I_{xz}^2\right)pq\right)$$

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta$$

$$\dot{\theta} = q\cos\phi - r\sin\phi$$

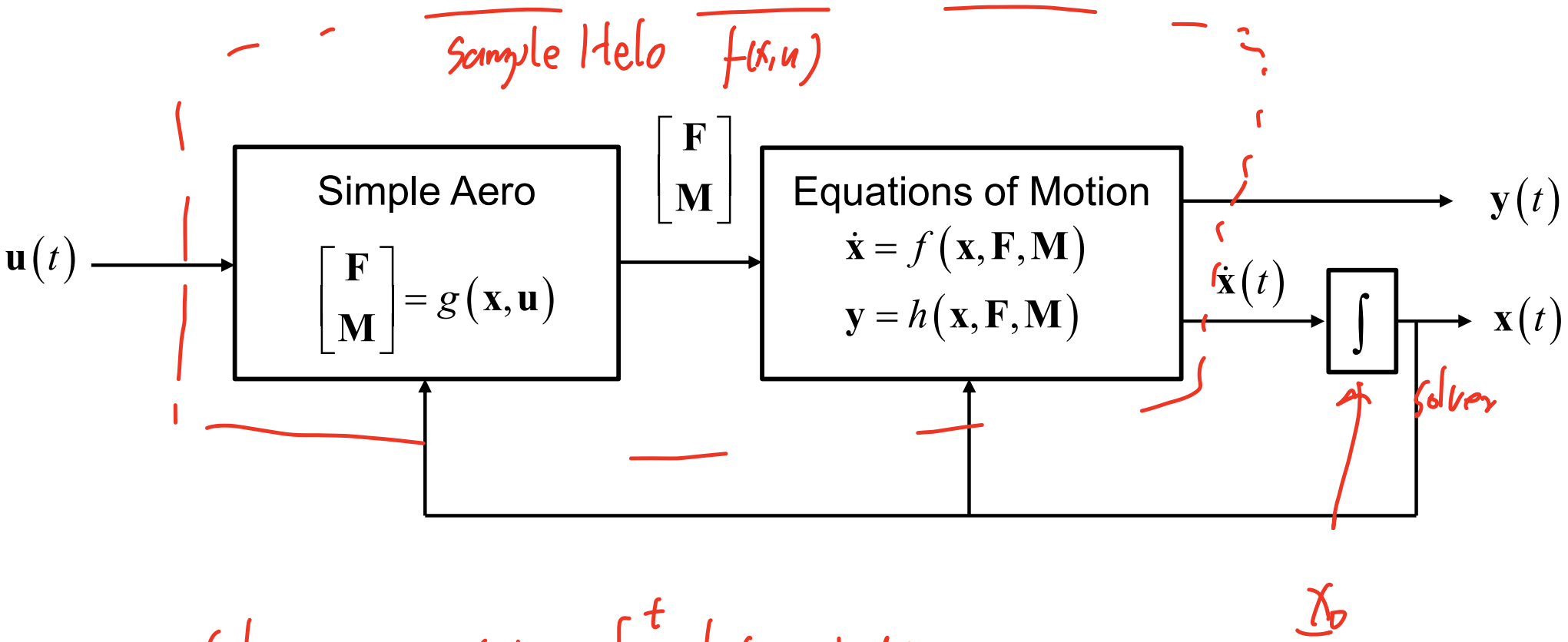
$$\dot{\psi} = q\sin\phi/\cos\theta + r\cos\phi/\cos\theta$$

$$\dot{x} = u\cos\theta\cos\psi + v\left(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi\right) + w\left(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\right)$$

$$\dot{y} = u\cos\theta\sin\psi + v\left(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi\right) + w\left(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\right)$$

$$\dot{z} = -u\sin\theta + v\sin\phi\cos\theta + w\cos\phi\cos\theta$$

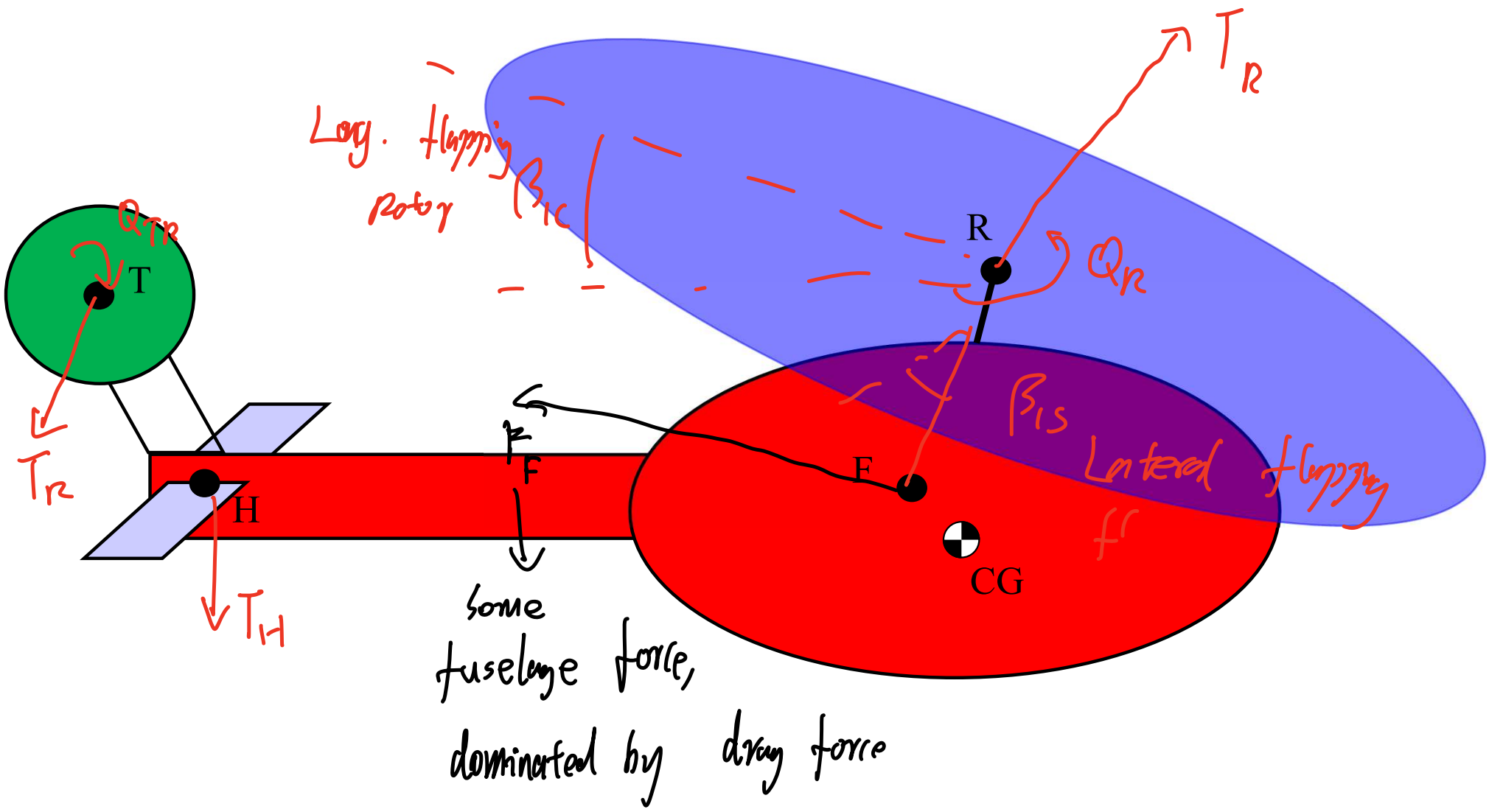
Overall Structure of Simple Helo Model



Solver: $\underline{X}(t) = \int_{t_0}^{t} f(\underline{x}, \underline{u})\, dt$

$\underline{X}(0) =$ I.C. from Trim Condition

Aerodynamic Component Models



Resultant Force and Moment Vectors at CG

$$\mathbf{F} = \mathbf{F}_R + \mathbf{F}_F + \mathbf{F}_T + \mathbf{F}_H$$

$$\mathbf{M} = \left(\mathbf{M}_R + \mathbf{r}_{R/CG} \times \mathbf{F}_R\right) + \left(\mathbf{M}_F + \mathbf{r}_{F/CG} \times \mathbf{F}_F\right) + \left(\mathbf{M}_T + \mathbf{r}_{T/CG} \times \mathbf{F}_T\right) + \left(\mathbf{M}_H + \mathbf{r}_{H/CG} \times \mathbf{F}_H\right)$$

=0, C.G. ≈ A.C.

=0

Local Velocities at Each Component

Local velocity of component relative to Earth frame:

For some component "X" (X can be R, F, T, H):

$$\mathbf{V}_{X/e} = \mathbf{V}_{G/e} + \boldsymbol{\omega}_{g/e} \times \mathbf{r}_{X/CG}$$

With wind or induced flow:

$$\mathbf{V}_{X/a} = \mathbf{V}_{X/e} - \mathbf{V}_{A/e}$$

Will use to include induced flow effects on fuselage and horizontal tail

(get correct AoA.)

Point A is point conincident with X at an instant in time, but fixed in the airmass frame F_a

Fuselage Aero

- Assume a simple force model due to drag forces, from Stevens, Lewis, and Jonson:

$$\mathbf{F}_F^g = -\tfrac{1}{2}\rho \begin{bmatrix} S_{X_F}\left|u_f\right|u_f \\ S_{Y_F}\left|v_f\right|v_f \\ S_{Z_F}\left|w_f\right|w_f \end{bmatrix}$$

S_{X_F} = Frontal Drag Area

S_{Y_F} = Side Drag Area

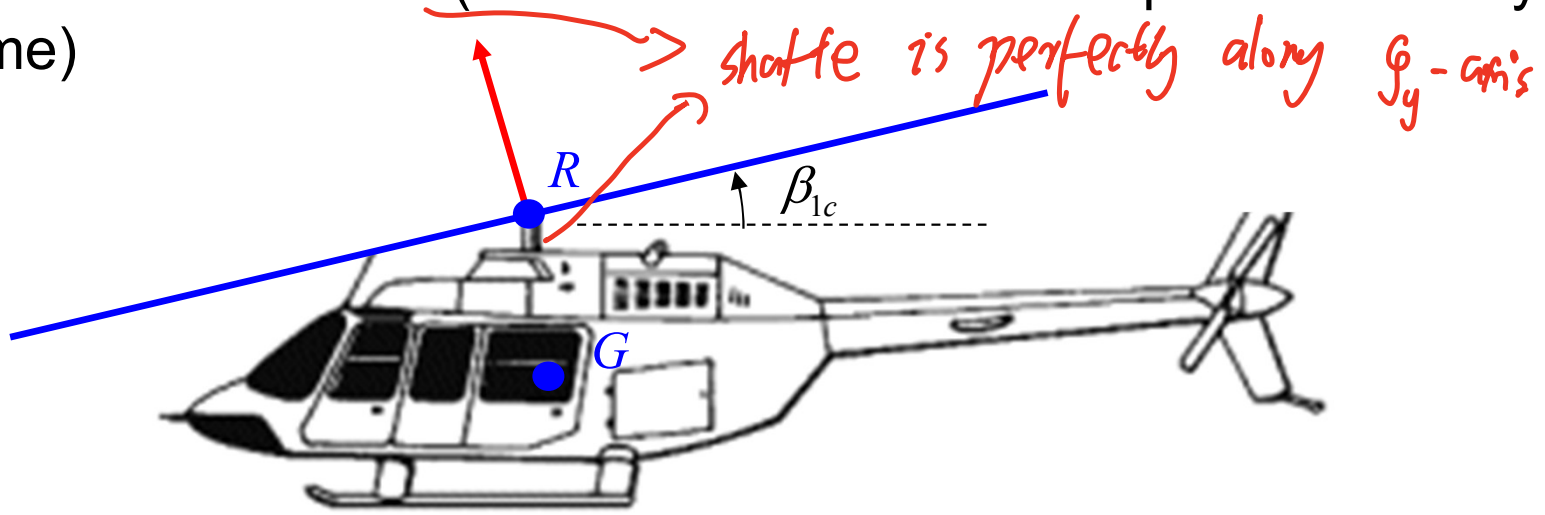
S_{Z_F} = Vertical Drag Area

$$w_f = w - v_{i_F}$$

- Neglect fuselage aerodynamic moments. Allows for moments due to distance from fuselage aero center to CG (which is set to 0 in this sim)

Main Rotor Model

- Assumptions:
  - Teetering rotor system (no hub moment)
  - Small flapping angles
  - "Actuator Disk" model: use simplified model to compute total thrust and torque of the main rotor
  - No flapping or inflow dynamics → means we have Happy and inflow, but just static.
  - Zero shaft incidence (Main Rotor Hub Frame is parallel to body frame)



Main Rotor Model

- Local velocity of main rotor hub:

$$\mathbf{V}_{R/e} = \mathbf{V}_{G/e} + \boldsymbol{\omega}_{g/e} \times \mathbf{r}_{R/CG} = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix}$$

- In-plane and perpendicular components expressed as non-dimensional advance ratios:

$$\mu = \text{Advance Ratio} = \frac{\sqrt{u_r^2 + v_r^2}}{\Omega R}, \mu_Z = \text{Vertical advance Ratio} = \frac{w_r}{\Omega R}$$

inplane vertical

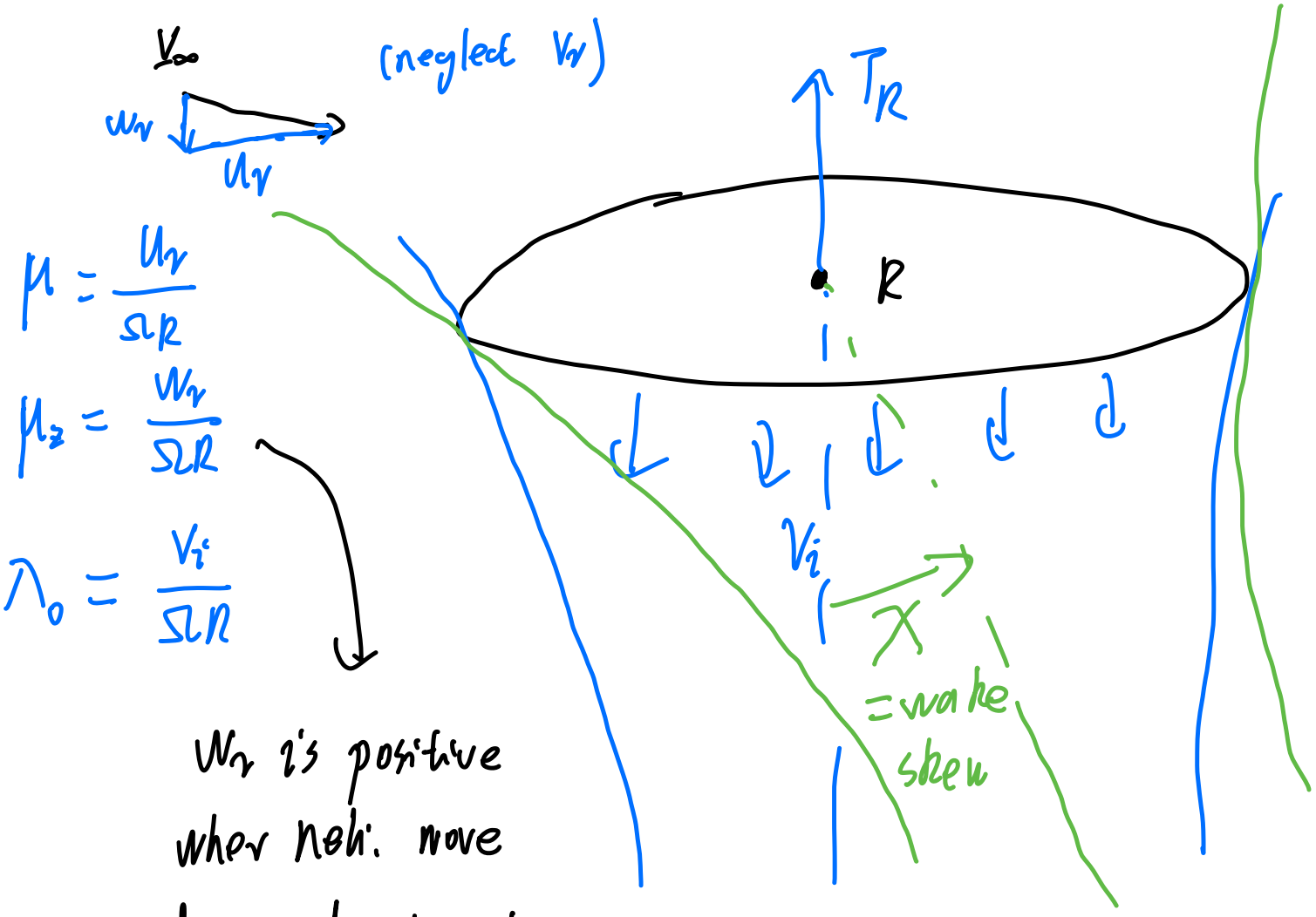
- Inflow and Main rotor wake skew

$$\chi = \tan^{-1}\left(\frac{\mu}{\lambda_0 - \mu_Z}\right) = \text{ Wake skew angle}$$

$$\lambda_0 = \text{Inflow ratio at rotor disk} = \frac{v_{i_R}}{\Omega R} \text{ from momentum theory}$$

Main Rotor Model

- Rotor inflow sketch



Main Rotor Model

- Rotor flapping and rotor forces:

Solved through iteration

$$C_T = f_T\left(\mu, \mu_Z - \lambda_0, \theta_0, \theta_{1s}\right) \qquad \beta_{1c} = f_{\beta_{1c}}\left(\mu, \mu_Z - \lambda_0, \theta_0, \theta_{1c}, \theta_{1s}, p, q\right)$$

$$C_Q = f_Q\left(\mu, \mu_Z - \lambda_0, \theta_0, \theta_{1s}\right) \qquad \beta_{1s} = f_{\beta_{1s}}\left(\mu, \mu_Z - \lambda_0, \theta_0, \theta_{1c}, \theta_{1s}, p, q\right)$$

$$\lambda_0 = f_\lambda\left(C_T, \mu, \mu_Z\right)$$

$$T = C_T\left(\rho\left(\Omega R\right)^2\left(\pi R^2\right)\right)$$

$$Q = C_Q\left(\rho\left(\Omega R\right)^2\left(\pi R^3\right)\right)$$

- Resultant Forces and Moments due to the main rotor at the CG

$$\mathbf{F}_R^g = \begin{bmatrix} T\beta_{1c} \\ -T\beta_{1s} \\ -T \end{bmatrix} \qquad \mathbf{M}_R^g = \begin{bmatrix} 0 \\ 0 \\ Q \end{bmatrix} + \mathbf{r}_{R/CG} \times \mathbf{F}_R$$

Tail Rotor Model

- Same model as main rotor, but assume no flapping
- Tail rotor frame is rotated 90 deg relative to body frame. Resulting coordinate transformation:

$$T_{t/g} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Local velocities in-plane and vertical velocities at tail rotor:

$$\mathbf{V}^t_{T/e} = T_{t/g}\left(\mathbf{V}^g_{G/e} + \boldsymbol{\omega}^g_{g/e} \times \mathbf{r}^g_{T/CG}\right) = \begin{bmatrix} u_t \\ v_t \\ w_t \end{bmatrix}$$

$$\mu_T = \frac{\sqrt{u_t^2 + v_t^2}}{\Omega_T R_T}, \mu_{Z_T} = \frac{w_t}{\Omega_T R_T}$$

Tail Rotor Model

- Tail rotor thrust and torque, similar to main rotor but simplified (e.g. no cyclic pitch):

$$C_T = f_T\left(\mu, \mu_Z - \lambda_0, \theta_0\right)$$
$$C_Q = f_Q\left(\mu, \mu_Z - \lambda_0, \theta_0\right)$$
$$\lambda_0 = f_\lambda\left(C_T, \mu, \mu_Z\right)$$

- Resulting forces and moments due to tail rotor at CG:

$$\mathbf{F}_T^t = \begin{bmatrix} 0 \\ 0 \\ -T_T \end{bmatrix} \qquad \mathbf{M}_T^t = \begin{bmatrix} 0 \\ 0 \\ Q_T \end{bmatrix}$$

$$\mathbf{M}_T^g = T_{g/t}\mathbf{M}_T^t + \mathbf{r}_{T/CG}^g \times T_{g/t}\mathbf{F}_T^t$$

=> will produce yaw moment in body frame.

Horizontal Stabilizer Model

- Assumptions:
  - Lift and drag forces only (no moments or side force)
  - No incidence (HT frame is parallel to body frame)
- Local velocity at horizontal stabilizer:

$$\mathbf{V}_{H/e}^{g} = \mathbf{V}_{H/e}^{g} + \boldsymbol{\omega}_{g/e}^{g} \times \mathbf{r}_{H/CG}^{g} = \begin{bmatrix} u_h \\ v_h \\ w_h \end{bmatrix}$$

这里 要推导得自己注一下.

- Angle of attack and local dynamic pressure:

$$\mathbf{V}_{H/a}^{g} = \mathbf{V}_{H/e}^{g} - \begin{bmatrix} 0 \\ 0 \\ v_{i_H} \end{bmatrix}, \; V_H^2 = u_h^2 + \left(w_h - v_{i_H} \right)^2, \; q_{HT} = \tfrac{1}{2}\rho V_H^2, \alpha_H = \operatorname{atan2}(w_h - v_{i_H}, u_h)$$

Horizontal Stabilizer Model

- Consider only lift forces, lift coefficient linear with angle of attack but limited to +/- 1.0

$$C_{L_H} = \max(\min(a_H \alpha_H, 1.0), -1.0)$$

- Resultant forces and moments due to horizontal tail at aircraft CG

$$\mathbf{F}_H^g = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{2}\rho V_H^2 S_H C_{L_H} \end{bmatrix}$$

$$\mathbf{M}_H^g = \mathbf{r}_{H/G}^g \times \mathbf{F}_H^g$$

→ Pitch moment due to lift.

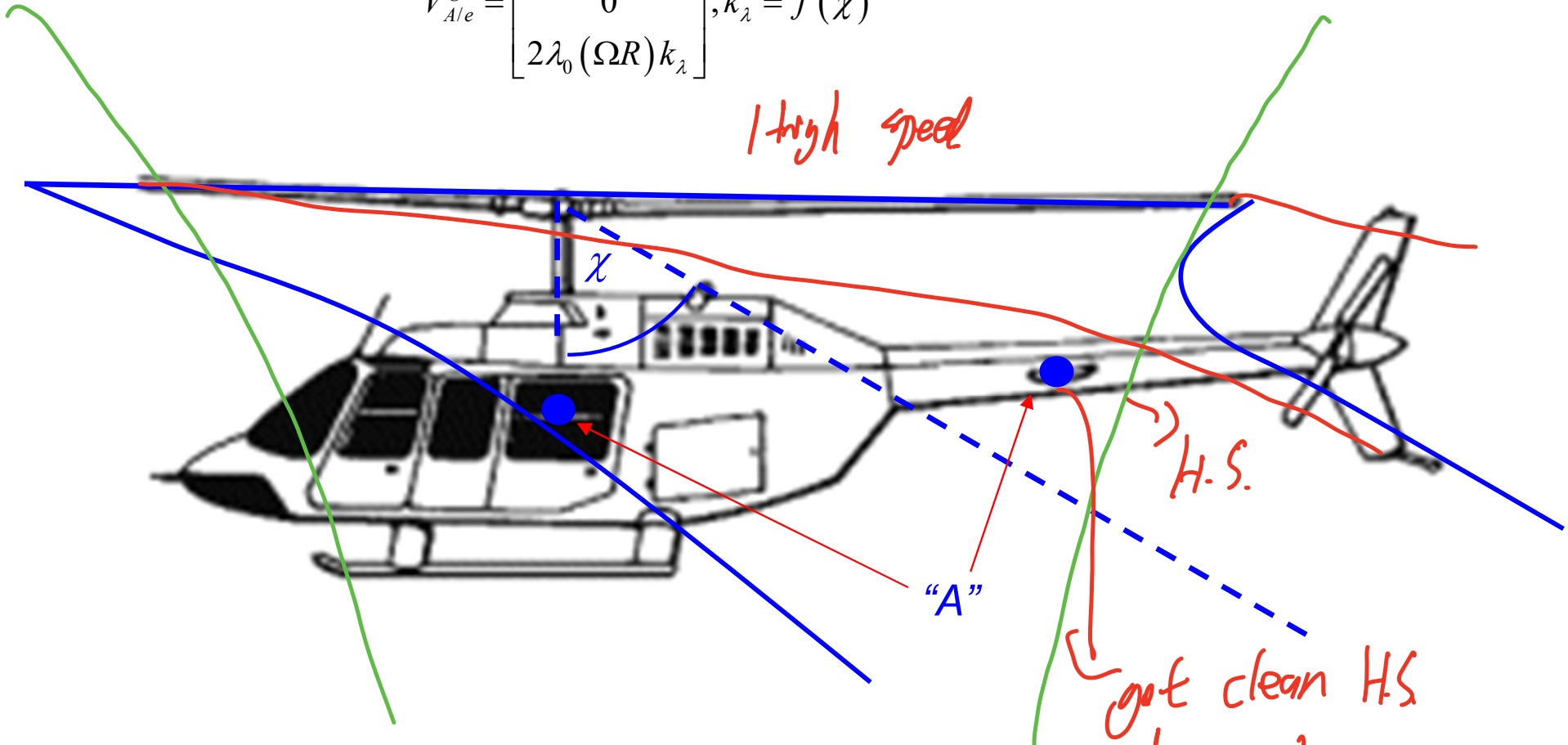
Aero Interference Effects

- Downwash velocities on fuselage and horizontal tail due to main rotor:

$$\chi = \tan^{-1}\left(\frac{\mu}{\lambda_0 - \mu_Z}\right) = \text{ Wake skew angle}$$

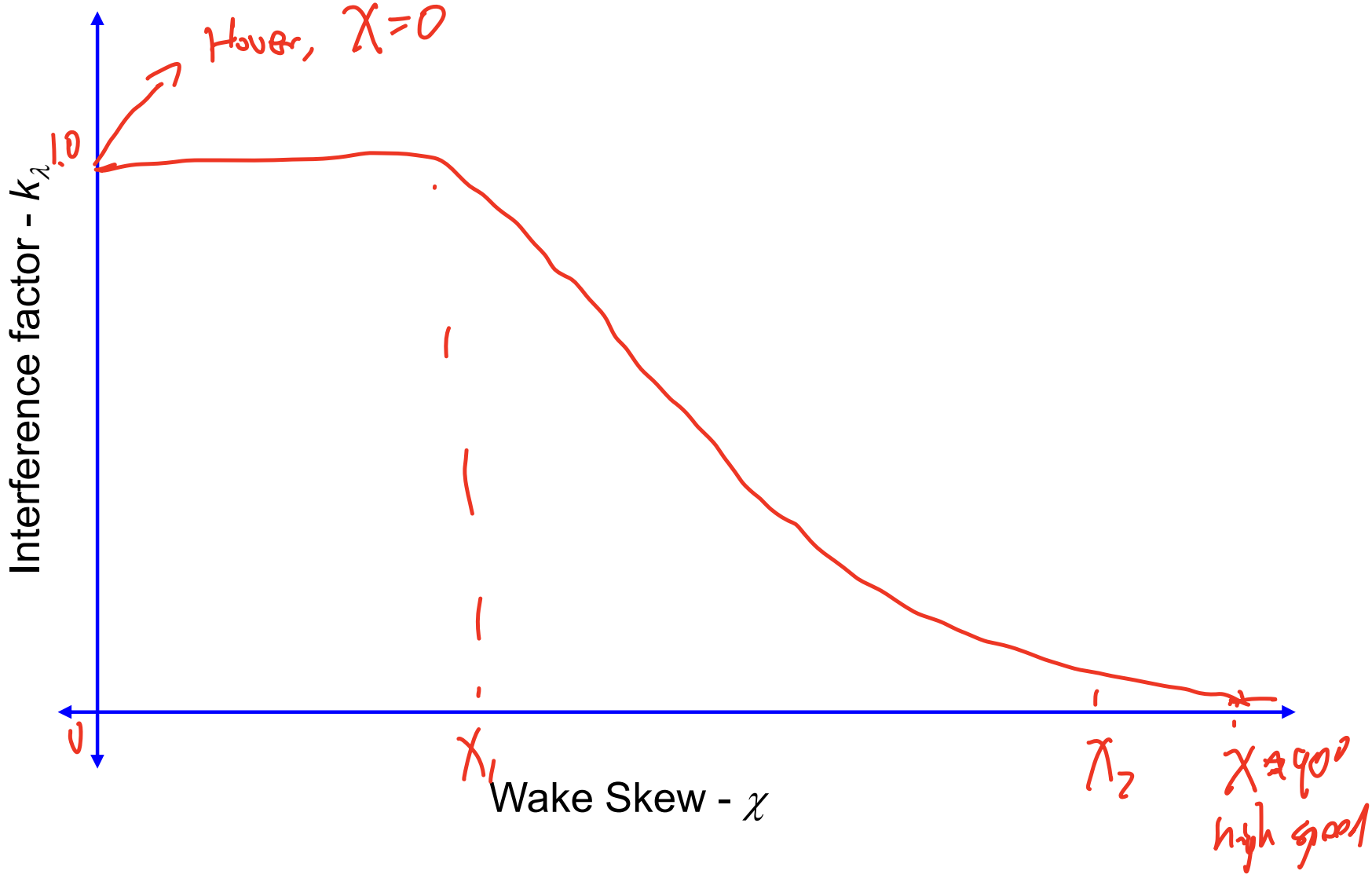
$$\lambda_0 = \text{Inflow ratio at rotor disk} = \frac{v_{i_R}}{\Omega R}$$

$$V_{A/e}^{g} = \begin{bmatrix} 0 \\ 0 \\ 2\lambda_0 (\Omega R) k_\lambda \end{bmatrix}, k_\lambda = f(\chi)$$



downwash once goes high speed.

Aero Interference Effects



Integration of the State Equations

- Numerical solution of the state equations given initial conditions and prescribed control inputs:

$$\dot{x} = f\left(x, \dot{x}, u\right) \qquad \text{Find: } x(t), y(t) \text{ for } t > 0$$

$$y = g\left(x, u\right) \qquad \text{Given: } x(0) = x_0,\ u(t) \text{ for } t > 0$$

- State variable equations in explicit form can readily be solved using various time marching methods (see Stevens, Lewis, and Johnson chapter 3.4) – will not cover in this class
  - Runge-Kutta Methods
  - Linear Multistep Methods

$$\dot{x}_k \approx \frac{x_{k+1} - x_k}{\Delta t} = f(x_k, u_k)$$

$$x_{k+1} = x_k + \Delta t \cdot f(x_k, u_k)$$

- In implicit form, numerical integration requires iteration at each step to resolve implicit state equations (e.g. fixed-point iteration)

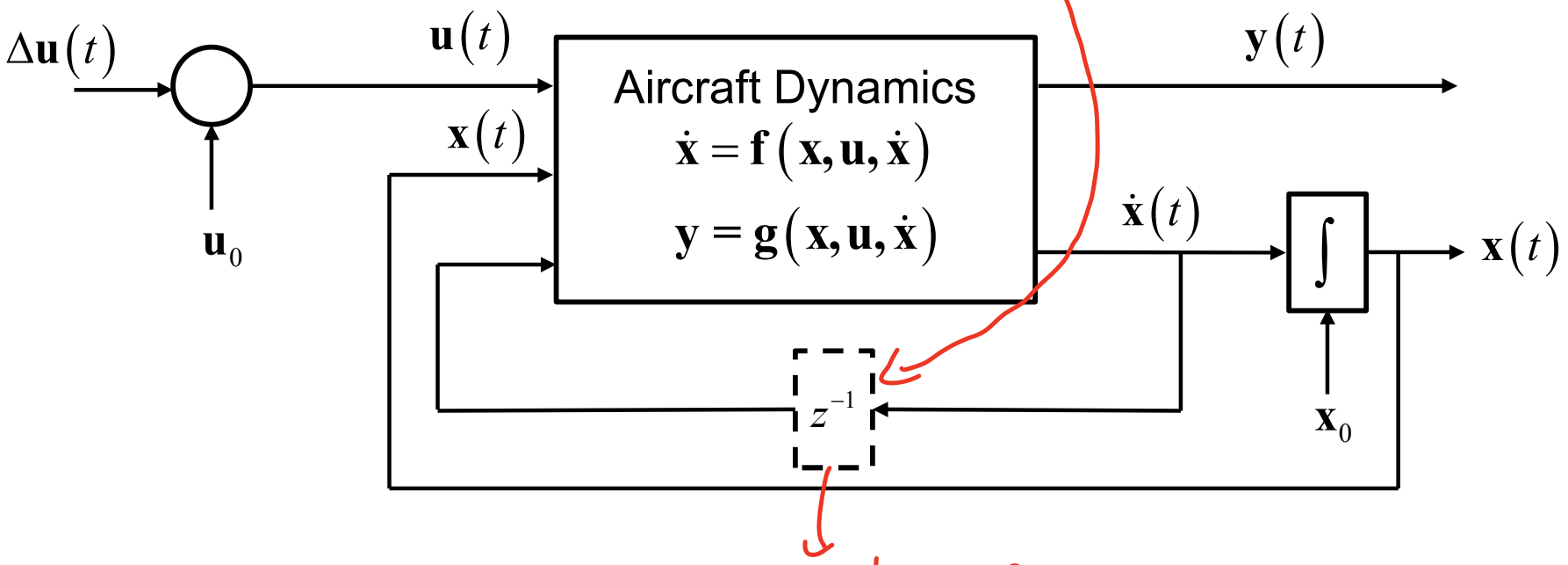
$$f'\left(x, \dot{x}, u\right) \triangleq f\left(x, \dot{x}, u\right) - \dot{x} = 0$$

- MATLAB / SIMULINK offers numerous solvers. SIMULINK is convenient as it is already set up for state equations with input signals, and it can handle "algebraic loops" which occur in implicit state eqns.

$$\text{Solves } \dot{x} = f\left(x, \dot{x}, u\right) \text{ and not just } \dot{y} = f\left(t, y\right)$$

Setup of Rotorcraft State Equations in Numerical Simulations

- General setup of a simulation can be written in block diagram as shown below.
- Algebraic loops lead to slower simulations due to the requirement to iterate on each time step
- Approximate spoliations can use a “first order hold” were model computations use state derivative from the previous time step



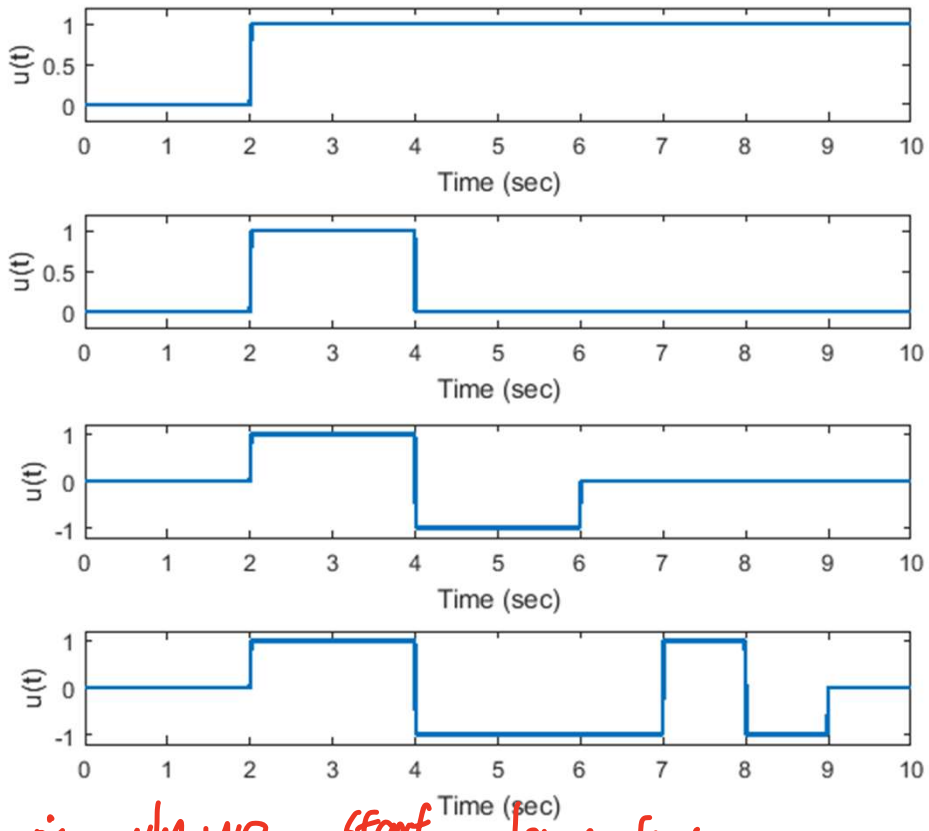
这是指这个，但你意思？

What is?

大概明白了。

Integration of State Equations

- For engineering analysis would typically use numerical solutions to analyze response to certain “canned” test inputs:
  - Step Input
  - Pulse Input
  - Doublet
  - 2-3-1-1



So I guess this is why we start from trim

- (Normally want to start the simulation from an equilibrium condition (trim) in to isolate the effect of the input (as opposed to the effect of initial conditions))
- Unstable responses are common in rotorcraft, need to constrain length of time history to have useful data when simulating unstable systems.

Generalized Trim

- We want to start most engineering simulations in trimmed flight
- Properties of generalized trimmed flight:
  - Constant airspeed

    $\dot{u} = \dot{v} = \dot{w} = 0$
  - Constant roll and pitch attitude

    $\dot{\phi} = \dot{\theta} = 0$
  - Constant turn rate

    $\dot{\psi}$ = constant ($\ddot{\psi} = 0$)
  - Constant rate of climb or descent

    $\dot{z}$ = constant
  - Constant control positions

    $\underline{u}$ = constant.

ψ has no impact on aerodynamic force and moments!)

some $\dot{x}$ = constant is also trim, rather than all $\dot{x} = 0$,

so we can have a constant acceleration, which in other words, a steady case.

Special Trim Conditions

- Rectilinear Trim: Constant heading (zero angular rates) $\Rightarrow \dot{\psi} = 0$
- Level Flight Trim: Constant altitude $\dot{z} = 0$

Trim (Equilibrium Computation)

- Given State Space System of the form:

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

- Trim seeks to solve for states and controls to meet a set of constraints on the state derivatives and output variables:

Find the equilibrium states and controls that meet the trim constraint:

$$\begin{bmatrix} \dot{x}_e \\ y_e \end{bmatrix} = \begin{bmatrix} f(x_e, \dot{x}_e, u_e) \\ g(x_e, u_e) \end{bmatrix}$$ ~ solve for x_e, u_e

- For example, the trim constraint might define a prescribed straight flight path:

$\dot u = \dot v = \dot w = \dot p = \dot q = \dot r = \dot\phi = \dot\theta = 0$

$\dot\psi$ -

γ - flight path angle

$\dot x = V\cos\gamma\cos\chi$

$\dot y = V\cos\gamma\sin\chi$

$\dot z = -V\sin\gamma$

Horn the directly 写出来, 后面八成也要熟练到这种程度.

Trim (Equilibrium Computation)

- Can set up solution by define the states and controls we want to solve (trim variables) to meet a set of constraints on the state derivatives and output variables (trim targets)
- Becomes a problem of solving a system of non-linear algebraic equations:

$$\text{Trim Variables: } \nu = \begin{bmatrix} x_e \\ u_e \end{bmatrix} \qquad \text{Trim Target: } x_T = \begin{bmatrix} \dot{x}_e \\ y_e \end{bmatrix}$$

$$\text{Trim Constriant: } \begin{bmatrix} \dot{x}_e \\ y_e \end{bmatrix} = \begin{bmatrix} f(x_e, \dot{x}_e, u_e) \\ g(x_e, u_e) \end{bmatrix} \Rightarrow \quad x_T = h(x_T, \nu)$$

- Typically, we define the problem so that the number of trim variables will equal the number of trim targets. But in some cases, there may be additional trim variables, in which case trim might become an optimization problem.

Trim (Equilibrium Computation)

- The trimmer.m routine uses a Newton-Raphson method:

$$h'(x_T, \nu) \triangleq x_T - h(x_T, \nu) = 0$$

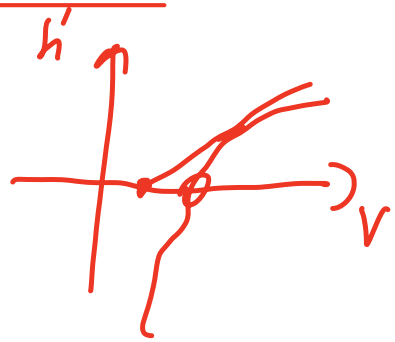
Given x_T, find ν such that $h'(x_T, \nu) = 0$

Initial Guess: ν_0 Initial Error: $\varepsilon_0 = h'(x_T, \nu_0)$

Iteration: $\nu_{k+1} = \nu_k - \lambda J_k^{-1} \varepsilon_k$

Error: $\varepsilon_k = h'(x_T, \nu_k)$ Jacobian Matrix: $J_k = \dfrac{\partial h'}{\partial \nu}(x_T, \nu_k)$

$\lambda =$ Relaxation Parameter, $0 < \lambda < 1$



- The constants structure defines the trim variables and the trim targets via the index arrays TRIMVAR and TRIMTARG:

$$\text{Trim Variables: } \nu = \begin{bmatrix} x_e \\ u_e \end{bmatrix} (\text{TRIMVAR}) \qquad \text{Trim Target: } x_T = \begin{bmatrix} \dot{x}_e \\ y_e \end{bmatrix} (\text{TRIMTARG})$$

Trim (Equilibrium Computation)

- The Jacobian is calculated via the numerical linearization scheme. It extracts a sub-matrix from the linear model matrices (will be discussed in following slides):

$$J = \begin{bmatrix} A & B \\ C & D \end{bmatrix} (\text{TRIMTARG}, \text{TRIMVAR})$$

- The constants structure also specifies the tolerances used in trim, and a vector of scale factors so that errors in each state are of comparable magnitude.
- Constants structure also specifies perturbation sizes used in numerical linearization.

Linear Models

- Linearized models are extremely useful in dynamic analysis and control design
- Starting from non-linear EOM in implicit state space

$$\dot{x} = f\left(x, u, \dot{x}\right)$$
$$y = g\left(x, u\right)$$

$x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$

- We want to extract a linearized model of the following form:

$$M\,\Delta\dot{x} = F\,\Delta x + G\,\Delta u$$
$$\Delta y = C\,\Delta x + D\,\Delta u$$

$$\Delta x = x - x_e$$
$$\Delta u = u - u_e$$
$$\Delta y = y - y_e$$

Will drop delta (Δ) notation for linear model variable in following notes

$$M = I - \left.\frac{\partial f}{\partial \dot{x}}\right|_{\substack{x=x_e \\ u=u_e}} \quad F = \left.\frac{\partial f}{\partial x}\right|_{\substack{x=x_e \\ u=u_e}} \quad G = \left.\frac{\partial f}{\partial u}\right|_{\substack{x=x_e \\ u=u_e}} \quad C = \left.\frac{\partial g}{\partial x}\right|_{\substack{x=x_e \\ u=u_e}} \quad D = \left.\frac{\partial g}{\partial u}\right|_{\substack{x=x_e \\ u=u_e}}$$

$\in \mathbb{R}^{n \times n}$ $\in \mathbb{R}^{n \times n}$ $\in \mathbb{R}^{n \times m}$ $\in \mathbb{R}^{p \times n}$ $\in \mathbb{R}^{p \times m}$

Numerical Linearization

- Can convert into a more common linear state space form:

$$\begin{aligned} M\dot{x} &= Fx + Gu \\ y &= Cx + Du \end{aligned} \quad \Rightarrow \quad \begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

- The state equation matrices are related by:

$$A = M^{-1}F, B = M^{-1}G$$

- Linearize.m applies a simple centered difference scheme with user specified perturbations:

$$F_{ij} = \frac{f_i\left(x_e + \Delta x_j, \dot{x}_e, u_e\right) - f_i\left(x_e - \Delta x_j, \dot{x}_e, u_e\right)}{2\delta x_j} \qquad G_{ij} = \frac{f_i\left(x_e, \dot{x}_e, u_e + \Delta u_j\right) - f_i\left(x_e, \dot{x}_e, u_e - \Delta u_j\right)}{2\delta u_j}$$

$$\Delta x_j = \begin{bmatrix} 0 & \cdots & 0 & \delta x_j & 0 & \cdots & 0 & 0 \end{bmatrix}^T \qquad \Delta u_j = \begin{bmatrix} 0 & \cdots & 0 & \delta u_j & 0 & \cdots & 0 & 0 \end{bmatrix}^T$$

C and D matrix computations are similar, but M matrix is as follows:

$$\hat{M}_{ij} = \frac{f_i\left(x_e, \dot{x}_e + \Delta\dot{x}_j, u_e\right) - f_i\left(x_e, \dot{x}_e - \Delta\dot{x}_j, u_e\right)}{2\delta\Delta\dot{x}_j} \qquad M = I - \hat{M}$$

$$\Delta\dot{x}_j = \begin{bmatrix} 0 & \cdots & 0 & \delta\dot{x}_j & 0 & \cdots & 0 & 0 \end{bmatrix}^T$$

Numerical Linearization

- The constants structure defines perturbation sizes:

  DELXLIN defines perturbations for states and state derivatives
  DELULIN defines perturbations for controls

- Need to be conscious of units when defining perturbation sizes. E.g. do not use perturbation of $O(1)$ when the unit is radians. 1 rad is a too large perturbation
- Resulting linear model matrices are then usable in LTI objects with the various tools in MATLAB control systems toolbox.

Numerical Linearization

- Linearize.m does not bother constructing a C matrix with the states as output.
- Please remember, if you are just interested in state outputs, you can always easily construct such a linear system by setting C to identity (or some submatrix of identity) and D as a zeros matrix.

$\hat{y} = \vec{C}x + \vec{D}u$

$\vec{y} = \begin{bmatrix} x \\ y \end{bmatrix}$

$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

这句 看不懂的