Flat Earth Equations of Motion
Useful for Studying Flight Dynamics and Control

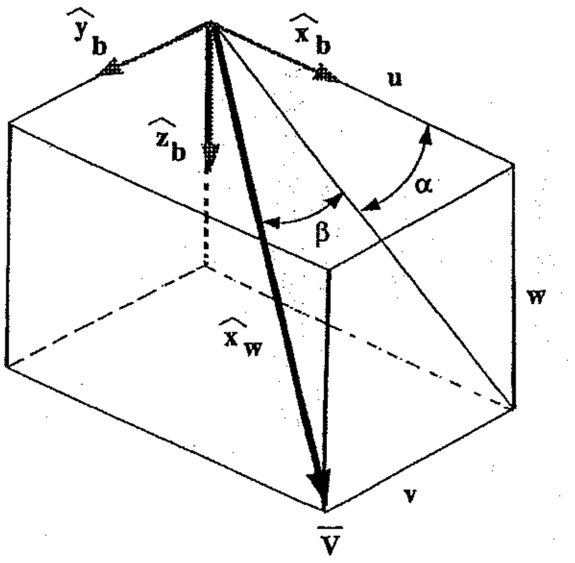
Professor Dominick Andrisani
School of Aeronautics and Astronautics
Purdue University

Reference: Brian L. Stevens and Frank L. Lewis, *Aircraft Control and Simulation,* John Wiley and Sons, 1992.

A. Nonlinear Aerodynamic Model

A.1. Body Axis System has unit vectors $(\hat{i}, \hat{j}, \hat{k})$ (not stability axis system). Let $\hat{i}_w, \hat{j}_w, \hat{k}_w$ be wind axis system unit vectors with $\hat{i}_w$ directed into the wind, $\hat{k}_w$ in the plane of symmetry but perpendicular to $\hat{i}_w$, $\hat{j}_w$ perpendicular to $\hat{i}_w, \hat{k}_w$ in a right hand rule sense (generally out the right wing)

A.2. Definitions of Angle of Attack and Sideslip



$$\overline{v} = |\overline{v}|\hat{x}_w = u\hat{x}_b + v\hat{y}_b + w\hat{z}_b$$

$$\sin\beta = \frac{v}{\sqrt{u^2 + v^2 + w^2}}$$

$$\tan\alpha = \frac{w}{u}$$

Assume still air.

Roskam notation is $\overline{V} = U\hat{i} + V\hat{j} + W\hat{k}$

A.3. Force Analysis

$$\overline{F}_A = -\underbrace{\text{Drag}}_{D}\,\hat{i}_w - \underbrace{\text{Lift}}_{L}\,\hat{k}_w + \underbrace{\text{Side Force}}_{Y}\,\hat{j}_w$$

$$\frac{\overline{F}_A}{\overline{q}S} = -C_D \hat{i}_w - C_L \hat{k}_w + C_{y_w} \hat{j}_w$$

We will use the transform between wind axis system and body axis system. This involves wind angles α and β (angle of attack and sideslip angle).

See Stevens and Lewis page 63, equations 2.3-2

$$\begin{aligned}\overline{Q} &= Q_{x_B}\hat{i} + Q_{y_B}\hat{j} + Q_{z_B}\hat{k} \\ &= Q_{x_w}\hat{i}_w + Q_{y_w}\hat{j}_w + Q_{z_w}\hat{k}_w\end{aligned}$$

$$\downarrow \; T^{B\ to\ w}$$

$$\begin{bmatrix} Q_{x_w} \\ Q_{y_w} \\ Q_{z_w} \end{bmatrix} = \left[\begin{array}{c|c|c} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{array}\right] \begin{bmatrix} Q_{x_B} \\ Q_{y_B} \\ Q_{z_B} \end{bmatrix}$$

$$\begin{bmatrix} Q_{x_B} \\ Q_{y_B} \\ Q_{z_B} \end{bmatrix} = \left[\begin{array}{c|c|c} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{array}\right] \begin{bmatrix} Q_{x_w} \\ Q_{y_w} \\ Q_{z_w} \end{bmatrix}$$

$$\uparrow \; T^{W\ to\ B}$$

where

$$\begin{aligned}&\tan\alpha = w/u \\ &\sin\beta = v/|\overline{v}| \\ &\left|\overline{v} = (u^2 + v^2 + w^{2)^{1/2}}\right|\end{aligned}$$

A.4. Examples of Using the Transformation Between Wind and Body Axis System

$$\overline{v} = u\hat{i} + v\hat{j} + w\hat{k} = |\overline{v}|\hat{i}_w$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} & & \\ & T^{W\ to\ B} & \\ & & \end{bmatrix} \begin{bmatrix} |\overline{v}| \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} |\overline{v}|\cos\alpha\cos\beta \\ |\overline{v}|\sin\beta \\ |\overline{v}|\sin\alpha\cos\beta \end{bmatrix}$$

$$\frac{\overline{F}_A}{\overline{q}S} = -C_D\hat{i}_w + C_y\hat{j}_w - C_L\hat{k}_w$$

$$= C_x\hat{i} + C_y\hat{j} + C_z\hat{k}$$

$$\begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = T^{\text{w to B}} \begin{bmatrix} -C_D \\ C_{y_w} \\ -C_L \end{bmatrix}$$

A.5. Moment Analyses

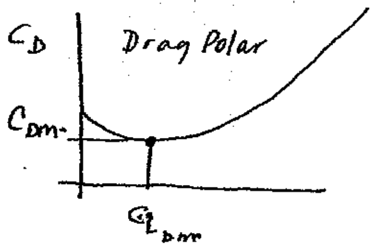
$$M_A = M_{A_x}\hat{i} + M_{A_y}\hat{j} + M_{A_z}\hat{k}$$

$$= \overline{q}S\left[bC_\ell\hat{i} + \overline{c}\,C_M\hat{j} + bC_N\hat{k} \right]$$

b is wing span $\qquad$ $\overline{c}$ mean aerodynamic chord

A.6. Aerodynamic Force Coefficient Model (Wind Axis System)

Drag Polar



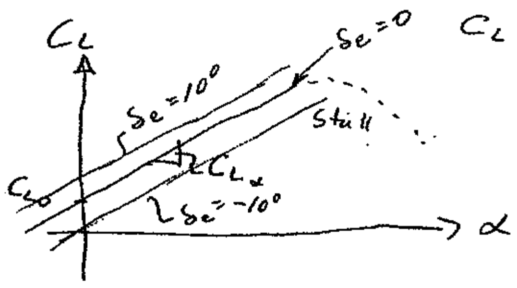
$C_{L_{DM}}$ is lift for minimum drag coefficient

C_{DM} is minimum drag coefficient

$$C_D = k\left(C_{L_{static}} + C_{L_{DM}}\right)^2 + C_{DM}$$ <<Drag Polar

Often k, $C_{L_{DM}}$ and C_{DM} are functions of Mach number.

Static Lift Coefficient



$$C_{L_{static}} = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta e}}\delta_e$$

We will use only the linear form for lift. Although to model still we need to use a different form

$$C_L = C_{L_{static}} + \underbrace{C_{L\dot\alpha}\frac{\dot\alpha\bar c}{2|\bar v|} + C_{L_Q}\frac{Q\bar c}{2|\bar v|}}_{C_{L_{dynamic}}}$$

$$C_{y_w} = \underset{\substack{\downarrow \\ C_{y_{static}}}}{C_{y_0} + C_{y_\beta}\beta + C_{y_{\delta r}}\delta r + C_{y_{\delta a}}\delta a} + \underbrace{C_{y_P}\frac{Pb}{2|\bar v|} + C_{y_R}\frac{Rb}{2|\bar v|}}_{C_{Y_{dynamic}}}$$

A.7. Aerodynamic Moment Model about cg (body axis system)

<u>Rolling Moment</u>

$$C_\ell = \underbrace{C_{\ell_0} + C_{\ell_\beta}\beta + C_{\ell_{\delta a}}\delta a + C_{\ell_{\delta r}}\delta r}_{C_{\ell_{static}}} + \underbrace{\frac{b}{2|\bar v|}\left[C_{\ell_P}P + C_{\ell_R}R\right]}_{C_{\ell_{dynamic}}}$$

<u>Pitching Moment</u> about cg

about cg $C_M = C_M^{ref} + C_Z\left(\frac{x_{ref}}{\bar c} - \frac{x_{cg}}{\bar c}\right)$

about arbitrary reference point.

$$C_M^{ref} = \overbrace{C_{M_0} + C_{M_\alpha}\alpha + C_{M_{\delta e}}\delta e}^{C_{M_{static}}} + \underbrace{\frac{\dot\alpha\bar c}{2|\bar v|}C_{M_{\dot\alpha}} + \frac{Q\bar c}{2|\bar v|}C_{M_Q}}_{C_{M_{dynamic}}}$$

often C_{M_0} C_{M_α} $C_{M_{\delta e}}$ are functions of Mach number. We will ignore this.

Yaving moment

about arbitrary reference point

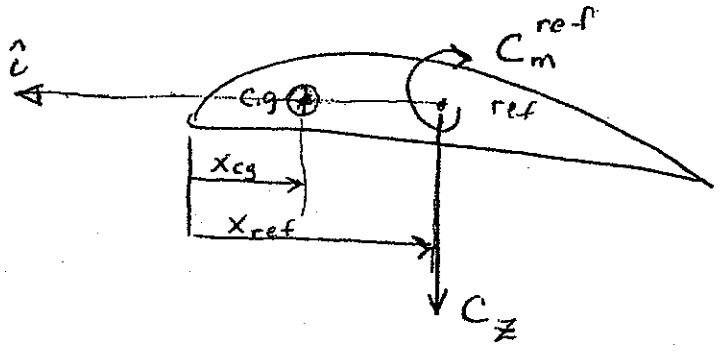
$$C_N^{ref} = \underbrace{C_{N_0} + C_{N_\beta}\beta + C_{N_{\delta a}}\delta a + C_{N_{\delta r}}\delta r}_{C_{N_{static}}} + \underbrace{\frac{b}{2|\bar{v}|}\left(C_{N_P}P + C_{N_R}R\right)}_{C_{N_{dynamic}}}$$

about cg

$$C_N = C_N^{ref} - \frac{\bar{c}}{b}C_Y\left(\frac{x_{ref}}{\bar{c}} - \frac{x_{cg}}{\bar{c}}\right)$$

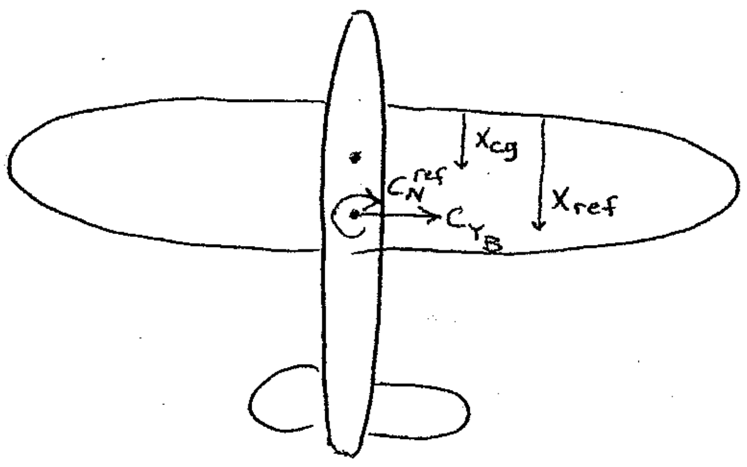
A.8. C. G. Different from Moment Reference Point

Assume there is a moment reference point located behind the cg



Ignore pitching moments due to drag and y-force

about cg $$C_M = C_M^{ref} + C_Z\left(\frac{x_{ref}}{z} - \frac{x_{cg}}{\bar{c}}\right)$$



Ignore yawing moments due to lift and drag forces

Body axis component of aero force in Y direction

$$C_N = C_N^{ref} - C_Y^{\downarrow}\left(\frac{x_{ref}}{b} - \frac{x_{cg}}{b}\right)$$

$$= C_N^{ref} - \frac{\bar{c}}{b} C_Y \left(\frac{x_{ref}}{\bar{c}} - \frac{x_{cg}}{\bar{c}}\right)$$

Summary of Unknown Aerodynamic Parameters

Expression	Symbol	Name	Importance
Drag	k		Big
	$C_{L_{DM}}$	Lift at minimum drag	Big
	C_{DM}	Minimum drag coefficient.	Big
Lift	C_{L_0}	Lift coefficient at zero α	big
	C_{L_α}	Lift curve slope	big
	$C_{L_{\delta e}}$	Lift due to elevator	big
	$C_{L_{\dot{\alpha}}}$	Lift due to $\dot{\alpha}$	medium size
	C_{L_Q}	Lift due to pitch rate	medium size
Side Force	C_{y_0}	Side force when $\beta = \delta r = \delta a = P = R = 0$	Usually zero
	C_{y_β}	Side force due to slideslip	big
	$C_{y_{\delta r}}$	Side force due to rudder	big
	$C_{y_{\delta a}}$	Side force due to aileron ,	small
	C_{y_P}	Side force due to roll rate	small
	C_{y_R}	Side force due to yaw rate	medium
Cg location	$x_{cg}/\bar{c}$		
	$x_{ref}/\bar{c}$		

Roll Mom.	C_{ℓ_0}	Roll mom when $\beta=\delta r=\delta a=P=R=0$	Usually zero
	$C_{\ell\beta}$	Rolling mom. due to slideslip, <u>(dihedral effect)</u>	Big
	$C_{\ell_{\delta a}}$	Rolling mom. due to aileron, <u>aileron effectiveness</u>	Big
	$C_{\ell_{\delta R}}$	Rolling mom. due to rudder	Medium
	C_{ℓ_P}	Rolling mom. due to roll rate, <u>damping in roll</u>	Big
	C_{ℓ_R}	Rolling mom. due to yaw rate	Small
Pitch Mom.	C_{M_0}	Pitch mom. when $\alpha=\delta e=\dot{\alpha}=Q=0$,	big
	C_{M_α}	Pitch mom. due to α	big
	$C_{M_{\delta e}}$	Pitch mom. due to δe, <u>elevator effectiveness</u>	big
	$C_{M_{\dot{\alpha}}}$	Pitch mom. due to $\dot{\alpha}$, <u>lag of downwash deriv.</u>	big
	C_{M_Q}	Pitch mom. due to Q, <u>damping in pitch</u>	big
Yawing Mom.	C_{N_0}	Yaw mom. when $\beta=\delta a=\delta r=P=R=0$	usually zero
	C_{N_β}	Yaw mom. due to sideslip, <u>weathercock stab.</u>	big
	$C_{N_{\delta a}}$	Yaw mom. due to aileron,	medium
	$C_{N_{\delta r}}$	Yaw mom. due to rudder, <u>rudder effectiveness</u>	big
	C_{N_P}	Yaw mom. due to roll rate	small
	C_{N_R}	Yaw mom. due to yaw rate, damping in yaw	big

B. A Minor Problem with $\dot{\alpha}$

B.1. Definition

since

$$\tan\alpha=\frac{W}{U}$$

$$\dot{\alpha}=\frac{U\dot{W}-W\dot{U}}{U^2+W^2}$$

An approximation to this equation is sometimes used. Since $U \gg W$ and $U \gg V$

$$U^2+W^2 \approx U^2$$

i.e.

V and W typically small compared to U

also $U >> W$ so

$$U\dot{W} >> W\dot{U}$$

Therefore

$$\dot{\alpha} \approx \frac{\dot{W}}{U}$$

B.2. The Problem

Notice that C_L and C_M a function of $\dot{\alpha}$ or equivalently $\dot{W}$. This makes F_{A_x} F_{A_y} and F_{A_z} functions of $\dot{W}$, i.e. $F_{A_z}(\dot{W}, \ldots)$. The equations of motion for body axis force have $\dot{W}$ on the left hand side and on the right hand side, i.e.

$$\dot{W} = QU - PV + g_0' \cos\phi \cos + \frac{F_{Az}}{m}(\dot{W}, \ldots)$$

We would like to solve for the $\dot{W}$ term on the right hand side, move it to the left hand side, combine terms on the LHS and divide through by the multiplier of $\dot{W}$.

Suppose

$$F_{A_z}(\dot{W}, \alpha, \delta_e, Q) = F_1(\dot{W}) + F_{A_z}'(\alpha, \delta e, Q)$$

where

$$F_1(\dot{W}) = a\dot{W}$$

$$\uparrow \text{ some multiplier of } \dot{W}$$

Then

$$\dot{W}\left(1 - \frac{a}{m}\right) = QU - PV + g_0' \cos\phi \cos\theta + \frac{F_z'}{m}(\alpha, \delta_e, Q)$$

and

$$\dot{W} = \frac{1}{1 - \frac{a}{m}}\left[QU - PV + g_0' \cos\phi \cos\theta + \frac{F_z'}{m}(\alpha, \delta_e, Q)\right]$$

Now we have a diff. eqn. where $\dot{W}$ only appears on the LHS.

When we go to solve the moment equation we have the following problem

$$\dot{Q} = C_s PR - C_6(P^2 - R^2) + C_7 M(\dot{W}, \alpha, \delta e, Q)$$

But this problem is simpler because we have an explicit equation for $\dot{W}$ above that we can use on the RHS of the $\dot{Q}$ equation. Therefore we can determine $\dot{Q}$.

It turns out that $a = -\bar{q}S\frac{\cos\alpha}{m}C_{L_{\dot{\alpha}}}\frac{\bar{c}}{2|\bar{v}|}\frac{1}{U}$

Recall

$$\dot{W} = QU - PV + g_0' \cos\phi \cos\theta + \frac{F_Z}{m}$$

but a portion of F_Z is a function of $\dot{\alpha}$ and therefore $\dot{W}$. Let's examine this term separately. Use the approximation that $\dot{\alpha} = \frac{\dot{W}}{U}$

$$\begin{bmatrix} F_{A_x} \\ F_{A_y} \\ F_{A_z} \end{bmatrix} = \bar{q}S \begin{bmatrix} T^{W\ to\ B} \end{bmatrix} \begin{bmatrix} -C_D \\ +C_y \\ -C_L \end{bmatrix}$$

but $C_L = f_1(\dot{\alpha}) + f_2(\alpha, \delta_e, Q)$

$$f_1(\dot{\alpha}) = C_{L_{\dot{\alpha}}} \frac{\dot{\alpha}\bar{c}}{2|\bar{v}|}$$

$$f_1(\dot{w}) = C_{L_{\dot{\alpha}}} \frac{\bar{c}}{2|\bar{v}|} \frac{\dot{w}}{U}$$

$$f_2 = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta e}}\delta e + C_{L_Q} \frac{Q\bar{c}}{2|\bar{v}|}$$

$$\begin{bmatrix} F_{A_x} \\ F_{A_y} \\ F_{A_z} \end{bmatrix} = \bar{q}S\ T^{W\ to\ B} \begin{bmatrix} o \\ 0 \\ -f_1(\dot{w}) \end{bmatrix} + \bar{q}S\ T^{W\ to\ B} \begin{bmatrix} -C_D \\ +C_y \\ -f_2 \end{bmatrix}$$

$$= \bar{q}S \begin{bmatrix} +\sin\alpha\ \ f_1(\dot{w}) \\ 0 \\ -\cos\alpha\ \ f_1(\dot{w}) \end{bmatrix} + \bar{q}S\ T^{W\ to\ B} \begin{bmatrix} -C_D \\ +C_y \\ -f_2 \end{bmatrix}$$

↑ ignore this term since it is small.

$$\begin{bmatrix} F_{A_x} \\ F_{A_y} \\ F_{A_z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\bar{q}S\cos\alpha C_{L_{\dot{\alpha}}} \frac{\bar{c}}{z|\bar{v}|} \frac{\dot{w}}{U} \end{bmatrix} + \underbrace{\bar{q}S\ T^{W\ to\ B} \begin{bmatrix} -C_D \\ +C_y \\ -f_2 \end{bmatrix}}_{\begin{bmatrix} F_{A_x} \\ F_{A_y} \\ F_{A_z'} \end{bmatrix}}$$

$$\therefore \quad \dot{w} = QU - PV + g_0' \cos\phi\cos\theta + \frac{F_{A_z'}}{m} - \bar{q}\frac{S\cos(\alpha)}{m} C_{L_{\dot{\alpha}}} \frac{\bar{c}}{2|\bar{v}|}\frac{\dot{w}}{U}$$

$$\dot{w}\left(1 - \frac{a}{m}\right) = QU - PV + g_0' \cos\phi\cos\theta + \frac{F_{A_z'}}{m}$$

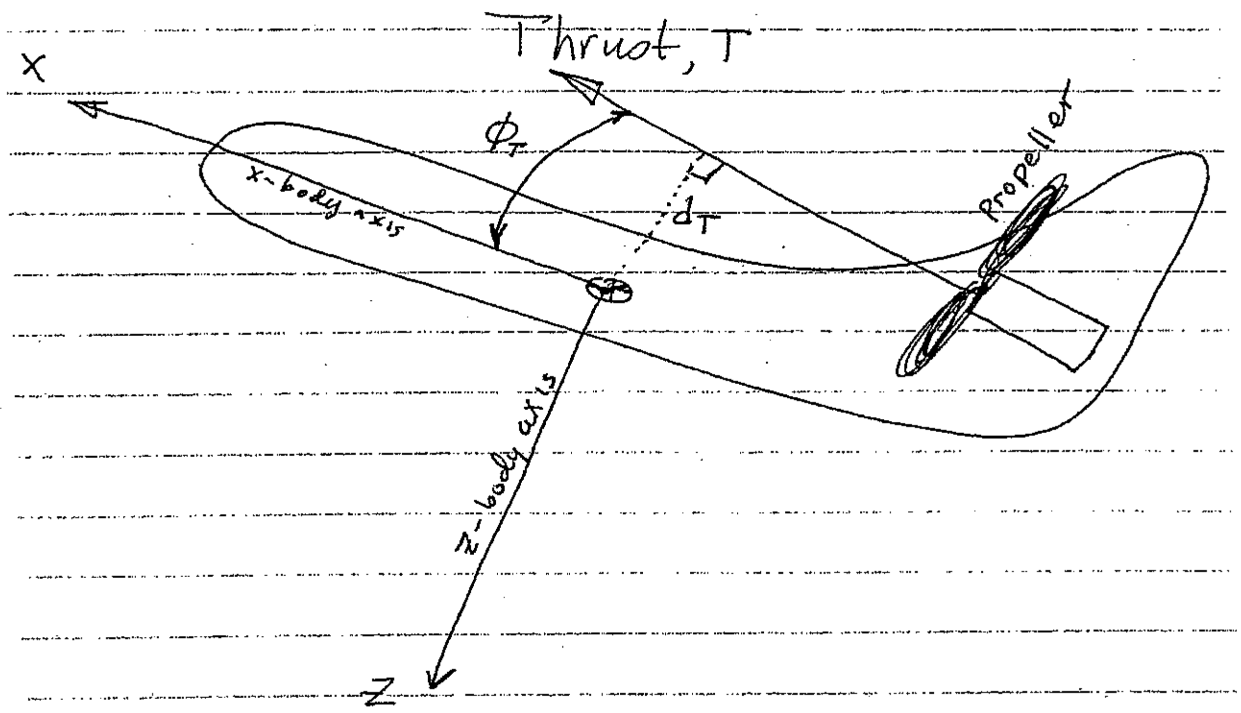
$$\dot{w} = \left(\frac{1}{1 - \frac{a}{m}}\right) = \left[QU - PV + g_0' \cos\phi\cos\theta + \frac{F_{A_z'}}{m}\right]$$

and then $\dot{\alpha} \cong \frac{\dot{w}}{U}$

$$\dot{\alpha} = \frac{1}{U}\left(\frac{1}{1 - \frac{a}{m}}\right)\left[QU - PV + g_0' \cos\phi\cos\theta + \frac{F_{A_z'}}{m}\right]$$

and we can substitute this into the pitching moment equation at the $C_{m_{\dot{\alpha}}}$ term.

C. Propeller Thrust Model



d_T is the thrust offset distance

ϕ_T is the thrust offset angle

T is the magnitude of the thrust

$T = 550\ \text{Bhp}\ \ N_p / V_t$ (lbf)

Bhp is the brake horsepower of the engine (hp)

N_p is the propeller efficiency (non-dimensional)

V_t is the aircraft speed (ft/sec)

Thrust is assumed to act in the x-z plane.

TABLE 2.5-1: The Flat-Earth, Body-Axes 6-DOF Equations

FORCE EQUATIONS

$$\dot{U} = RV - QW - g_D \sin\theta + (X_A + X_T)/\mathrm{m}$$
$$\dot{V} = -RU + PW + g_D \sin\phi \,\cos\theta + (Y_A + Y_T)/\mathrm{m}$$
$$\dot{W} = QU - PV + g_D \cos\phi \,\cos\theta + (Z_A + Z_T)/\mathrm{m}$$

KINEMATIC EQUATIONS

$$\dot{\phi} = P + \tan\theta\,(Q \,\sin\phi + R \,\cos\phi)$$
$$\dot{\theta} = Q \,\cos\phi - R \,\sin\phi$$
$$\dot{\psi} = (Q \,\sin\phi + R \,\cos\phi)/\cos\theta$$

MOMENT EQUATIONS

$$\Gamma\dot{P} = J_{xz}\left[J_x - J_y + J_z\right]PQ - \left[J_z(J_z - J_y) + J_{xz}^2\right]QR + J_z\ell + J_{xz}n$$
$$J_y\dot{Q} = (J_z - J_x)PR - J_{xz}(P^2 - R^2) + \mathrm{m}$$
$$\Gamma\dot{R} = \left[(J_x - J_y)J_x + J_{xz}^2\right]PQ - J_{xz}\left[J_x - J_y + J_z\right]QR + J_{xz}\ell + J_x n$$
$$\Gamma = J_x J_z - J_{xz}^2$$

NAVIGATION EQUATIONS

$$\dot{p}_N = U\mathrm{c}\theta\mathrm{c}\psi + V(-\mathrm{c}\phi\mathrm{s}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi) + W(\mathrm{s}\phi\mathrm{s}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{c}\psi)$$
$$\dot{p}_E = U\mathrm{c}\theta\mathrm{s}\psi + V(\mathrm{c}\phi\mathrm{c}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi) + W(-\mathrm{s}\phi\mathrm{c}\psi + \mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi)$$
$$\dot{h} = U\mathrm{s}\theta - V\mathrm{s}\phi\mathrm{c}\theta - W\mathrm{c}\phi\mathrm{c}\theta$$

The Array 67 Element Called "constant"

**Definitions of the array called "constant"
containing all the constants necessary to describe
the aerodynamic, thrust, and inertia properties of an air vehicle
for use in flight simulation software**

This includes
aircraft geometry,
mass properties,
inertia properties,
vehicle reference points,
stability and control derivatives,
propeller properties, and
some trim conditions.

Definition of the Vehicle Specific Constants Called “constant” in flight Simulation Software

```
% Mass related inputs
constant(1)=W;              % W, Weight, pounds (lbf)
constant(2)=32.17405;       % g, Acceleration of gravity, ft/(sec*sec)
constant(3)=constant(1)/constant(2);        % mass, slugs
constant(4)=Ixx;            % Ixx, slug*ft*ft
constant(5)=Iyy;            % Iyy, slug*ft*ft
constant(6)=Izz;            % Izz, slug*ft*ft
constant(7)=Ixz;            % Ixz, slug*ft*ft
constant(8)=eta_p;          % propeller efficiency, eta, nondimensional
constant(9)=0;              % unassigned

% Derived constants from the inertia data
constant(10)=constant(4)*constant(6)-constant(7)*constant(7);                              %gamma
constant(11)=((constant(5)-constant(6))*constant(6)-constant(7)*constant(7))/constant(10);  % c1
constant(12)=(constant(4)-constant(5)+constant(6))*constant(7)/constant(10);               % c2
constant(13)= constant(6)/constant(10);                                                    % c3
constant(14)= constant(7)/constant(10);                                                    % c4
constant(15)=(constant(6)-constant(4))/constant(5);                                        % c5
constant(16)= constant(7)/constant(5);                                                     % c6
constant(17)= 1/constant(5);% c7
constant(18)=  (constant(4)*(constant(4)-constant(5))+constant(7)*constant(7))/constant(10);% c8
constant(19)=  constant(4)/constant(10);% c9

% aircraft geometry
constant(20)=S_w;    % S, wing area, ft^2
constant(21)=c_w;    % cbar, mean geometric chord, ft
constant(22)=b_w;    % b, wing span, ft
constant(23)=0;      % phiT, thrust inclination angle, RADIANS
constant(24)=0;      % dT, thrust offset distance, ft
```

```
% Nondimensional Aerodynamic stability and control derivatives

% Drag Polar CD=k(CLstatic-CLdm)^2 + CDm
constant(25)=Cd_0;          % CDm, CD for minimum drag
constant(26)=k;             % k
constant(27)=0;             % CLdm, CL at the minimum drag point

% Lift Force
constant(28)=CL_0(S_w,S_h,M,tc_w,alpha_0,epsilon_t,i_w,i_h,epsilon_0_h,AR_w,Lambda_c4,Lambda_c2,lambda_w,kappa,beta,b_w,d,AR_h,eta_h);     % CL0
constant(29)=CL_alpha(AR_w,AR_h,Lambda_c2,lambda_w,l_h,h_h,b_w,d,eta_h,S_h,S_w,kappa_h,Lambda_c2_h,beta,kappa);     % CLalpha
constant(30)=CL_de(S_w,S_h,AR_h,ce_ch,eta_oe,eta_ie,beta,kappa_h,lambda_h,Lambda_c2_h,tc_h,delta_e,Cl_alpha_h);     % CLdeltaE
constant(31)=CL_alpha_dot(l_h, h_h, b_w, lambda, AR_w, AR_h, Lambda_c4, Lambda_c4_h, beta, kappa, kappa_h, V_h, eta_h);     % CLalphadot
constant(32)=CL_q(Xw,b_w,c_w,c_h,AR_w,Lambda_c4,Lambda_c2,Lambda_c2_h,Xh,S_h,S_w,eta_h,AR_h,beta, V_h,b_h, kappa, kappa_h);     % CLQ

% Side Force
constant(33)=0;     % CY0
constant(34)=Cy_beta(Cl_alpha_w,two_r_one,eta_v,beta,AR_v,b_v,Z_h,x_over_c_v,lambda_v,S_v,S_w,Lambda_c4,Lambda_c4_v,Z_w,d,dihedral,wingloc,Z_w1,S_h,S_o);     % CYbeta
constant(35)=Cy_da(S_w);     % CYdeltaA
constant(36)=Cy_dr(S_w,b_w,S_h,S_v,b_v,c_v,x_AC_vh,two_r_one,AR_v,l_v,Z_v,eta_or,eta_ir,cr_cv,beta,kappa_v,Lambda_c2_v,lambda_v,delta_r,alpha); % CydeltaR
constant(37)=Cy_p(b_w,l_v,Z_v,alpha,two_r_one,eta_v,beta,AR_v,b_v,Z_h,x_over_c_v,lambda_v,S_v,S_w,Lambda_c4_v,Z_w,d,dihedral,wingloc,Z_w1,S_h,S_o);     % Cy_p
constant(38)=Cy_r(alpha,two_r_one,eta_v,beta,AR_v,b_v,b_w,Z_h,x_over_c_v,lambda_v,S_v,S_w,Lambda_c4,Z_w,d,dihedral,wingloc,Z_w1,S_h,S_o,l_v,Z_v);     % CYR

% Rolling Moment
constant(39)=0;     % Cl0
constant(40)=Cl_beta(CL_wb,Cl_alpha_w,theta,theta_h,Lambda_c2, Lambda_c4,Lambda_c4_h,S_w,b_w,AR_w,AR_h,Z_v,alpha,l_v,...
M,two_r_one,eta_v,beta,AR_v,b_v,Z_h,x_over_c_v,lambda_v,S_v,Z_w,d,wingloc,Z_w1,S_h,S_o,dihedral,dihedral_h,Lambda_c2_h,l_b,b_h,CL_hb,lambda_w,lambda_h);     % Clbeta
constant(41)=Cl_da(S_w,AR_w,ca_cw,eta_ia,eta_oa,beta,kappa,Lambda_c4,lambda_w,tc_w,Cl_alpha_w);     % CldeltaA
constant(42)=Cl_dr(S_w,b_w,S_h,S_v,b_v,c_v,x_AC_vh,two_r_one,AR_v,l_v,Z_v,eta_or,eta_ir,cr_cv,beta,kappa_v,Lambda_c2_v,lambda_v,delta_r,alpha); % CldeltaR
constant(43)=Cl_p(b_h,b_w,Z_v,two_r_one,eta_v,AR_v,b_v,Z_h,x_over_c_v,lambda_v,S_v,S_w,Lambda_c4,Lambda_c4_h,Z_w,d,lambda_w,wingloc,Z_w1,S_h,S_o,beta,Cl_al
pha,Cl_alpha_h,AR_w,lambda_h, AR_h);     % ClP
constant(44)=Cl_r(l_v,alpha,Z_v,S_h,S_v,x_over_c_v,b_v,Z_h,two_r_one,lambda_v,S_w,Z_w,d,AR_w,beta,Lambda_c4,c_w,Xw,AR_v,eta_v,b_w,cf,delta_f,theta,lambda_w,Ga
mma,kappa,B);     % ClR

% Pitching Moment
constant(45)=Cm_0(S_h_slip,S_w,S_h,M,tc_w,alpha_0,epsilon_t,i_w,i_h,epsilon_0_h,AR_w,Lambda_c4,lambda_w,beta,Cm_0_r,Cm_o_t,Lambda_c2_h,kappa_h,AR_h,Xh,c_w,eta_h);  % CM0
constant(46)=Cm_alpha(AR_w,AR_h,Lambda_c2,lambda_w,l_h,h_h,b_w,c_w,d,eta_h,S_h,S_w,kappa_h,Lambda_c2_h,beta,kappa,Xref,Xacwb);     % Cmalpha
constant(47)=Cm_de(S_w,S_h,AR_h,ce_ch,eta_oe,eta_ie,beta,kappa_h,lambda_h,Lambda_c2_h,tc_h,delta_e,Cl_alpha_h,V_h);     % CMdeltaE
constant(48)=Cm_a_dot(l_h,h_h,b_w,lambda_w,AR_w,Lambda_c4,M,Cl_alpha,eta_h,S_h,Xh,c_w,S_w,kappa,kappa_h,Lambda_c2,AR_h);     % CMalphadot
constant(49)=Cm_q(Xw,b_w,c_w,c_h,AR_w,Lambda_c4,Lambda_c2,Xh,S_h,S_w,eta_h,AR_h,beta,V_h,b_h,Cl_alpha,B);     % CMQ
```

```
% Yawing Moment
constant(50)=0;                                                                                                  % CN0
constant(51)=Cn_beta(Cl_alpha_w,S_w,b_w,alpha,l_v,Z_v,l_f,S_b_s,Rl_f,x_m,h1_fuse,h2_fuse,hmax_fuse,wmax_fuse,two_r_one,eta_v,M,AR_v,b_v,Z_h,x_over_c_v,lambda_v
,S_v,Z_w,d,Z_w1,S_h,Lambda_c4,Lambda_c2_v);                                                                      % CNbeta
constant(52)=Cn_da(S_w,AR_w,ca_cw,eta_ia,eta_oa,beta,kappa,Lambda_c4,lambda_w,tc_w,Cl_alpha_w,CL);               % CNdeltaA
constant(53)=Cn_dr(S_w,b_w,S_h,S_v,b_v,c_v,x_AC_vh,two_r_one,AR_v,l_v,Z_v,eta_or,eta_ir,cr_cv,beta,kappa_v,Lambda_c2_v,lambda_v,delta_r,alpha);  % CNdeltaR
constant(54)=Cn_p(c_w,B,theta,adelf,delf,b_w,l_v,b_h,Z_v,two_r_one,eta_v,AR_v,b_v,Z_h,x_over_c_v,lambda_v,S_v,S_w,Lambda_c4,Lambda_c2,Lambda_c4_h,Z_w,d,lambd
a_w,wingloc,Z_w1,S_h,S_o,beta,Cl_alpha,Cl_alpha_h,AR_w,lambda_h, AR_h, b_f,Xw,CL,Lambda_c4_v,alpha); % CNP
constant(55)=Cn_r(CL,C_bar_D_o,l_v,alpha,Z_v,S_h,S_v,x_over_c_v,b_v,Z_h,two_r_one,lambda_v,S_w,Z_w,d,AR_w,beta,Lambda_c4,c_w,Xw,AR_v,eta_v,b_w); % CNR

% Reference positions
constant(56)=Xref/c_w;        % XbarRef, nondimensional, measured aft from leading edge of wing mean aerodynamic chord.
constant(57)=.25;             % XbarCG,  nondimensional, measured aft from leading edge of wing mean aerodynamic chord.
% constant(57 can be changed at a later time if you use program FlatEarth or E_Earth.

% Trim conditions. These may or not be used by subsequent programs LongSC.m and LatSC.m. Small
% variations in these trim flight conditions are OK.
constant(58)=U1;              % Trim speed, Vt, ft/sec
constant(59)=altitude;        % Trim altitude, ft
constant(60)=0;               % Trim alpha, >>>DEGREES<<<This is not used by LongSC

% The constants below are used only by LongSC.m and LatSC.m MATLAB scripts for linear small perturbation
%  longitudinal and lateral-directional analysis as defined in Reference 1.
constant(61)=0;         % CLu
constant(62)=0;         % CDu
constant(63)=0;         % CTxu
constant(64)=0;         % Cmu
constant(65)=0;         % CmTu
constant(66)=0;         % CmTalpha
constant(67)=0;         % CDdeltae
```

Software for Simulating Six Degree of Freedom Motion of a Rigid Aircraft

Professor Dominick Andrisani, II
andrisan@purdue.edu

School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN

January 18, 2006

Introduction

The software described here allows for six degree of freedom simulation of the arbitrary motion of a rigid aircraft. The nonlinear differential equations are of the form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{g}(\mathbf{x},\mathbf{u}), \ \ \mathbf{x}(0) \\ \bar{\mathbf{y}} &= \bar{\mathbf{h}}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \\ \mathbf{u}(t) &= \textbf{specified} \end{aligned}$$

where x, y and u are the state, output and control vectors respectively and x(0) is the initial condition.

Files created for Simulink are Matlab version dependent. Simulink files provided below work in Matlab 6 and 7.

Two different ways of getting the solution to these nonlinear differential equations are provided. The first method is by numerical integration. SIMULINK is used for this. The second method uses linearization to obtain the total approximate solution. These methods are depicted in Figure 1. On the figure are the names of MATLAB scripts (ending with .m) and SIMULINK model files (ending with .mdl) that implement the various functions in each block.

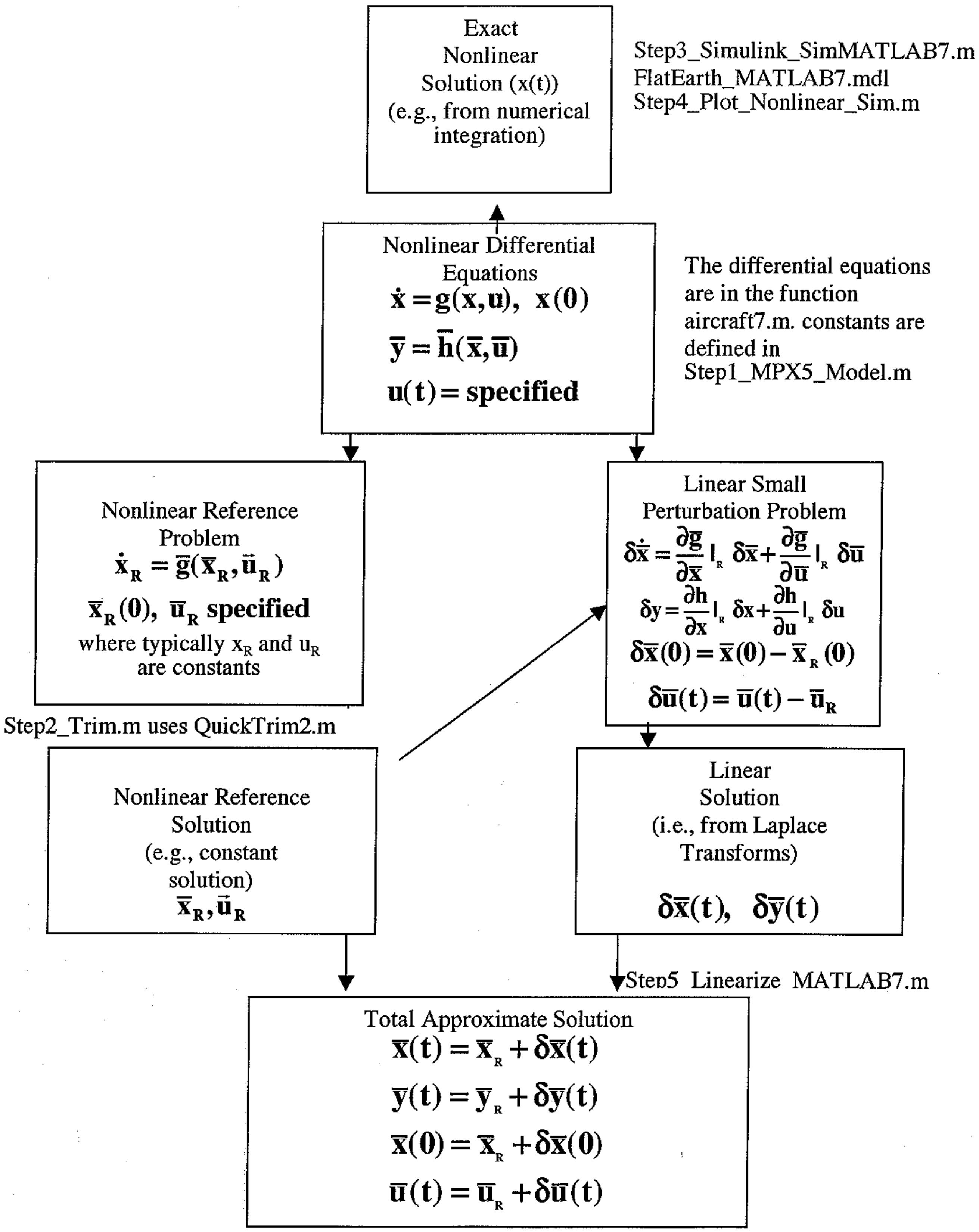

Exact Nonlinear Solution (x(t)) (e.g., from numerical integration)
Step3_Simulink_SimMATLAB7.m
FlatEarth_MATLAB7.mdl
Step4_Plot_Nonlinear_Sim.m
Nonlinear Differential Equations
$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x},\mathbf{u}),\ \mathbf{x}(0)$
$\bar{\mathbf{y}} = \bar{\mathbf{h}}(\bar{\mathbf{x}},\bar{\mathbf{u}})$
$\mathbf{u}(\mathbf{t}) = \textbf{specified}$
The differential equations are in the function aircraft7.m. constants are defined in Step1_MPX5_Model.m
Nonlinear Reference Problem
$\dot{\mathbf{x}}_R = \bar{\mathbf{g}}(\bar{\mathbf{x}}_R,\vec{\mathbf{u}}_R)$
$\bar{\mathbf{x}}_R(0),\ \bar{\mathbf{u}}_R$ specified
where typically x_R and u_R are constants
Linear Small Perturbation Problem
$\delta\dot{\bar{\mathbf{x}}} = \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{x}}}\Big|_R \delta\bar{\mathbf{x}} + \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{u}}}\Big|_R \delta\bar{\mathbf{u}}$
$\delta \mathbf{y} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big|_R \delta\mathbf{x} + \frac{\partial \mathbf{h}}{\partial \mathbf{u}}\Big|_R \delta\mathbf{u}$
$\delta\bar{\mathbf{x}}(0) = \bar{\mathbf{x}}(0) - \bar{\mathbf{x}}_R(0)$
$\delta\bar{\mathbf{u}}(\mathbf{t}) = \bar{\mathbf{u}}(\mathbf{t}) - \bar{\mathbf{u}}_R$
Step2_Trim.m uses QuickTrim2.m
Nonlinear Reference Solution (e.g., constant solution)
$\bar{\mathbf{x}}_R, \vec{\mathbf{u}}_R$
Linear Solution (i.e., from Laplace Transforms)
$\delta\bar{\mathbf{x}}(\mathbf{t}),\ \delta\bar{\mathbf{y}}(\mathbf{t})$
Step5_Linearize_MATLAB7.m
Total Approximate Solution
$\bar{\mathbf{x}}(\mathbf{t}) = \bar{\mathbf{x}}_R + \delta\bar{\mathbf{x}}(\mathbf{t})$
$\bar{\mathbf{y}}(\mathbf{t}) = \bar{\mathbf{y}}_R + \delta\bar{\mathbf{y}}(\mathbf{t})$
$\bar{\mathbf{x}}(0) = \bar{\mathbf{x}}_R + \delta\bar{\mathbf{x}}(0)$
$\bar{\mathbf{u}}(\mathbf{t}) = \bar{\mathbf{u}}_R + \delta\bar{\mathbf{u}}(\mathbf{t})$

Figure 1 Solution Procedures for Nonlinear Differential Equations

Software Availability

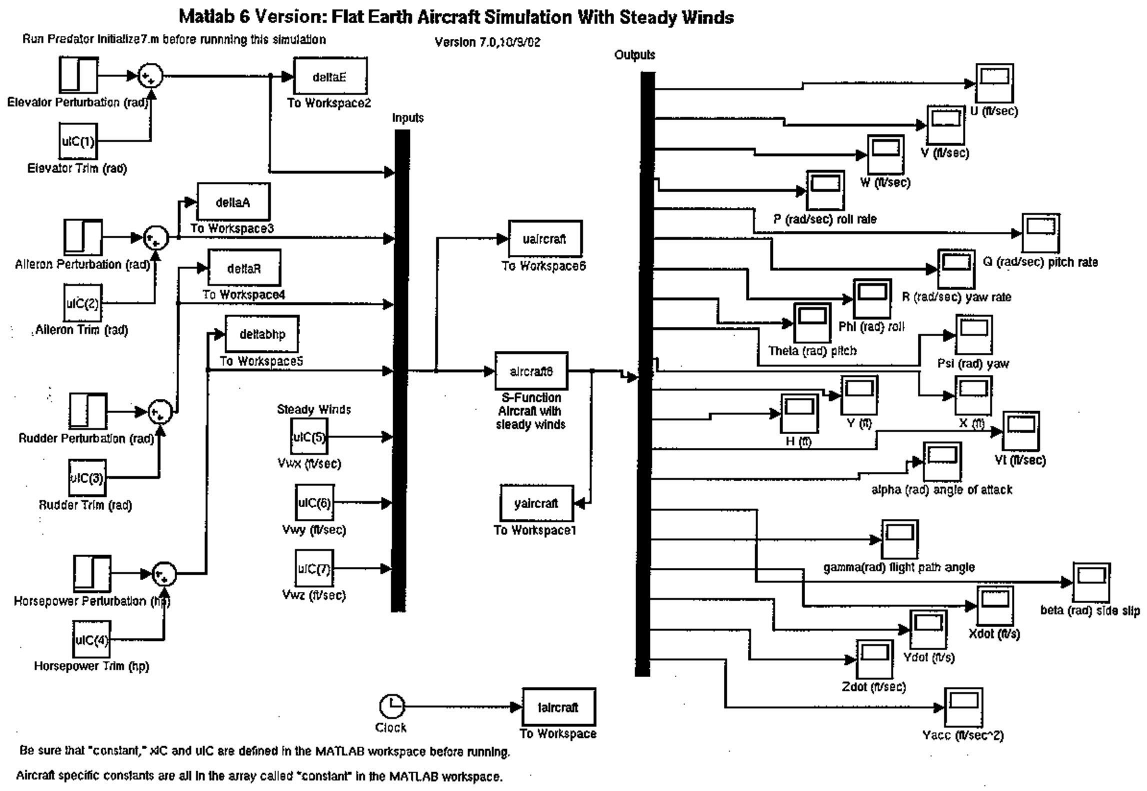
Wed Address

http://roger.ecn.purdue.edu/%7Eandrisan/Courses/AAE565_S2006/Buffer/

All MATLAB Scripts and models for the nonlinear simulation of a MPX5 UAV in a single zip file can be downloaded in a single zipped file (FlatEarth_9.2.zip). Alternately, you can view and save each file separately using a link to the folder that has all the individual MATLAB scripts and models.

SIMULINK Models

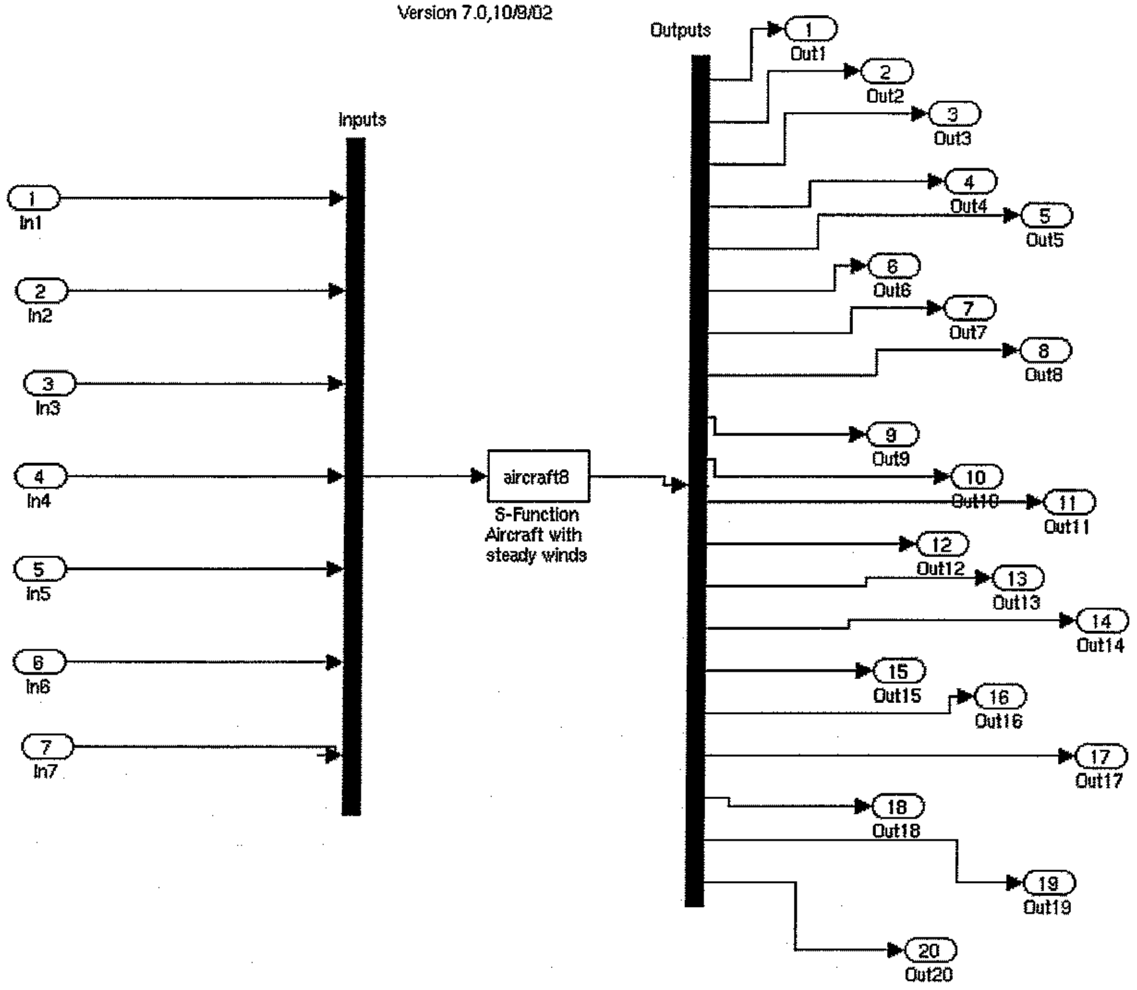
FlatEarth_MATLAB7.mdl



FlatEarth_ports_MATLAB7.mdl

Matlab 6 Version: Flat Earth Aircraft Simulation With Steady Winds

Version 7.0,10/8/02



MATLAB Scripts

Step0_ReadMe.m   Version 8.0 1/23/03

These MALAB scripts are designed to analyze and simulate arbitrary rigid aircraft over the flat earth. A vehicle description file called Basic_Constants_MPX5 is included allowing analysis of the MPX5 UAV.

It is important that these MATLAB scripts be run in order.

Step1_MPX5_Model.m
Step2_Trim.m
Step3_Simulink_SimMATLAB5.m or Step3_Simulink_SimMATLAB6.m
Step4_Plot_Noninear_Sim.m (optional step)
Step5_Linearize_MATLAB5.m or Step5_Linearize_MATLAB6.m
Step6_Longitudinal_Design.m
Step7_Lateral_Design.m (can be run after Step5)

MATLAB 5 and 6 versions of several files are provided allowing users of either version of MATLAB to use the software.

Step1_MPX5_Model.m   Version 8.0 1/23/03

OBJECTIVE: Develop the aerodynamic and thrust model for a particular aircraft.

INPUTS: a file of basic vehicle constants, e.g. Basic_Constants_MPX5.m

OUTPUTS: In the MATLAB workspace will be defined an array called constant.
Note: memory is cleared at the start of this script.

Step2_Trim.m   Version 8.0 1/23/03

OBJECTIVE: Trim the aircraft and develop initial conditions for the
SIMULINK nonlinear simulation.

INPUTS: the array called constant

OUTPUTS: In the MATLAB workspace will be defined initial conditions on the state and controls (xIC and uIC). The array constant, xIC and uIC are all required by the SIMULINK nonlinear simulation.

Step3_Simulink_SimMATLABx.m Version 8.0 1/23/03

OBJECTIVE: Simulate an aircraft in 6 degree of freedom motion
with nonlinear equations of motion and nonlinear
aerodynamic and thrust models. A flat earth and
rigid body dynamics are assumed.

INPUT: In the MATLAB workspace will be defined initial conditions
on the state and controls (xIC and uIC). The array constant, xIC and uIC
are all required by this SIMULINK nonlinear simulation.

OUTPUT: Arrays in the MATLAB workspace called taircraft and yaircraft
that contain an aray of time values and a matrix of output state
time histories.

Step4_Plot_Noninear_Sim.m Version 8.0 1/23/03

OBJECTIVE: Plot results from the nonlinear flat earth simulation.

INPUTS: In the MATLAB workspace must be an array of simulation times (taircraft)
and a matrix of simulation outputs (yaircraft). These are generally
created by the SIMULINK simulations FlatEarth_MATLABx.mdl.

OUTPUTS: Four figures, each with three subplots.

Step5_Linearize_MATLABx.m Version 8.0 1/23/03

OBJECTIVES: 1. Script to linearize the aircraft system assuming the nonlinear
aircraft model in SIMULINK model FlatEarth_MATLABx.mdl.
2. Perform a linear simulation for the same conditions as
used in the SIMULINK nonlinear simulation.
3. Overplot the nonlinear simulation results and the linear simulation
results in order to verify the accuracy of the linearization.
4. Split the 12th order system into the smaller longitudinal
and lateral-directional subsystems.
5. Save to binary files the longitudinal and lateral-directional
state space subsystems (modelLong.mat, modelLat.mat).
6. Determine if we are on the frontside or the backside of
the power required curve.

INPUTS: 1. In the MATLAB workspace must be an array of simulation times (taircraft)
and a matrix of simulation outputs (yaircraft). These are generally
created by the SIMULINK simulations FlatEarth_MATLABx.mdl.
2. In the MATLAB workspace will be defined initial conditions

on the state and controls (xIC and uIC) andthe array constant.

OUTPUTS: 3 figures of overplots,
plot of the power required curve,
2 output files (modelLong.mat, modelLat.mat),
transfer functions, poles, natural frequencis and damping ratios in the command window.

Step6_Longitudinal_Design.m Version 8.0 1/23/03

OBJECTIVE: Set up linear models for use in linear control system design for the longitudinal subsystem

INPUTS: The file modelLong.mat created by Step5_Linearize_MATLABx.m

OUTPUTS: Many transfer functions, poles, natural frequencies and damping ratios in the MATLAB command window. Also several linear time invariant systems are created in the MATLAB workspace f or subsequent analysis (Lsys, Ltfsys, Lzpksys).

NOTES: 1. This code can be run after Step5_Linearize_MATLABx.m has been run.
2. This code clears memory at the start if execution

Step7_lateral_Design.m Version 8.0 1/23/03

OBJECTIVE: Set up linear models for use in linear control system design for the lateral-directional subsystem

INPUTS: The file modelLat.mat created by Step5_Linearize_MATLABx.m

OUTPUTS: Many transfer functions, poles, natural frequencies and damping ratios in the MATLAB command window. Also several linear time invariant systems are created in the MATLAB workspace f or subsequent analysis (LDsys, LDtfsys, LDzpksys).

NOTES: 1. This code can be run after Step5_Linearize_MATLABx.m has been run.
2. This code clears memory at the start if execution

Complete Listing of Scripts and Model Files

Basic_Constants_MPX5.m
CL_0.M
CL_alpha.m
CL_alpha_dot.m
CL_de.m
CL_q.m
Check_Constants.m
Cl_beta.m
Cl_da.m
Cl_dr.m
Cl_p.m
Cl_r.m
Cm_0.m
Cm_a_dot.m
Cm_alpha.m
Cm_de.m
Cm_q.m
Cn_beta.m
Cn_da.m
Cn_dr.m
Cn_p.m
Cn_r.m
Cy_beta.m
Cy_da.m
Cy_dr.m
Cy_p.m
Cy_r.m
FlatEarth_MATLAB5.mdl
FlatEarth_MATLAB6.mdl
FlatEarth_ports_MATLAB5.mdl
FlatEarth_ports_MATLAB6.mdl

Make_Constants.m
NameScript.m
QuickTrim2.m

Step0_ReadMe.m
Step1_MPX5_Model.m
Step2_Trim.m
Step3_Simulink_SimMATLAB5.m
Step3_Simulink_SimMATLAB6.m
Step4_Plot_Nonlinear_Sim.m
Step5_Linearize_MATLAB5.m
Step5_Linearize_MATLAB6.m
Step6_Longitudinal_Design.m
Step7_Lateral_Design.m
TransformB2NED.m
TransformB2Wind.m
TransformNED2B.m
TransformWind2B.m
aeroforce.m
aeromoment.m
aircraft7.m
interlim1.m
interlim2.m
interlim3.m
modelLat.mat
modelLong.mat
nufun.m
rhofun.m
selector.m
thrust.m

Brief Description of Other MATLAB Functions and Scripts

Basic_Constants_MPX5.m contains basic constants for the MPX5. Basic constants include things like aspect ratio of the vertical tail and horizontal tail area.

CL_0.M computes the stability derivative CL0.

CL_alpha.m computes the stability derivative CLa (lift curve slope).

CL_alpha_dot.m computes the stability derivative Clalphadot.

CL_de.m computes the control derivative Clde (elevator effectiveness).

CL_q.m computes the stability derivative CLq.

Check_Constants.m is a script to perform a believability check on the array called constant which contains the stablility and control derivatives and inertia properties.

Cl_beta.m computes the stability derivative Clbeta (dihedral effect).

Cl_da.m computes the control derivative Clda (aileron effectiveness).

Cl_dr.m computes the control derivative Cldr (rolling moment due to rudder).

Cl_p.m computes the stability derivative Clp (damping in roll).

Cl_r.m computes the stability derivative Clr.

Cm_0.m computes the stability derivative Cm0.

Cm_a_dot.m computes the stability derivative Cmalphadot (lag of downwash effect).

Cm_alpha.m computes the stability derivative Cmalpha (pitch stiffness).

Cm_de.m computes the control derivative Cmde (elevator effectiveness).

Cm_q.m computes the stability derivative Cmq (damping in pitch).

Cn_beta.m computes the stability derivative Cnbeta (weathercock stability derivative).

Cn_da.m computes the control derivative Cnda (adverse/proverse aileron yaw).

Cn_dr.m computes the control derivative Cndr (rudder effectiveness).

Cn_p.m computes the stability derivative Cnp.

Cn_r.m computes the stability derivative Cnr (damping in yaw).

Cy_beta.m computes the stability derivative Cybeta (side force due to sideslip).

Cy_da.m computes the control derivative Cyda.

Cy_dr.m computes the control derivative Cydr.

Cy_p.m computes the stability derivative Cyp.

Cy_r.m computes the stability derivative Cyr.

Make_Constants.m uses Basic_Constants_MPX5 to compute the array called constant.

NameScript.m helps in the believability check performed by Check_Constants.

QuickTrim2.m trims the aircraft at a given speed and altitude in steady level flight.

TransformB2NED.m computes transformation matrix from body to North-East-Down frame.

TransformB2Wind.m computes transformation matrix from body to wind frame.

TransformNED2B.m computes transformation matrix North-East-Down to body frame.

TransformWind2B.m computes transformation matrix wind to body frame.

aeroforce.m computes the aerodynamic force vector in body axis components.

aeromoment.m compute the aerodynamic moment vector in body axis components.

aircraft7.m contains the equations of motion and output equations in s-function format for use in an s-function by SIMULINK.

interlim1.m is a 1-dimensional table look-up function with limits. If the inputs are beyond the range specified in the table, the outputs are limited to last value in the table and a non-fatal warning message is generated. Extrapolation is never performed. The informed student should check out these non-fatal warnings to verify that the code handled this ambiguous situation in the proper manner.

interlim2.m is a 3-dimensional table look-up function with limits. If the inputs are beyond the range specified in the table, the outputs are limited to last value in the table and a non-fatal warning message is generated. Extrapolation is never performed. The informed student should check out these non-fatal warnings to verify that the code handled this ambiguous situation in the proper manner.

interlim3.m is a 3-dimensional table look-up function with limits. If the inputs are beyond the range specified in the table, the outputs are limited to last value in the table and a non-fatal warning message is generated. Extrapolation is never performed. The informed student should check out these non-fatal warnings to verify that the code handled this ambiguous situation in the proper manner.

modelLat.mat is a MALLAB binary file generated by the statement

```
save modelLat aLD bLD cLD dLD Vt Hp aircraft
```

It contains the lateral-directional subsystem in state space form (e.g., a, b, c, d matrices).

modelLong.mat is a MALLAB binary file generated by the statement

```
save modelLong aL bL cL dL Vt Hp aircraft
```

It contains the longitudinal subsystem in state space form (e.g., a, b, c, d matrices).

nufun.m calculates kinematic viscosity in the standard atmosphere.

rhofun.m calculates air density in the standard atmosphere.

selector.m creates a smaller state space subsystem from a larger state space system by selecting certain states, control and outputs to be retained in the smaller subsystem.

thrust.m generated the thrust force and moment in body axis components.

Simulink Introduction

Get Matlab into the correct working directory. Click the Simulink symbol or type simulink in the command window.

```
>> simulink
```

From a Simulink window select New Model
Save as (give the file name you want, (test1))

Open a few Simulink libraries
    Sources, Sinks, Continuous, Math Operations

Drag Simulink blocks to the New Model window
    Integrator, step, scope

Go into simulation parameters
    Check the start time and stop time
    Click start simulation

Open Scope window and look at step response of integrator.

Add feedback
    Add summer and gain element.
    Change sign of feedback to -
    Simulate for various gains.

Examine .mdl file in editor window. Lots of text!

Make system a subsystem to simplify diagram.
    Highlight all elements between source and sink.
    Under the Edit menu select Create a subsystem.

Create a library of your own blocks for convenient re-use of work.

LINEARIZATION OF NONLINEAR EQUATIONS
By Dominick Andrisani

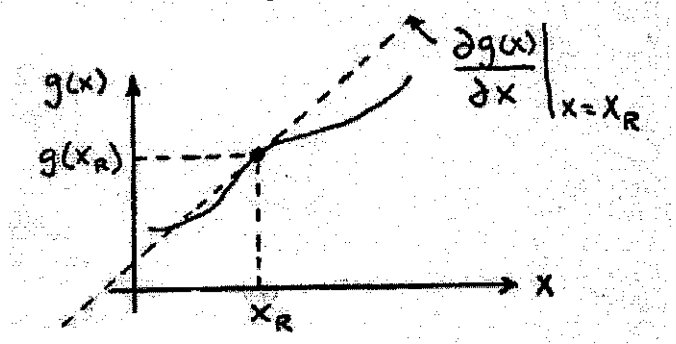
A. Linearization of Nonlinear Functions

A.1 Scalar functions of one variable.

We are given the nonlinear function g(x). We assume that g(x) can be represented using a Taylor series expansion about some point $\mathbf{X_R}$ as follows

$$g(x) = g(x)|_{x=x_R} + \frac{dg(x)}{dx}|_{x=x_R} (x - x_R) + \frac{1}{2!}\frac{d^2g(x)}{dx^2}|_{x=x_R} (x - x_R)^2$$

$$+ \text{ higher order terms}$$

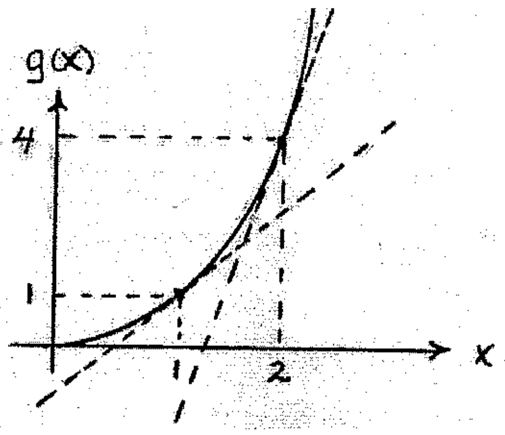


A linear approximation for g(x) involves taking only the first two terms

$$g(x) \approx g(x)|_{x=x_R} + \frac{dg(x)}{dx}\Big|_{x=x_R} (x - x_R)$$

This approximation is most accurate if $(x - x_R)$ is small so that the neglected higher order terms are negligible.

Example: $\mathbf{g(x) = x^2}$



Expanding g(x) about $\mathbf{x_R = 2}$ gives

$$\mathbf{g(x) \approx g(x)|_{x=x_R} + \frac{dg}{dx}|_{x=x_R} (x - x_R)}$$

$$\mathbf{\approx 2^2 + 2x|_{x=x_R} (x-2)}$$

$$\mathbf{= 4 + 4(x-2) = -4 + 4x}$$

Notice that this is a linear function of x. The simplification resulted because we evaluated all nonlinear terms at the number $\mathbf{x = x_R = 2}$. Because we evaluated the terms on the right hand side of the equation above at $\mathbf{x = x_R = 2}$, the only term that depends on x is the **x-2** term, and this is a linear term. This approximation to **g(x)** is valid near **x=2.**

We can generate another approximation to **g(x)** by expanding **g(x)** about $\mathbf{x_R}$=1 gives

$$\mathbf{g(x) = 1^2 + 2x|_{x=1} (x-1) = 1 + 2(x-1)}$$

$$\mathbf{= -1 + 2x}$$

This second approximation is valid near **x=1.** Clearly the linear approximation depends on the choice of reference point $\mathbf{x_R}$.

A.2 Scalar function of 2 variables

Given the nonlinear function $\mathbf{g(x_1, x_2)}$. This function can be represented by a Taylor series expansion about $\mathbf{x_{1_R}, x_{2_R}}$ as follows

$$g(x_1,x_2) = \underline{g(x_{1_R},x_{2_R})} + \frac{\partial g}{\partial x_1}\Big|_{x_1=x_{1_R},x_2=x_{2_R}} (x_1 - x_{1_R}) + \frac{\partial g}{\partial x_2}\Big|_{x_1=x_{1_R},x_2=x_{2_R}} \underline{(x_2 - x}$$

$$\frac{1}{2!}\left[\frac{\partial}{\partial x_1}\frac{\partial g}{\partial x_1}\Big|_{x_1=x_{1_R},x_2=x_{2_R}} (x_1 - x_{1_R})^2 + \frac{\partial}{\partial x_2}\frac{\partial g}{\partial x_2}\Big|_{x_1=x_{1_R},x_2=x_{2_R}} (x_2 - x_{2_R})^2\right]$$

$$+\frac{\partial}{\partial x_1}\frac{\partial g}{\partial x_2}\Big|_{x_1=x_{1_R},x_2=x_{2_R}} (x_1 - x_{1_R})(x_2 - x_{2_R}) + \text{h.o.t.}$$

A linear approximation of g can be obtained by retaining the first three terms above (underlined). The two variables in this problem can be associated together in a vector $\overline{x}$ as follows

$$g(\overline{x}) \text{ where } \overline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Example:

$$g(x_1,x_2) = x_1^2 \cos x_2$$

can be approximated about $x_{1_R} = 2,\ \ x_{2_R} = 0$ as follows

$$g(x_1,x_2) = (x_1^2 \cos x_2)\Big|_{x_{1_R}=2,\, x_{2_R}=0} + (2x_1 \cos x_2)\Big|_{x_{1_R}=2,\, x_{2_R}=0} (x_1 - 2)$$

$$-(x_1^2 \sin x_2)\Big|_{x_{1_R}=2,\, x_{2_R}=0} (x_2 - 0)$$

$$= 4 + 4(x_1 - 2) + 0 = -4 + 4x_1$$

The same function can be approximated about $x_{1_R} = 2 \quad x_{2_R} = \pi/4$

$$g(x_1,x_2) = 2^2 \cos(\frac{\pi}{4}) + (2x_1 \cos x_2)\Big|_{x_{1_R}=2,\, x_{2_R}=\frac{\pi}{4}} (x_1 - 2)$$

$$-(x_1^2 \sin x_2)\Big|_{x_{1_R}=2,\, x_{2_R}=\frac{\pi}{4}} (x_2 - \frac{\pi}{4})$$

$$= 0 + 0 - 4(x_2 - \frac{\pi}{4}) = \pi - 4x_2$$

Notice again how important the linearization point or reference point is to the linearized result.

A.3 Vector function of a vector of variables.

Let $\overline{g}(\overline{x})$ be an nx1 vector of nonlinear functions. Let $\overline{x}$ be an nx1 vector of variables

$$\overline{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}, \quad \overline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

A linear approximation about $\overline{x}_R$ is

$$\overline{g} \approx \overline{g}(\overline{x}_R) + \frac{\partial \overline{g}}{\partial \overline{x}}\Big|_{\overline{x}=\overline{x}_R} (\overline{x} - \overline{x}_R)$$

where

$$\overline{g}(\overline{x}_R) = \begin{bmatrix} g_1(x_{1_R}, & \cdots & x_{n_R}) \\ \vdots & & \\ g_n(x_{1_R}, & \cdots & x_{n_R}) \end{bmatrix} = \text{nx1 vector}$$

$$\frac{\partial \overline{g}}{\partial \overline{x}} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \cdots & \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} \cdots & \frac{\partial g_n}{\partial x_n} \end{bmatrix} = \text{Jacobian Matrix} = \text{nxn matrix}$$

A.4 Accuracy of linearized solution.

When we approximate $\overline{g}(\overline{x})$ by retaining only the linear terms, we must guarantee that the deleted terms, i.e., the h.o.t. are negligible. This is true only when $\overline{x} - \overline{x}_R$ is small, i.e. when the perturbations from the reference point are small.

B. Linearization on Nonlinear Differential Equations in First Order Form

B.1 First order form

Nonlinear differential equations in first order form can be written as

$$\dot{\bar{x}} = \bar{g}(\bar{x}, \bar{u}), \qquad \bar{x}(0)$$

where

$$\bar{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \bar{g} = \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}, \quad \dot{\bar{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix}, \quad \bar{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}$$

Note that $\bar{u}$ represents specified forcing functions and $\bar{x}(0)$ is a specified initial condition vector.

Example B.1a

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \bar{g} = \begin{bmatrix} x_2{}^2 - u^2 \\ -x_1{}^2 + 1 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2{}^2 - u^2 \\ -x_1{}^2 + 1 \end{bmatrix}$$

B.2 The reference or trim solution

When we were linearizing nonlinear functions, we saw how important the choice of reference point was. In linearizing nonlinear differential equations, we are also concerned with the reference about which we linearize. However, we are now interested in obtaining a linearized solution *valid for all time*. This requires that we linearize around a reference solution, which is valid for all time.

Let $\bar{x}_R(t)$ be a known solution to the nonlinear differential equation with specified forcing function $\bar{u}_R(t)$ and specified initial condition $\bar{x}_R(0)$. i.e.,

$$\dot{\bar{x}}(t) = \bar{g}(\bar{x}_R(t), u_R(t)) \qquad \bar{x}_R(0)$$

$x_R(t)$ is said to be the *reference solution to the nonlinear differential equation.*

Example B.1b

For the differential equations given in Example B.1a

$$\bar{x}_R(t) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad u_R(t) = 1, \quad \dot{\bar{x}}_R(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

is a constant solution to the nonlinear differential equation. Verify this fact for yourself by substituting this solution into the differential equation given in Example B.1a. Please keep straight in your mind the difference between a differential equation (e.g. $\dot{\mathbf{x}} = \mathbf{x}$) and a solution to a differential equation (e.g. $\mathbf{x} = \mathbf{0}$ **for** $\dot{\mathbf{x}} = \mathbf{x}$).

Example B.1c

For the differential equations given in Example B.1a

$$\mathbf{x}_R(t) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \mathbf{u}_R(t) = -1 \quad \dot{\bar{\mathbf{x}}}_R = [\begin{matrix} 0 \\ 0 \end{matrix}]$$

is another constant solution to the nonlinear differential equations.

Example B.1d

For the differential equations given in Example B.1a

$$\bar{\mathbf{x}}_R = \begin{bmatrix} \mathbf{x}_1 = \pm 1 \\ \mathbf{x}_2 = \pm \mathbf{u}_R = \mathbf{const} \end{bmatrix} \quad \mathbf{u}_R = \mathbf{const} \quad \dot{\mathbf{x}}_R = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

is a constant solution to the nonlinear differential equations for any constant.

B.3 Linearization about a reference solution

Let $\bar{\mathbf{x}}_R(\mathbf{t}),\ \bar{\mathbf{u}}_R(\mathbf{t})$ be a reference solution. We now want to find a linearized solution to the nonlinear differential equation about this reference solution.

We again expand $\bar{\mathbf{g}}(\bar{\mathbf{x}})$ in a Taylor series expansion about $\bar{\mathbf{x}}_R$ and $\bar{\mathbf{u}}_R$ i.e.,

$$\dot{\bar{\mathbf{x}}} = \bar{\mathbf{g}}(\bar{\mathbf{x}}_R, \bar{\mathbf{u}}_R) + \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{x}}}\Big|_{x=\bar{x}_R,\ u=\bar{u}_R} (\bar{\mathbf{x}} - \bar{\mathbf{x}}_R) + \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{u}}}\Big|_{x=\bar{x}_R,\ u=\bar{u}_R} (\bar{\mathbf{u}} - \bar{\mathbf{u}}_R)$$

+h.o.t.

The linear approximation is obtained by assuring that $\bar{\mathbf{x}} - \bar{\mathbf{x}}_R$ and $\bar{\mathbf{u}} - \bar{\mathbf{u}}_R$ are small enough that the h.o.t. can be neglected.

B.4 Definition of small distribution variables

Define

$$\delta\bar{\mathbf{x}} = \bar{\mathbf{x}} - \mathbf{x}_R$$
$$\delta\bar{\mathbf{u}} = \bar{\mathbf{u}} - \bar{\mathbf{u}}_R$$
$$\delta\dot{\bar{\mathbf{x}}} = \dot{\bar{\mathbf{x}}} - \dot{\bar{\mathbf{x}}}_R$$

For the linearized solution to be valid, these perturbations must be "small."

B.5 Separation of the linearized differential equations into two parts

Assuming that the perturbations are small, we can write the approximation to the differential equations as

$$\dot{\bar{x}} = \bar{g}(\bar{x}_R, \bar{u}_R) + \frac{\partial \bar{g}}{\partial \bar{x}}|_R \, (\bar{x} - \bar{x}_R) + \frac{\partial \bar{g}}{\partial \bar{u}}|_R \, (\bar{u} - \bar{u}_R)$$

we can now substitute the small perturbation variables

$$\underline{\dot{\bar{x}}_R} + \delta\dot{\bar{x}} = \underline{g(\bar{x}_R, \bar{u}_R)} + \frac{\partial g}{\partial x}|_R \, \delta\bar{x} + \frac{\partial g}{\partial u}|_R \, \delta\bar{u}$$

In the equation above we have simplified the notation with $|_R$ to denote $|_{\bar{x}=\bar{x}_R,\ \bar{u}=\bar{u}_R}$.

Notice that the underlined terms are numerically equal from the definition of reference solution. Since they are equal, they can be cancelled out leaving

$$\delta\dot{\bar{x}} = \frac{\partial g}{\partial x}|_R \, \delta\bar{x} \quad + \quad \frac{\partial g}{\partial u}|_R \, \delta\bar{u}$$

This is a set of linear small perturbation differential equations. In summary, the original nonlinear problem

$$\dot{\bar{x}} = \bar{g}(\bar{x}, \bar{u}), \qquad \bar{x}(0)$$

with solution $\bar{x}(t)$ for specified input u(t) has been decomposed into two separate problems.

- The reference problem

$$\dot{\bar{x}}_R = \bar{g}(\bar{x}_R, \vec{u}_R)$$

  with initial condition $\bar{x}_R(0)$ with solution $\bar{x}_R(t)$ to input $\bar{u}_R(t)$

- The small perturbation problem

$$\delta\dot{\bar{x}} = \frac{\partial \bar{g}}{\partial \bar{x}}|_R \, \delta\bar{x} + \frac{\partial \bar{g}}{\partial \bar{u}}|_R \, \delta\bar{u}$$

  with initial condition

$$\delta\bar{x}(0) = \bar{x}(0) - \bar{x}_R(0)$$

  with solution $\delta\bar{x}(t)$ to input $\delta\bar{u}(t)$.

Finally the total approximate solution is given by the entire solution procedure is shown in Figure 1.

$$\bar{x}(t) = \bar{x}_R(t) + \delta\bar{x}(t)$$

B.6 On picking a reference solution

Any solution to $\dot{\overline{\mathbf{x}}} = \overline{\mathbf{g}}(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ makes a good reference solution but these solutions can be hard to find. An easier set of solutions are constant solutions i.e., solutions so that $\dot{\overline{\mathbf{x}}}_{\mathbf{R}}(\mathbf{t}) = \overline{\mathbf{0}}$ and $\overline{\mathbf{x}}_{\mathbf{R}}(\mathbf{t}) =$ constant for $\overline{\mathbf{u}}_{\mathbf{R}}(\mathbf{t}) =$ constant. For constant reference solutions, finding the reference solution to a *nonlinear differential equation* becomes a problem of finding the solution to a *nonlinear algebraic equation*

$$\mathbf{g}(\overline{\mathbf{x}}_{\mathbf{R}}, \overline{\mathbf{u}}_{\mathbf{R}}) = \overline{\mathbf{0}}$$

B.7 Linearization Example

$$\dot{\overline{\mathbf{x}}}(\mathbf{t}) = \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_2^{\ 2} - \mathbf{u}^2 \\ -\mathbf{x}_1^{\ 2} + \mathbf{1} \end{bmatrix}$$

a) Choice of Reference Solution

To simplify our choice, assume that the reference solution is constant, i.e., $\dot{\mathbf{x}}_1 = \dot{\mathbf{x}}_2 = \mathbf{0}$. This requires that $\mathbf{x}_2^{\ 2} - \mathbf{u}^2 = \mathbf{0}$ and $-\mathbf{x}_1^{\ 2} + \mathbf{1} = \mathbf{0}$. These equations can be satisfied whenever

$$\mathbf{x}_2^{\ 2} = \mathbf{u}^2 \text{ and } \mathbf{x}_1^{\ 2} = \mathbf{1}$$

Values of $\mathbf{x}_1$ and $\mathbf{x}_2$ which satisfy these equations are

$$\mathbf{x}_2 = \pm\mathbf{u} \text{ where } \mathbf{u} \text{ is any constant}$$

$$\mathbf{x}_1 = \pm\mathbf{1}$$

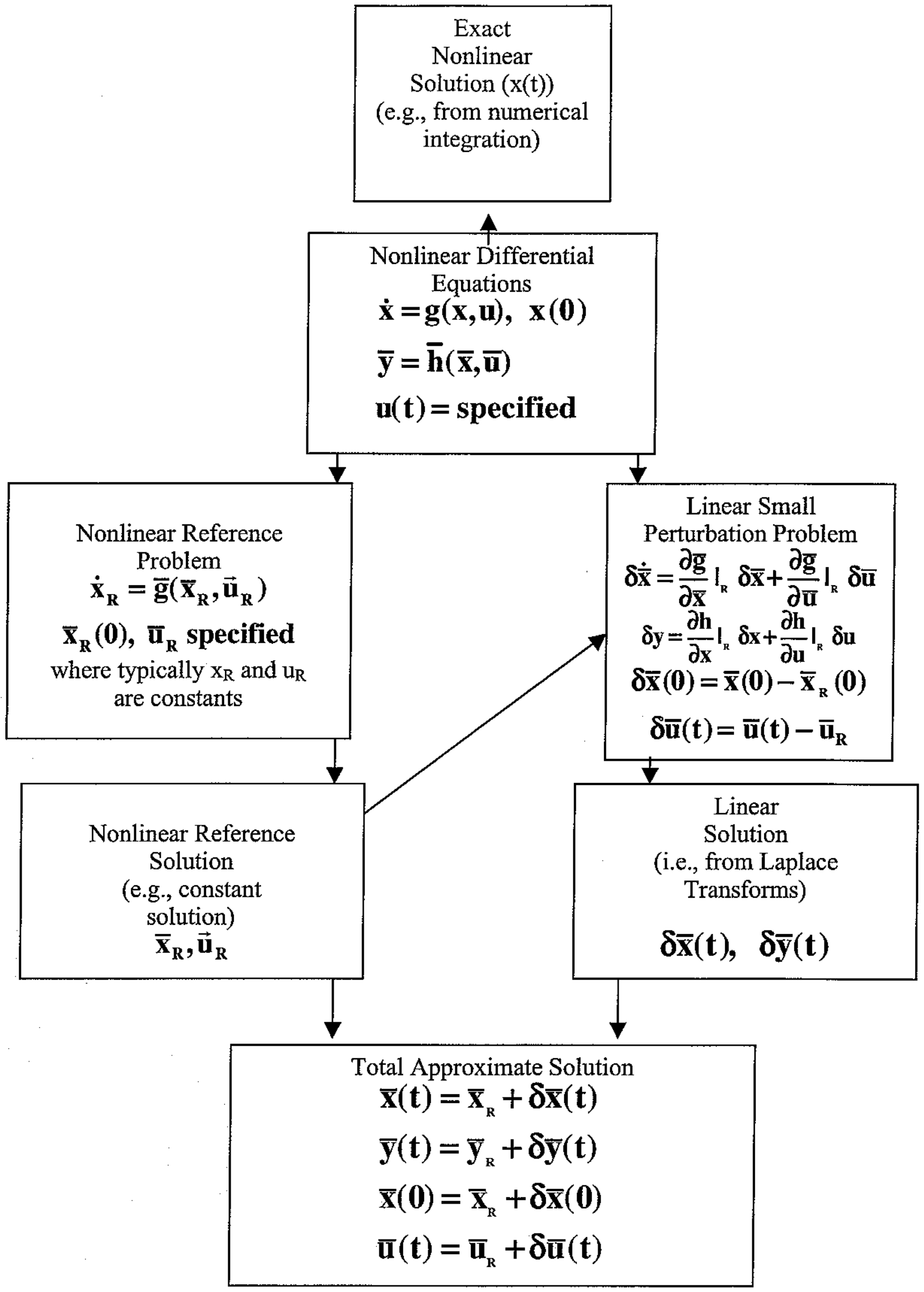

Exact Nonlinear Solution (x(t)) (e.g., from numerical integration)
Nonlinear Differential Equations
$\dot{x} = g(x,u),\ x(0)$
$\bar{y} = \bar{h}(\bar{x},\bar{u})$
$u(t) = \text{specified}$
Nonlinear Reference Problem
$\dot{x}_R = \bar{g}(\bar{x}_R,\vec{u}_R)$
$\bar{x}_R(0),\ \bar{u}_R$ specified
where typically x_R and u_R are constants
Linear Small Perturbation Problem
$\delta\dot{\bar{x}} = \frac{\partial \bar{g}}{\partial \bar{x}}|_R\ \delta\bar{x} + \frac{\partial \bar{g}}{\partial \bar{u}}|_R\ \delta\bar{u}$
$\delta y = \frac{\partial h}{\partial x}|_R\ \delta x + \frac{\partial h}{\partial u}|_R\ \delta u$
$\delta\bar{x}(0) = \bar{x}(0) - \bar{x}_R(0)$
$\delta\bar{u}(t) = \bar{u}(t) - \bar{u}_R$
Nonlinear Reference Solution (e.g., constant solution)
$\bar{x}_R, \vec{u}_R$
Linear Solution (i.e., from Laplace Transforms)
$\delta\bar{x}(t),\ \delta\bar{y}(t)$
Total Approximate Solution
$\bar{x}(t) = \bar{x}_R + \delta\bar{x}(t)$
$\bar{y}(t) = \bar{y}_R + \delta\bar{y}(t)$
$\bar{x}(0) = \bar{x}_R + \delta\bar{x}(0)$
$\bar{u}(t) = \bar{u}_R + \delta\bar{u}(t)$

Figure 1 Solution Procedures for Nonlinear Differential Equations

We will consider two different reference solutions

$$\underline{\textbf{Ref.\# 1}} \qquad\qquad \underline{\textbf{Ref.\# 2}}$$

$$\bar{x}_R(t)=\begin{bmatrix}+1\\+1\end{bmatrix},\ u_R(t)=+1 \qquad \bar{x}_R(t)=\begin{bmatrix}-1\\-1\end{bmatrix},\ u_R(t)=-1$$

$$\bar{x}_R(0)=\begin{bmatrix}1\\1\end{bmatrix} \qquad x_R(0)=\begin{bmatrix}-1\\-1\end{bmatrix}$$

b) Small Perturbation Equations of Motion

$$\delta\dot{\bar{x}}=\frac{\partial\bar{g}}{\partial\bar{x}}|_R\,\delta\bar{x}\ +\ \frac{\partial\bar{g}}{\partial\bar{u}}|_R\,\delta u$$

where $\delta\bar{x}=\bar{x}-\bar{x}_R \quad \delta u=u-u_R$

$$A=\frac{\partial\bar{g}}{\partial\bar{x}}=\begin{bmatrix}\frac{\partial g_1}{\partial x_1}&\frac{\partial g_1}{\partial x_2}\\ \frac{\partial g_2}{\partial x_1}&\frac{\partial g_2}{\partial x_2}\end{bmatrix},\quad B=\frac{\partial\bar{g}}{\partial u}=\begin{bmatrix}\frac{\partial g_1}{\partial u}\\ \frac{\partial g_2}{\partial u}\end{bmatrix}$$

$$\frac{\partial\bar{g}}{\partial\bar{x}}=\begin{bmatrix}0&2x_2\\-2x_1&0\end{bmatrix},\quad \frac{\partial\bar{g}}{\partial u}=\begin{bmatrix}-2u\\0\end{bmatrix}$$

Using Ref. #1 $\bar{x}_R=\begin{bmatrix}1\\1\end{bmatrix},\ u_R=1$

$$\begin{bmatrix}\delta\dot{x}_1\\\delta\dot{x}_2\end{bmatrix}=\begin{bmatrix}0&+2\\-2&0\end{bmatrix}\begin{bmatrix}\delta x_1\\\delta x_2\end{bmatrix}+\begin{bmatrix}-2\\0\end{bmatrix}\delta u$$

Using Ref. #2 $\bar{x}_R=\begin{bmatrix}-1\\-1\end{bmatrix},\ u_R=1$

$$\begin{bmatrix}\delta\dot{x}_1\\\delta\dot{x}_2\end{bmatrix}=\begin{bmatrix}0&-2\\+2&0\end{bmatrix}\begin{bmatrix}\delta x_1\\\delta x_2\end{bmatrix}+\begin{bmatrix}+2\\0\end{bmatrix}\delta u$$

c. The Linear Solution for Reference #1

(1) $\delta\dot{\bar{x}}_1=2\delta x_2-2\delta u$

(2) $\delta\dot{x}_2 = -2\delta x_1$

take Laplace transforms

(1) $s\delta x_1(s) - \delta x_1(0) = 2\delta x_2(s) - 2\delta u(s)$

(2) $s\delta x_2(s) - \delta x_2(0) = -2\delta x_1(s)$

multiply (2) by s

$$s^2\delta x_2(s) - s\delta x_2(0) = -2s\delta x_1(s)$$

multiply (1) by –2

$$-2s\delta x_1(s) = -4\delta x_2(s) + 4\delta u(s) - 2\delta x_1(0)$$

set these equal

$$s^2\delta x_2(s) - s\delta x_2(0) = -4\delta x_2(s) + 4\delta u(s) - 2\delta x_1(0)$$
$$\delta x_2(s)\left[s^2 + 4\right] = s\delta x_2(0) - 2\delta x_1(0) + 4\delta u(s)$$
$$\delta x_2(s) = \frac{s\delta x_2(0) - 2\delta x_1(0)}{s^2+4} + \left(\frac{4}{s^2+4}\right)\delta u(s)$$

The first term on the right gives initial condition response. The second term on the right contains the transfer function $\dfrac{\delta x_2(s)}{\delta u(s)} = \dfrac{4}{s^2+4}$.

To find $\delta x_1(t)$ take the inverse Laplace transform. From (2)

$$s\delta x_2(s) - \delta x_2(0) = -2\delta x_1(s)$$
$$\delta x_1(s) = -\frac{1}{2}\left[s\delta x_2(s) - \delta x_2(0)\right]$$

To find the solutions $\delta x_1(t)$ and $\delta x_2(t)$ you must be given the input $\delta u(t)$ and the initial conditions $\left(\delta x_1(0), \delta x_2(0)\right)$. Then the solutions can be found using inverse Laplace transforms.

d) Total Solution for Reference #1

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} x_{1_R}(t) \\ x_{2_R}(t) \end{bmatrix} + \begin{bmatrix} \delta x_1(t) \\ \delta x_2(t) \end{bmatrix} = \begin{bmatrix} 1+\delta x_1(t) \\ 1+\delta x\ \ (t) \end{bmatrix}$$

$$u(t) = u_R(t) + \delta u(t) = 1 + \delta u(t)$$

$$\overline{x}_1(0) = \begin{bmatrix} 1+\delta x_1(0) \\ 1+\delta x_2(0) \end{bmatrix}$$

e) Comment:

For this procedure to be valid the perturbations must be small, i.e.,

all must be small.

Suppose we have the nonlinear problem with

$$\mathbf{x_1(0) = 1.01}$$

$$\mathbf{x_2(0) = .99}$$

$$\mathbf{u(t) = 1.0 + .01 \sin \omega t}$$

then we can use Ref.#1 and then we can have

$$\mathbf{\delta x_1(0) = .01}$$

$$\mathbf{\delta x_2(0) = -.01}$$

$$\mathbf{\delta u(t) = .01 \sin \omega t}$$

On the other hand if for the nonlinear problem we have

$$\mathbf{x_1(0) = -1.01}$$

$$\mathbf{x_2(0) = -.99}$$

$$\mathbf{u(t) = -1 - .01 \sin \omega t}$$

We would use Ref. #2 with

$$\mathbf{\delta x_1(0) = -.01}$$

$$\mathbf{\delta x_2(0) = .01}$$

$$\mathbf{\delta u(t) = -.01 \sin \omega t}$$

C. Output Equations

Often nonlinear differential equations are associated with nonlinear output equations. This may come about in modeling the sensors aboard an aircraft. The sensors are often nonlinear functions of the state vector and control vector. Output equations can be expressed as follows.

$$\bar{\mathbf{y}} = \bar{\mathbf{h}}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \text{ where is } \bar{\mathbf{y}} \text{ a px1 vector}$$

This can also be linearized about reference solution $\mathbf{x}_R$ and $\mathbf{u}_R$ as follows.

$$\mathbf{y} = \underline{\mathbf{y}_R} + \delta\mathbf{y} \approx \underline{\mathbf{h}(\mathbf{x},\mathbf{u})|_R} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}}|_R\,(\mathbf{x} - \mathbf{x}_R) + \frac{\partial \mathbf{h}}{\partial \mathbf{u}}|_R\,(\mathbf{u} - \mathbf{u}_R)$$

The underlined terms are equal by definition of y on the reference and can be cancelled out on both sides of the equation. That leaves the linear small perturbation output equations in terms of small perturbation variables.

$$\delta\mathbf{y} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}|_R\,\delta\mathbf{x} + \frac{\partial \mathbf{h}}{\partial \mathbf{u}}|_R\,\delta\mathbf{u} = \mathbf{C}\cdot\delta\mathbf{x} + \mathbf{D}\cdot\delta\mathbf{u}$$

The total output equation in linear form is then given by the following.

$$\mathbf{y}(\mathbf{t}) \approx \mathbf{y}_R + \delta\mathbf{y}(\mathbf{t})$$

D. Stability

One of many possible definitions of dynamic stability for nonlinear systems is given in terms of the eigenvalues of the Jacobian matrix, $\mathbf{A} = \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{x}}}|_R = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} \end{bmatrix}_R$.

If all the eigenvalues of **A** have negative real parts we say that the reference solution, $(\mathbf{x}_R, \mathbf{u}_R)$, is **stable.**

If at least one of the eigenvalue of **A** has a positive real part we say that the reference solution, $(\mathbf{x}_R, \mathbf{u}_R)$, is **unstable.**

If at least one eigenvalues of **A** has a zero real part, and if all the other eigenvalues have negative real parts, we can draw **no conclusion** about the stability of the reference solution, $(\mathbf{x}_R, \mathbf{u}_R)$.

E. Concluding Comments

We have seen how the solution to nonlinear differential equations can be found by decomposing the problem into two simpler parts. The reference part is simpler because it is often a nonlinear algebraic problem. The second small perturbation part is simpler because it often involves solving linear differential equations with constant coefficients. The total approximate solution to the original nonlinear differential equation was shown to be the sum of the two simpler parts.