ASWING: Practical Use Manual, Version 1.0

Leonardo AVONI

September 15, 2025

Contents

Chapter 1

ISAE-Supaero/ENAC Documentation

1.1 Introduction

This document is made to explain, in a quick and concise way what ASWING is, its reliability, how it works and mostly how one can use it in practice. This report will not go in the details of all the math behind the software, already covered by the following references:

- ASWING Technical Description, by M. Drela [2], steady and unsteady
- ASWING documentation (.txt file, [2])
- Experimental validation of ASWING aerodynamics[5], propellers [6], structures[7] and aeroelasticity [8] reports, by R. Jan [5, 6, 7, 8]
- Energy Harvesting on Flexible UAVs (Thesis, R.Jan, [3])
- Enhancing ASWING Flight Dynamics Simulations with Closed-Loop Control for Flexible Aircraft, by Avoni [1]

Note that the ASWING version used is 5.98, with the Unsteady Extension.

1.2 Aswing General Theory

1.2.1 ASWING Results Reliability

The PhD thesis of Jan focused in great part on the quantitative assessment of ASWING, through comparison with high-order CFD or experimental results. His findings are summarized in Table 1.1.

Table 1.1: ASWING 5.98 capabilities, as evaluated within reports by Jan [5, 6, 7, 8]

Category	**Details**
Aerodynamics	
Wings	Simulation Capabilities: • Rectangular, sweptback, dihedral, quasi-elliptical, crescent... • Accuracy can be improved using polar graphs (depends on quality of polars) • Includes steady wake interaction Reliability Conditions: • Aspect ratio (AR) > 5 • Sweep or dihedral $\leq 45°$ • Reynolds number $> 100{,}000$ • Valid up to stall

Category	Details
Unsteady Wing Aerodynamics	• Based on Theodorsen's theory • Includes 3D wing effects • Only reliable in attached regime • No LEV, hence no dynamic stall modelization
Ailerons	• Predicts aileron rolling moment • Does not predict yaw moment (asymmetric drag unmodeled) • Valid for aileron deflections up to $\pm 20°$ (aileron stall beyond) • Not reliable in post-stall conditions
Tandem Wings	• Predicts downstream wing loss of lift due to upstream wing wake • Predicts stall individually for each surface
Winglets	• Computes drag reduction from single-surface winglets • Tip sails not supported
Fuselage	• Circular cross-section only • Fuselage lift distribution accurate overall except at tail, leading to overestimated pitch-up moment • Reliable if the Aspect Ratio (AR) > 5
Compressible Flow	• Prandtl-Glauert accurately captures lift slope for a clean wing configuration. • Drag predictions are weak, needing improvement with AoA and Mach dependence. • Compressible flow analysis is unreliable, with accurate lift data but wide errors in drag data.
Ground effect	• Reliably modeled for clean wing configurations up to half-chord altitudes
Propulsion	
Propeller Performance in Non-Axial Flow	• Axial thrust well-predicted up to $60°$ propeller axis angle when corrected • Reliability tied to engine/propeller data quality • Non-axial efforts poorly modeled • Propeller cone impact ignored
Propeller impact on downstream wing	• Reliable spanwise propeller position impact on downstream wing • Unreliable vertical propeller position impact on downstream wing
Elasticity	

Category	Details
What beams can be modeled	• Non-straight, twisted beams • Supports large deflections (non-linear) • Includes structural damping (Kelvin-Voigt) • Handles non-uniform beams • Accounts for shear, extensional, bending, and torsion stresses • Allows arbitrary locations for tension, elastic, and center of gravity axes • Incorporates point masses
Reliability	• Predicts static nonlinear deflection for high aspect ratio beams. • Good predictions for the first five structural modes • Neglecting some extensional/bending terms has limited impact on the modal response • Effectively models the impact of concentrated weights in spanwise and chordwise positions on wing modal response. • Useful for studying tank or nacelle position effects on wing flutter response.
Aeroelasticity	
Quasi-Steady	• Models steady aeroelastic deformations • Captures wing tailoring effects, except for negative bending-torsion coupling • Predicts aileron/flap reversal • Identifies divergence nature (torsional vs. mode coalescence)
Flutter and Gust Response	• Predicts flutter speed and frequency • Accounts for nacelle/tank position effects on flutter characteristics • Predicts whirl flutter speed for tiltrotor configurations (from literature) • Models gust response for reduced frequency k< 0.2 - 0.5
Other	• Models folding wingtip devices • Post-flutter limit cycle oscillations (LCOs) predictions unreliable
Other Points	
	• Supports linear, airspeed and altitude bilinear-scheduled controllers only • Opensource • Not widely used by community • No OPENMDAO coupling yet

1.2.2 Structural and Aerodynamic Equations

ASWING is a simulation tool designed for the dynamic modeling and control of flexible aircraft, particularly under subsonic, attached-flow conditions. It integrates nonlinear structural dynamics with potential-flow aerodynamics and allows for the simulation of both open-loop and closed-loop flight scenarios, incorporating control laws and gust disturbances. The aircraft in ASWING is modeled as a connected structure of beam elements—either lifting surfaces or fuselage members—each discretized along its span with a coordinate t, and characterized by spanwise-distributed properties such as mass, inertia, stiffness, chord or aerodynamic coefficients. These properties are provided via tabulated input and interpolated along t using cubic splines. Beam joints between elements may be rigid or flexible, allowing for multibody simulations. The aircraft model also supports singular elements such as lumped masses, point engines, struts (cables), beam joints, and sensors. The ASWING aircraft model is displayed in Figure 1.1.

The model uses three reference frames: the **Earth frame** (X,Y,Z), an inertial frame where body positions and attitudes are expressed; the **body frame** (x,y,z), attached to the aircraft, used for velocities,

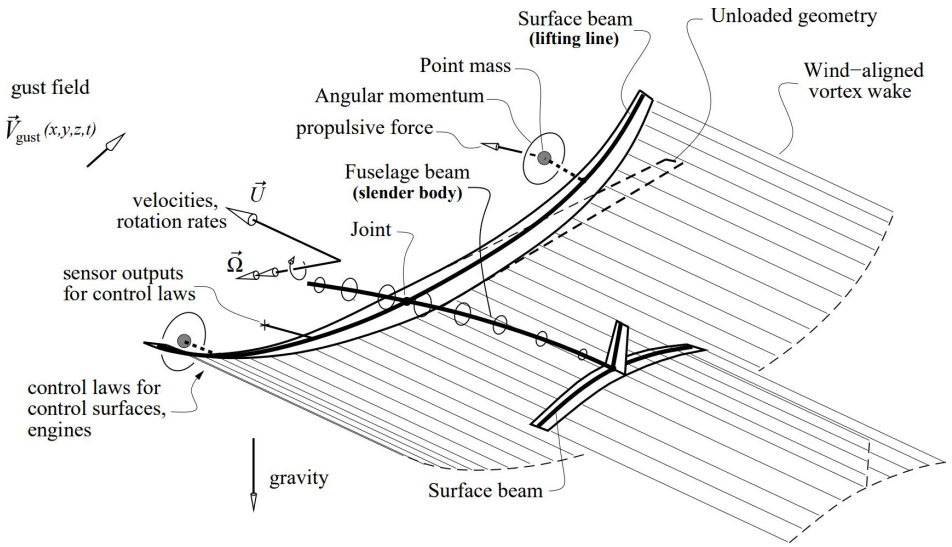


Figure 1.1: Typical ASWING aircraft model scheme, including most elements available in the software. From [2]

angular rates, and control inputs, and as the reference for local positions; and the **local frame** (c,s,n), defined along structural elements to express sectional quantities such as loads and deformations. The three frames are shown in Figure 1.2.

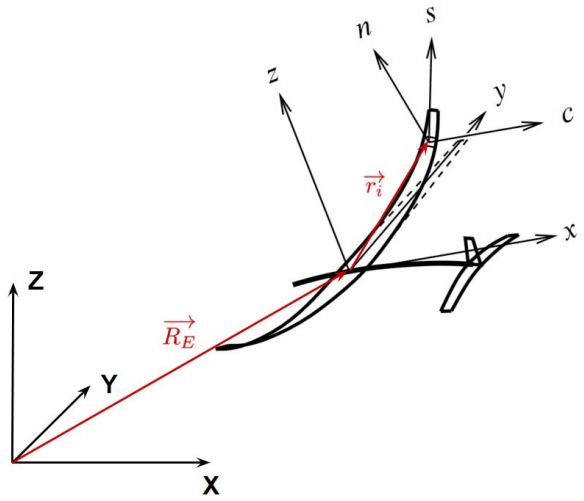


Figure 1.2: Main axis system used in ASWING. Only one (c,s,n) coordinate system is illustrated; other systems (e.g., Prandtl–Glauert and engine axes) are omitted for brevity

ASWING's aerodynamic model is based on potential flow theory, assuming high Reynolds numbers and subsonic Mach numbers, allowing simplification of the Navier-Stokes equations into Laplace's equation. Lifting surfaces are represented by a nonlinear lifting-line theory enhanced with unsteady wake modeling; the resultant spanwise-distributed circulation fulfilling the local Kutta condition is responsible for lift. Horseshoe vortices are shed into a wake sheet to capture unsteady aerodynamic behavior, leading to unsteady aerodynamics aligned with Theodorsen theory [10]. Fuselage beams lift on the other hand are modeled using slender-body theory. For both lifting and non-lifting bodies, added-mass effects are accounted. Non-induced drag is introduced via user-defined coefficients, and ground effect is simulated using the mirror-image method. Prandtl-Glauert corrections can be used if necessary. Tridimensional gusts profiles are modeled under the frozen-flow assumption.

Aircraft propulsion is ensured by point force, augmented if needed by an extended actuator disk model for propellers, and an additional propulsive jet to model jet/wing interaction.

The structural formulation in ASWING is based on an extended Euler-Bernoulli nonlinear beam

theory, suitable for long, slender wings and fuselage segments. The equations of motion include fully coupled structural, aerodynamic, and rigid-body dynamics (*aircraft states* in Equation 1.1), solved simultaneously for each timestep through Newton-Raphson iterations, and propagated in time through a backward-difference time integration scheme. These equations are capable of simulating highly flexible airframes undergoing steady or transient motion. ASWING supports a range of simulations, including steady-state trimming for general helical flight paths (e.g., straight flight, coordinated turns), time-domain simulations involving gusts and maneuvers, and linearized frequency-domain analyses for eigenmodes and harmonic excitations (excitations by control surfaces, engines or harmonic gusts profiles).

1.2.3 Simulations Available in ASWING

ASWING allows computations of steady-state aircraft behavior (trimming points). Those points are flight scenarios where the following quantities are constant:

- vertical velocity, in Earth Frame U_Z
- airspeed V_{IAS}
- climb rate γ
- bank angle θ
- turn radius R

This case is basically a helical flight path. The simplest version of such scenario is straight leveled flight.

The different types of free-flight simulations (straight, leveled trimming point calculation, balanced turn trimming state, free flight...) can be solved by correct constraint selection, as specified in subsection 1.2.6.

ASWING can also perform transient simulation (time domain aircraft dynamics). Those simulations are done by solving iteratively (Newton's method) the equations of motion (as in the steady case) for each time step.

ASWING is also capable of performing forced response analysis (forced aileron deflection) and eigenmode analysis. Those computations are performed in the frequency domain, around a previously computed, trimmed point (linearized solution).

Once an eigenmode analysis is done, it is possible to build a Reduced-Order Model (ROM), defining the simplified dynamics of the aircraft, based on its eigenvalues and eigenvectors.

1.2.4 States, Inputs and Outputs

The simulation at each time step k involves solving three sets of equations:

$$\begin{cases} \mathbf{0} = \tilde{\mathbf{r}_k}\left(\mathbf{x}_k, \dot{\mathbf{x}}_k, \mathbf{u}_k\right) & \textit{aircraft states} \\ \mathbf{y}_k = \mathbf{y}_k\left(\mathbf{x}_k, \dot{\mathbf{x}}_k, \mathbf{u}_k\right) & \textit{outputs} \\ \mathbf{0} = \mathbf{c_k}\left(\mathbf{x_k}, \mathbf{y_k}, \mathbf{u_k}\right) & \textit{control} \end{cases} \tag{1.1}$$

Note that ASWING Technical Description [2] specifies $\tilde{\mathbf{r}_k}$ and $\mathbf{c}_k$ combined in a single $\mathbf{r}_k = \mathbf{0}$ equation. The state vector $\mathbf{x_k}$, output $\mathbf{y_k}$ and input $\mathbf{u_k}$ are defined in Equation 1.2, Equation 1.4, Equation 1.5, where the variables are expressed at timestep k, which is omitted for clarity. The state vector $\mathbf{x_k}$ contains both global quantities (body frame states or beam joints states) and spanwise-distributed states. The latter include for example $\mathbf{r_i}$ indicating the aircraft beam deflections, with i indicating the spanwise index across all aircraft beams. Another distributed quantity is the $A_1, A_2, \ldots, A_F$ serie, containing all the circulation Fourier across all beams. The control input vector $\mathbf{u}$ contains the commands sent by the user to the ASWING control law, which will affect the state of control surfaces and engines at each time step.

$$\begin{aligned} \mathbf{x} = \Big(& \mathbf{r}_i, \boldsymbol{\theta}_i, \mathbf{u}_i, \boldsymbol{\omega}_i, \mathbf{F}_i, \mathbf{M}_i, && \textit{spanwise section states} \\ & \Delta\mathbf{r}_J, \Delta\boldsymbol{\theta}_J, \mathbf{M}_J, \mathbf{F}_J, && \textit{beam joints states} \\ & A_1, A_2, \ldots, A_F, && \textit{circulation Fourier coefficients} \\ & \mathbf{R}_E, \boldsymbol{\Theta}, \mathbf{U}, \boldsymbol{\Omega}, \mathbf{a}_o, \boldsymbol{\alpha}_o, && \textit{body frame states} \\ & \delta_{f_1}, \delta_{f_2}, \ldots, \delta_{f_{Nf}}, \Delta_{e_1}, \Delta_{e_2}, \ldots, \Delta_{e_{Ne}}, && \textit{flap and engines states} \\ & \Delta_{g_1}, \ldots, \Delta_{g_4}, \mathbf{e_j} \Big) && \textit{forcing gusts and errors} \end{aligned} \tag{1.2}$$

$$\mathbf{e_j} = \int \left[(V_{j_s} - V_{j_c}), (\alpha_{j_s} - \alpha_{j_c}), (\beta_{j_s} - \beta_{j_c}), (\boldsymbol{\theta_{j_s}} - \boldsymbol{\theta_{j_c}}), (\boldsymbol{\omega_{j_s}} - \boldsymbol{\omega_{j_c}}), (\mathbf{a_{j_s}} - \mathbf{a_{j_c}}) \right]^T dt \tag{1.3}$$

$$\mathbf{y} = (V_S, \alpha_S, \beta_S, \Phi_S, \Theta_S, \Psi_S, \ldots) \tag{1.4}$$

$$\mathbf{u} = \left(\mathbf{y}, \delta_{f_1,c}, \ldots, \delta_{f_{Nf},c}, \Delta_{e_1,c}, \ldots, \Delta_{e_{Ne}}\right) \tag{1.5}$$

The components of output vector $\mathbf{y}$ indicated in Equation 1.4 are far from being exaustive. In reality, $\mathbf{y}$ can be built within ASWING to include several of the elements in the following non-exhaustive list:

- Body Position and Attitude $\mathbf{R}_E, \boldsymbol{\Theta}$
- Body Velocities $\mathbf{U}, \boldsymbol{\Omega}$
- Body Accelerations $\mathbf{a}_o, \boldsymbol{\alpha}_o$
- Airspeed V, sideslip β, angle of attack α
- Flaps and Engines $\delta_{f_1}, \ldots, \delta_{f_{Nf}}, \Delta_{e_1}, \ldots, \Delta_{e_{Ne}}$
- Drag, Lift and Moment L, D, M
- Lift, Drag and Moment coefficients C_L, C_D, C_M
- Center of Gravity coordinates x_{CG}, y_{CG}, z_{CG}
- Local position $\mathbf{r}_i$ and attitude $\boldsymbol{\theta}_i$
- Local velocities $\mathbf{u}_i, \boldsymbol{\omega}_i$
- Local structural efforts $\mathbf{F}_i, \mathbf{M}_i$
- Local airspeed V_j, sideslip β_j, angle of attack α_j
- Circulation Γ_j (computed from $A_1, \ldots, A_K$)
- Integral error $\mathbf{e_j}$

A scheme indicating ASWING single timestep solving is shown in Figure 1.3

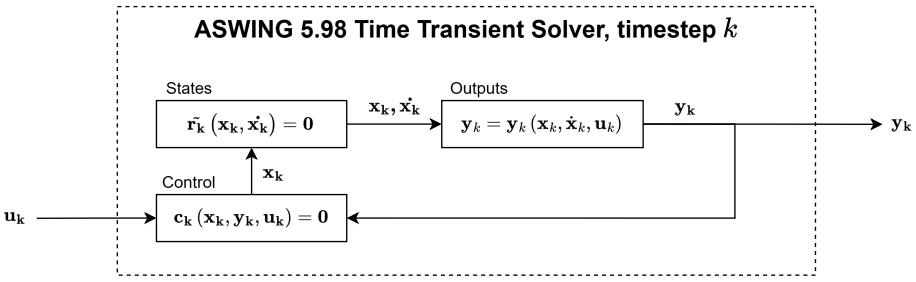


Figure 1.3: ASWING internal procedure for single timestep k solving. Current timestep state $\mathbf{x_k}$, state derivative $\mathbf{\dot{x}_k}$, inputs $\mathbf{u_k}$ and outputs $\mathbf{y_k}$ are solved for the same timestep using Newton-Raphson iterations. Input is provided by the user, along with simulation constraints and settings

ASWING solves 1.1 once for each trimming condition, or iteratively over successive time steps in the case of a time-transient simulation. For steady simulations, once a steady-state operating point is defined, ASWING can perform linearization around it for frequency-domain analysis. This includes forced responses to harmonic control surface or engine excitation, but also eigenmode extraction—providing coupled eigenpairs. The latter is particularly relevant for flutter prediction, divergence analysis, and general stability assessments.

Using the computed eigenpairs, ASWING can also produce reduced-order models (ROMs) by mode truncation, suitable for control synthesis, as well as a complex state-space model.

1.2.5 Spanwise Motion Equations

ASWING solves, at each timestep, the equation of motion for the whole aircraft. This is done by solving each spanwise element equation of motion. A local beam element composing one of the beams of the airframe is shown in Figure 1.4

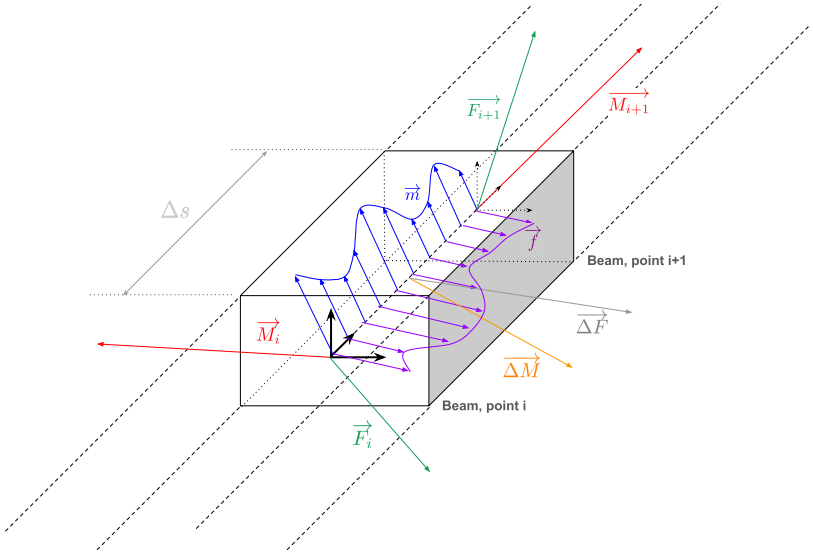


Figure 1.4: Beam element between position i and $i+1$. The scheme includes structural forces and moments at position i ($\overrightarrow{F_i}$, $\overrightarrow{M_i}$) and $i+1$ ($\overrightarrow{F_{i+1}}$, $\overrightarrow{M_{i+1}}$), the distributed forces and moments ($\vec{f}$, $\vec{m}$) and concentrated forces and moments ($\overrightarrow{\Delta F}$, $\overrightarrow{\Delta M}$)

,

For each beam element (discretized), as shown in Figure 1.4, the motion equations to solve (placed at position i) are the following:

$$\overrightarrow{M_{i+1}} - \overrightarrow{M_i} + \overrightarrow{m_a}\Delta s + \overrightarrow{\Delta M} + \Delta\vec{r} \times \overrightarrow{F_{i+1}} = \vec{0} \quad (1.6)$$

$$\underbrace{\overrightarrow{F_{i+1}}}_{\text{Term 1}} - \underbrace{\overrightarrow{F_i}}_{\text{Term 2}} + \underbrace{\vec{f_a}\Delta s}_{\text{Term 3}} + \underbrace{\overrightarrow{\Delta F}}_{\text{Term 4}} \underbrace{\quad}_{\text{Term 5}} = \vec{0} \quad (1.7)$$

$$\overrightarrow{m_a} = \frac{1}{\Delta s}\int_{s=s_i}^{s_{i+1}} \vec{m}\, ds \quad (1.8)$$

$$\vec{f_a} = \frac{1}{\Delta s}\int_{s=s_i}^{s_{i+1}} \vec{f}\, ds \quad (1.9)$$

$$\Delta s = s_{i+1} - s_i \quad (1.10)$$

$$\Delta r = r_{i+1} - r_i \quad (1.11)$$

- Term 1 and 2 indicate the structural forces and moments due to beam bending at the limits of the local beam element.
- Term 3 indicates the distributed forces and moments such as lift , drag, apparent mass and mass acceleration efforts.
- Term 4 indicates the concentrated forces and moments acting on such beam element (point masses, engines, strut and joint efforts)
- Term 5 is used to account for the structural forces impact on the bending moment, when performing Newton's second law halfway through position i and $i+1$ (he contribution of distributed loads on the moment is alerady accounted within $\vec{m}$ distribution).
- Note that Equation 1.6 and Equation 1.7 sum up to $\vec{0}$ since the inertial loads are already included in Term 3.

1.2.6 User-Provided Constraints

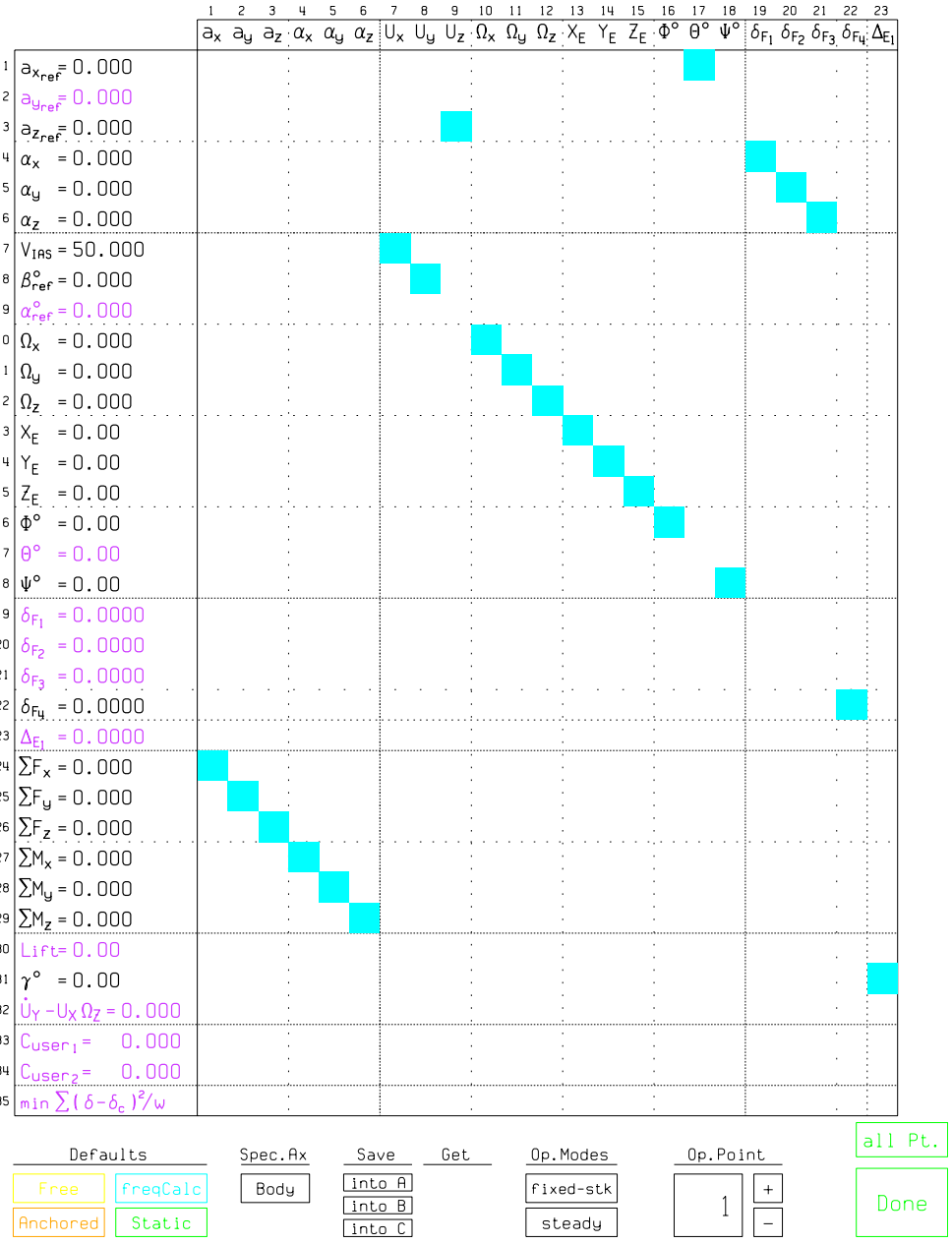


Figure 1.5: Toggle menu, accessible from the OPER menu, used to set the different constraints during simulations. The case shown concerns the straight leveled flight trimming point calculation of an aircraft.

In ASWING, constraints are set through the menus of Figure 1.5. It is accessible in the OPER, MODE or BODE menu by typing T (mouse completion) or % (keyboard completion). The method uses a table to link the state variables shown in the top row, to constraints listed on the left columns. The link between the two is done using blue squares. Clicking inside the table places a blue square, linking a variable to a certain constraint.

Clicking on the constraints allows to change the reference value (clicking on the $V_{IAS} = 50.000$ will create an empty line on the terminal to write the new desired V_{IAS} to keep); the same operation can be performed with the ! command in the OPER menu (by typing !V 50 for example). Constraints shown in purple are not used.

Clicking on a state variable shown on the top row will deactivate it; resulting in that variable keeping its value computed for previous trimming point. This helps for example in case of symmetric problems, where the value is known beforehand to be zero, or in case of time-transient operations with fixed-stick flight. Disabling a variable will save computations, and numerical errors on that variable will be avoided.

Deactivating a variable will change the color of that variable to blue.

Explanations of the constraints set in Figure 1.5 for trimmed leveled flight are provided below:

- First 6 columnns: 2nd Law of Newton (F and M include also the inertia forces, thus it's $\Sigma\overrightarrow{F} = \overrightarrow{0}$ and $\Sigma\overrightarrow{M} = \overrightarrow{0}$)
- Columnn 7: fixing the airspeed to 50 (units specified on the terminal, from `.asw` file)
- Columnn 8: no sideslip
- Columnn 9: U_z such that no vertical acceleration is perceived
- Columnn 10 to 12: no angular velocity (no rotations)
- Columns 13 to 15: Earth coordinates are kept constant
- Columns 16 to 18: Attitude of the aircraft is zero, except for the pitch that is such that no acceleration in x is perceived (body axis)
- Columns 19 to 21: finding the correct control surface deflection to make the aircraft have no angular acceleration over x,y,z
- Columnn 22: δ_4 is fixed to 0 (we could have clicked on that variable)
- Column 23: the engine power is such that the aircraft path angle with-respect-to the horizontal (γ) is zero (horizontal flight)

In conclusion, the toggles simplify in: solve the equations of motion such that there is no acceleration on the x-z body plane (which is also the X-Z Earth plane since Φ and Ψ are zero). Sideslip to 0, airspeed to 50, no rotations (apart for the pitch). Hence the only variables to be found are the pitch, control surfaces positions 1 to 3 and engine power.

Note that the constraints do not link exactly 1:1 the variables to what is wanted. For instance, constraint on columnn 8 does not fix U_x = constant, but the horizontal speed, combination of U_x and U_z

For the unsteady scenario, the toggle menu of Figure 1.6 is available. The constraints differ slightly from the steady case.

The constraints of Figure 1.6 can be explained as follows:

- First 6 columnns: 2nd Law of Newton (F and M include also the inertia forces, thus it's $\Sigma\overrightarrow{F} = \overrightarrow{0}$ and $\Sigma\overrightarrow{M} = \overrightarrow{0}$)
- Columns 7 to 18: aircraft velocity, angular velocity, position and attitude vary according to the aircraft kinematics
- Columns 19 to 23: control surfaces and engines are forced to their trimming point values

When setting the constraints, default settings are available:

- `Free`: sets the default parameters used for a free-flying aircraft (not necessarily trimmed)
- `freqCalc`: sets the simulation to "unsteady"
- `Static`: settings for quasi-static simulations. Maybe it is to assess the impact of dynamics against static scenarios
- `Anchored`: the aircraft reference point is fixed to the ground

Note that the toggle menu allows to switch between variables expressed in Body or Earth axis (it is easier to set some constraints in some coordinate axis). It also allows to set the simulation to be transient or steady. Finally, it can allow a control law to be run, or to simply fly the aircraft in fixed-stick.

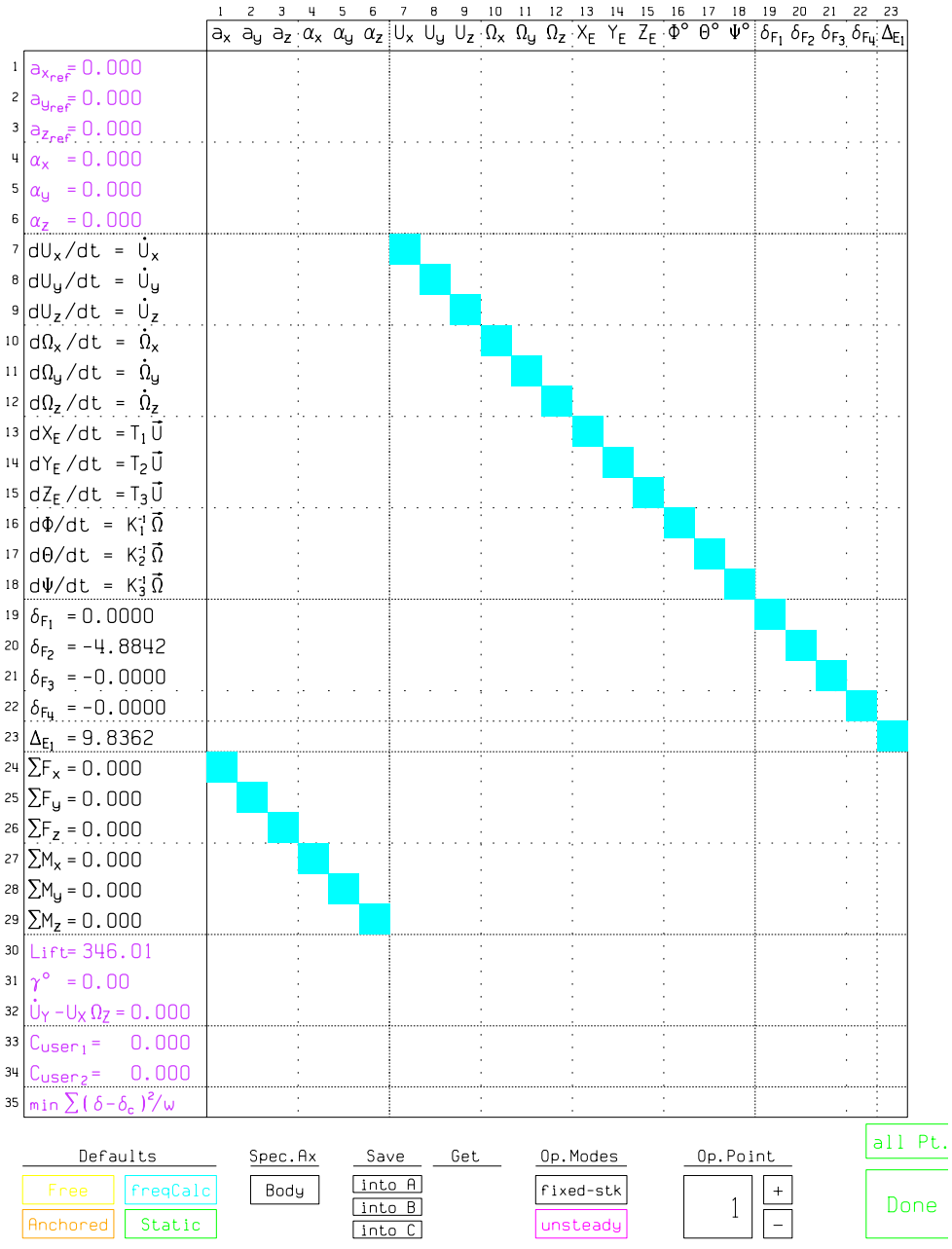


Figure 1.6: Toggle menu, accessible from the OPER menu, used to set the different constraints during time transient simulations.

1.2.7 Closed-Loop Control

ASWING's control framework relies on proportional control between outputs $\mathbf{y_k}$ and inputs $\mathbf{u_k}$ according to user-specified parameters for the control equations $\mathbf{c}$. . While the default implementation assumes fixed-stick laws (such as $\mathbf{c} = \delta_{f_1} - \delta_{f_{1_c}} = \mathbf{0}$ for flap 1), more complex schemes such as scheduled PID controllers with saturation and anti-windup can be incorporated[4]. Control dynamics are resolved simultaneously with the aircraft dynamics at every timestep, ensuring consistency and numerical stability without time-delay; meaning that $u_k = f(x_k, x_{k-1}, x_{k-2}...)$, thus avoiding the impact of time discretization on the control scheme performance. This characteristic was checked by plotting and verifying the timeseries results using OPER>P

The working scheme of linear controllers implementable in ASWING is shown in Figure 1.3, while the contribution by Jan can be viewed in Figure 1.7.

Current control architecture within ASWING has the advantage of being already integrated in the

Fortran code, thus making the resolution fast. Next, the fact that the aircraft state $\mathbf{x}$, output $\mathbf{y}$ and input $\mathbf{u}$ are solved at each timestep make the aircraft independent to time-delay due to the timestep duration; making the selected time discretization only impact the aerodynamic numerical convergence. Finally, when performing eigenmode calculations in the `MODE` menu, ASWING can compute the eigenpairs of the closed-loop system, hence allowing pole placement control synthesis ASWING. Extensions in the possible linear control capabilities of ASWING were made[4], allowing sensor and actuation dynamics, input/output saturation, as well as anti-windup.

Additional control extensions were provided by Avoni [1], allowing the implementation of any kind of controller, as well as extraction of proper state space matrices for the aircraft in trimmed conditions.

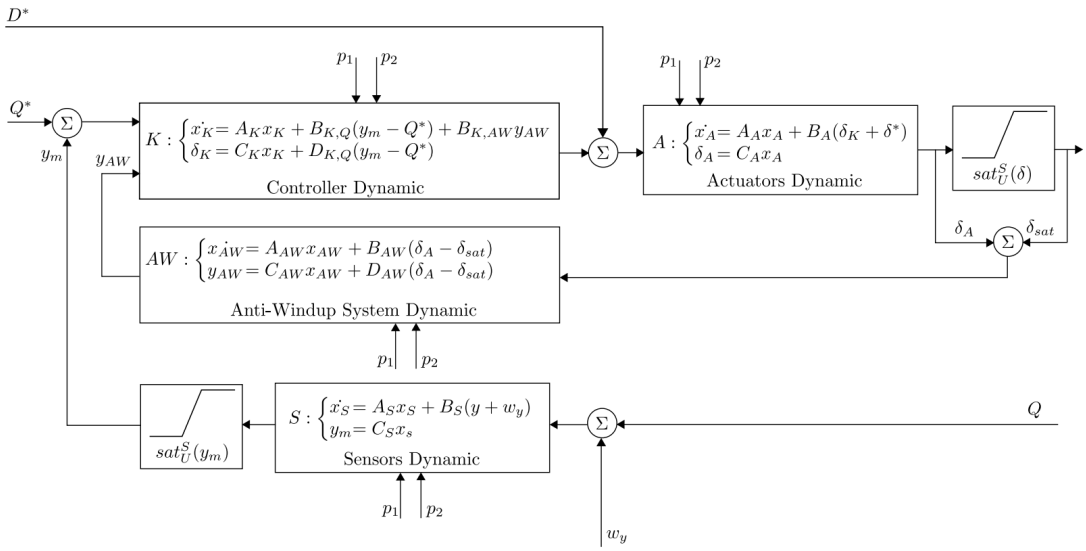
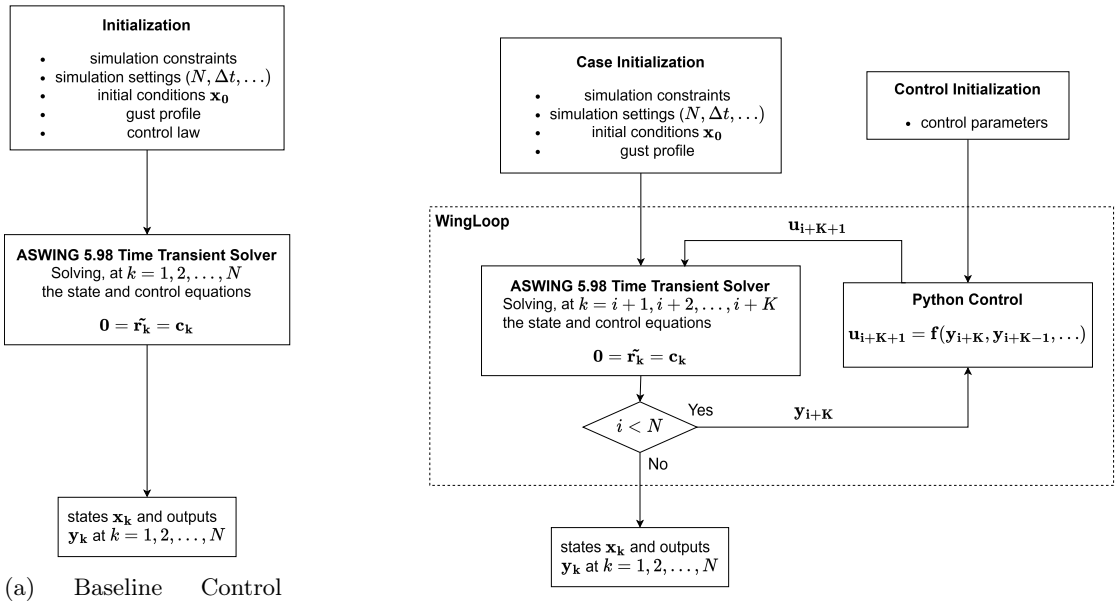


Figure 1.7: Control scheme implemented by Jan [4]. Q^* is the commanded input (provided either via trimming point state, or via specified input as in subsection 1.3.7, when typing `X input_file`), D^* is the trimmed point specifics, while p_1 and p_2 are two variables that can be used for gain scheduling (usually altitude[1] and airspeed). Finally, Q is the aircraft state.



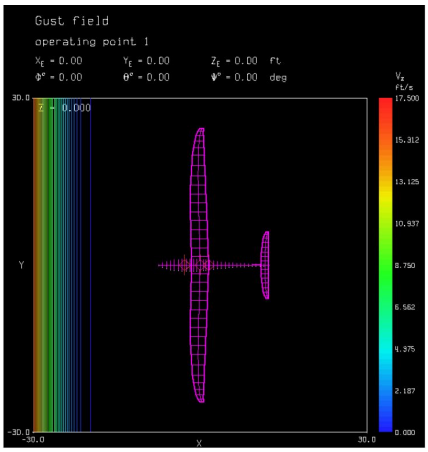
(a) Baseline Control Scheme

(b) WingLoop Control Framework

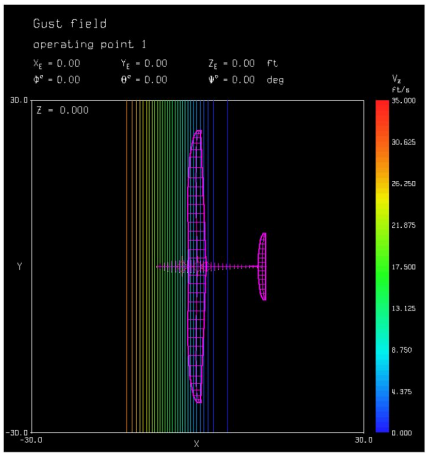
Figure 1.8: Baseline ASWING time-transient simulation (Figure 1.8a), and WingLoop time-transient simulation (Figure 1.8b) pipelines

1.2.8 User-Provided Gust

ASWING allows modeling of gusts fields fulfilling frozen-flow assumption. Note however that in case a gust profile has been specified (with `GGET` and/or `GMOD`, subsection 1.3.4) the position of the gust compared to the aircraft will impact the flow not only for unsteady simulations, **but also for steady computations**. Hence, performing a trimming point computation for the configurations shown in Figure 1.9a will not lead to the same trimming position as the one obtained from Figure 1.9b because of the flow difference around the aircraft.



(a) Aircraft outside a vertical gust



(b) Aircraft inside a vertical gust

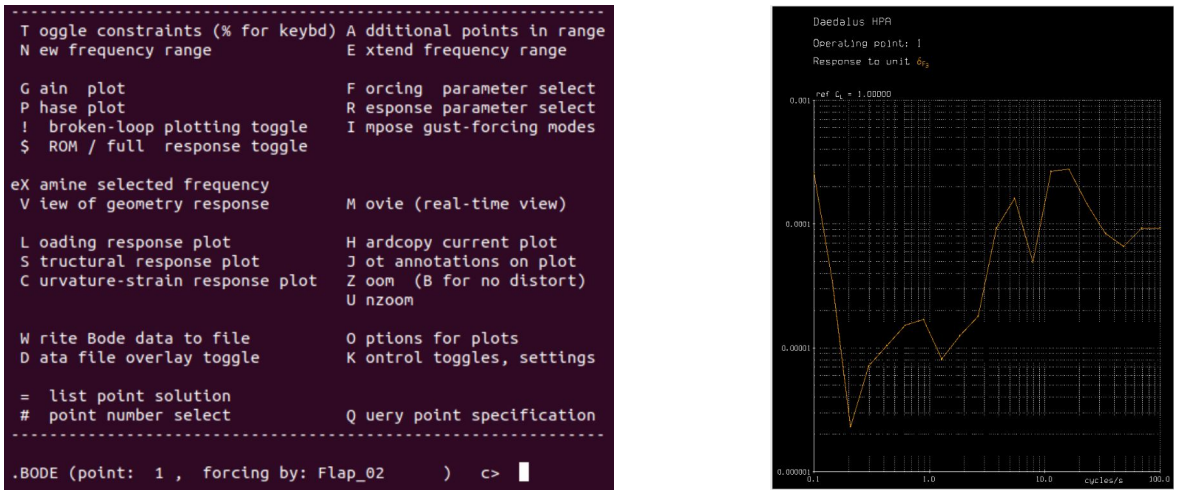
Figure 1.9: Aircraft and gust in ASWING

1.3 ASWING Practical Use

1.3.1 Practical Considerations

ASWING runs on macOS and Linux. Its interface is by terminal only (no Graphical User Interface). Plots and images are generated by pop-up windows during the code execution. Input and output files are text files.

The typical ASWING view is composed of two windows: the terminal and the figure. An example is shown in Fig. Figure 1.10



(a) Terminal view of the BODE menu of ASWING

(b) Bode plot generated by ASWING for the Daedalus aircraft

Figure 1.10: Typical screen views of ASWING during execution

ASWING was initially designed to work in single screen only. It is however possible to use it with several screens if care is put on the proper window size. If crashes occur (in general the plot window freezes), restarting ASWING is the only option. It is recommended, in that case, to lower the window size using `PLPA`

1.3.2 ASWING Data Flow

ASWING data flow is shown in Fig. Figure 1.11. The files typically used by ASWING are separated in different prefixes:

- `.asw`: text parameter and geometry file
- `.con`: text control-law file
- `.pnt`: text operation-points file
- `.set`: text settings file
- `.gust`: text gust perturbation file
- `.state`: binary file containing all states (steady and transient) computed by ASWING. Useful as a recovery, save or backup

Note that the suffix of the file (`.asw`, `.con`, ...) does not actually matter within ASWING execution (one could manually load a geometry file named `test.test` if needed). However, the use of prefixes allows to have the same initial name, but different roles, while also loading all the useful files at startup. For instance, `hawk.asw` contains the geometry, `hawk.con` contains the corresponding control laws, `hawk.set` contains the setting used... If ASWING is started using `../bin/aswing hawk`, all those files will be automatically loaded. If different names have been used, the individual files should be loaded one by one. Refer to Section subsection 1.3.3.

The definition of the various parts of an `.asw` file is already provided in the ASWING official manuals, copied in chapter 2.

Other default file prefixes used by ASWING also exist, and will be found for outputs. For instance:

- `.ps`: image file, containing hardcopies of ASWING plots
- `.t`: text file containing tabulated results
- `.rom`: text file containing Reduced Order Model information (from `MODE` menu)
- `.fXX`: text file containing the frequency analysis number XX (from `BODE` menu)
- `.eXX`: text file containing the eigenvector number XX (from `MODE` menu)

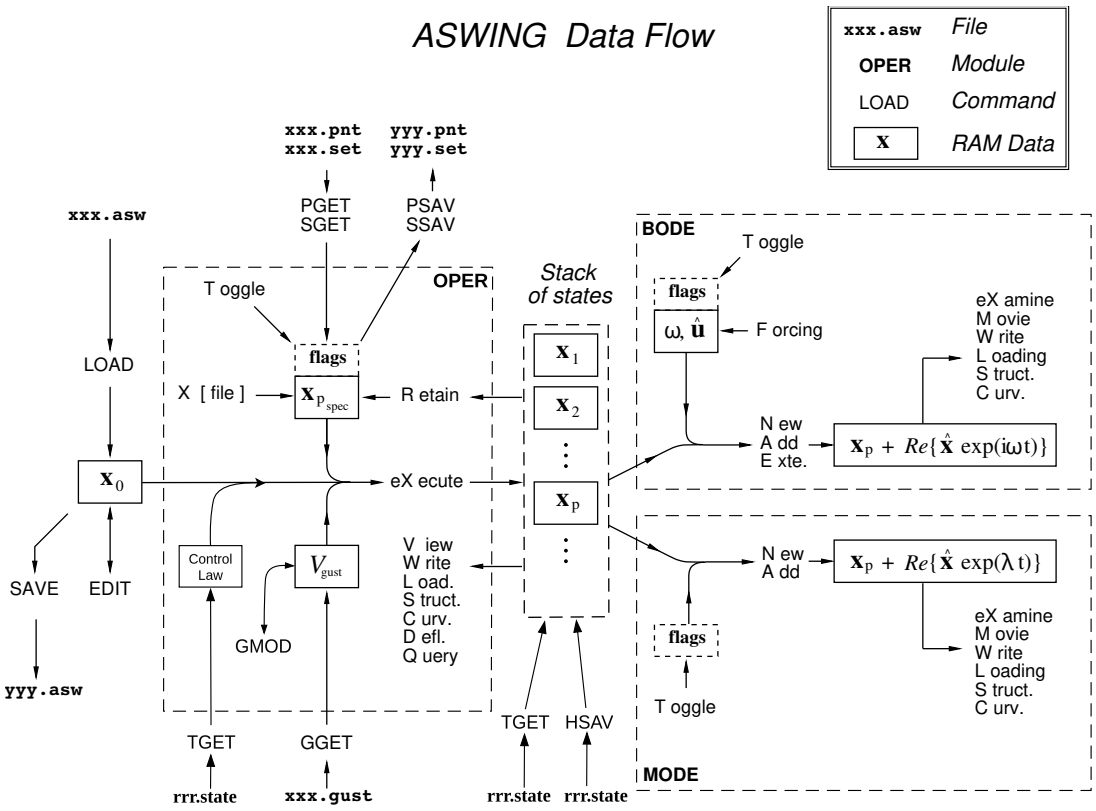


Figure 1.11: Schematic of ASWING typical pipeline [2]

A typical ASWING use session proceeds as follows:

1. ASWING is started, a `.asw` file is loaded, to specify the geometry used
2. a first trimmed state (X_0) computation is initiated. This is done by going in the `OPER` menu:
   - from `OPER`, trimmed state computations (getting the first x_0 state such that, usually, the aircraft is in leveled flight, at a certain airspeed) or time-transient computations (aircraft crossing a gust) can be computed.
   - the simulations take into account the simulation settings `.set`, the operation point specifics in `.pnt` (like required bank angle, required airspeed...), and the constraints toggled in the `T` menu of `OPER` (shown in Fig. Figure 1.5).
3. once one or more steady operation points are retained, it is possible to perform analyses in the `MODE` or `BODE` menus:
   - `MODE`: this menu is used to perform eigenmode calculations around a certain trimming point (previously computed in `OPER`), to understand what the natural frequencies and vibration modes of the aircraft are. This menu allows to plot pole locations, and spot unstable modes. It also allows the generation of a Reduced Order Model (ROM) of the aircraft. While working in frequency domain, it is possible to select one mode and simulate it in time domain, visualizing the associated eigenvector

- `BODE`: this menu is used to simulate the forced response of the aircraft to a harmonic input, whether it is a control surface or a sinusoidal gust field. Analyses are performed around a certain trimming point, previously computed in the `OPER` menu. This menu allows to generate bode plots of quantities of interest, or even simulate time-domain behavior for a specified frequency within the bode plot

4. For all menus, results can be exported either as graphics (snapshots of generated plots), or as text files, tabulating the required data.

The states, inputs and outputs created and used by ASWING are specified in subsection 1.2.4

1.3.3 Startup and Generic Controls

To start ASWING, one has to go to the `runs` folder of the ASWING installation, then open a terminal and type `../bin/aswing`. Starting ASWING in such way will open the software without preloading any file. It is also possible to open the software by typing the command `../bin/aswing xxx`, with `xxx` being the file name. Doing so will not only start ASWING, but also load (if found) the following files, if they are found in the `runs` directory:

- `xxx.asw`: defining the geometry
- `xxx.set`: settings, parameters
- `xxx.pnt`: operation point definition
- `xxx.con`: control law to be used

Gust files `xxx.gust` (defining gust phenomena to be used) and state files `xxx.state` (defining state history of a certain aircraft under certain constraints) are not read automatically at startup.

It is also possible for `XXX` to be in the form `directory_1/directory_2/filename`, or even `../directory_3/filename`. This can be useful for organization purposes.

Once ASWING is started, the menus shown in Figure 1.12 are visible. In case one is lost, typing `?` will display the available menus.

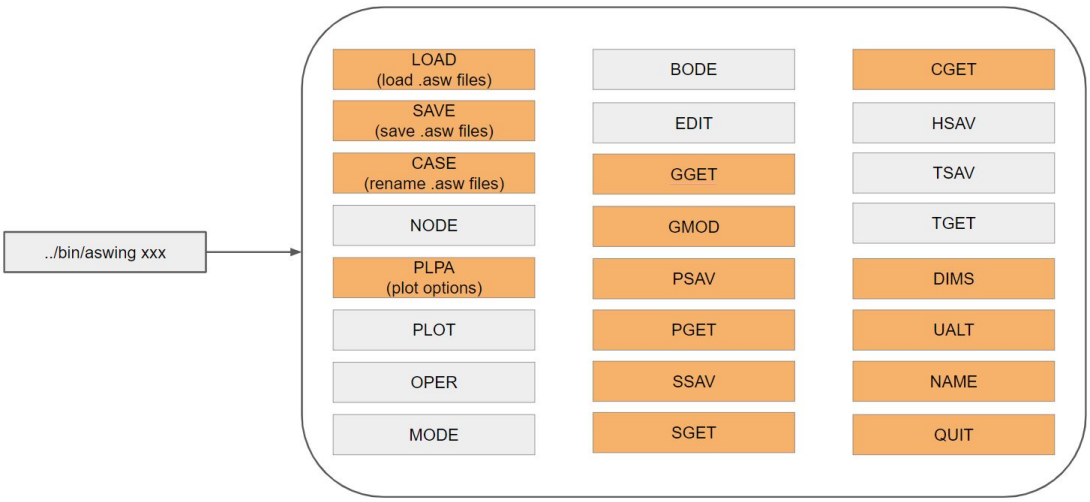


Figure 1.12: Available choices at ASWING startup. The sub-menus useful for simulation are colored in grey, while those dealing more with options/settings are labeled in orange

In general, for most menus involving plots and graphics, the following commands can be useful:

- `U:` unzoom
- `Z:` zoom (aspect ratio changes)
- `B:` zoom (aspect ratio is maintained)
- `J:` make annotations

- `H:` make hardcopy (screenshots) of current plot. Hardcopies may not be done instantly; also depending on the setting used, hardcopies may overwrite previous images. It is still possible to create hardcopies, even if the image is not visualized during ASWING execution (disabled graphics in `PLPA` menu)
- `D:` changes plot size (also available in the `PLPA` menu from subsection 1.3.4)
- `N`: rescales the axis
- `G:` show/hide the grid
- For 3D views:
  - `L`: rotate towards left
  - `R`: rotate towards right
  - `U`: rotate up
  - `D`: rotate down
  - `I/O`: ingress/outgress, hence change the perspective view
- `.` or `<`: display the next timestep results
- `,` or `>`: display the previous timestep results
- `0`: display the first timestep results
- `1`: display the last timestep results
- `Space`: exit the plot window

Note that throughout ASWING, commands written in capital letters or not work the same (h and H do the same thing).

1.3.4 Options/settings sub-menus

The sub-menus labeled as options (orange ones in Figure 1.12) are explained as follows:

- `LOAD`: loads geometry (usually .asw) files (useful if ASWING is not started with aswing xxx). When using this command, the filename must specified `with` the `.asw` suffix. The name can also have the form `../directory\_1/filename.asw` in case wanted to fetch files from various locations
- `SAVE`: saves geometry files
- `CASE`: renames current geometry file
- `PLPA`: deals with graphical and plot options (window size, plot object size, font size, outputs in color or black-and-white, append data or create a new file for each data export...). Through this menu it is also possible to set graphics to False (images are not shown during ASWING execution)
- `GGET`: loads a gust file (.gust usually)
- `GMOD`: modifies current gust parameters. The gust can either be the one previously loaded with `GGET`, or a different one, taken from the catalog of internally available ASWING gust profiles. The unit used for the gust are the one used for the `.asw` geometry. The gust profiles available are the ones provided in the `VGUST.f` script.
- `PSAV`: writes operation point file, containing the constraints and specifics set using the toggle menu. It includes for example the defined aileron deflection or the trimming point airspeed. It does not write the computed state obtained after running the simulation(s)
- `PGET`: reads operation point file, previously written with `PSAV`
- `SSAV`: writes simulation setting (usually .set) file. Those include for example the content of the `OPER>K` menu, like the maximum number of Newton iterations
- `SGET`: reads simulation setting (usually .set) file, previously written with `SSAV`

- CGET: reads control law file (.con file usually)
- HSAV: creates a .bin file (typically with .state extension) saving the states obtained after simulations done in OPER. It can save both steady (trim point state) or transient (state time history). It also saves the constraints used in OPER>T.
- TSAV: (**unknown**) generates a weird 3-line file, giving only some data from OPER
- TGET: allows to load previously computed states informations, obtained previously using OPER. Using TGET makes you avoid running again a simulation you already performed. It allows to recover not only the state (or state history) of the aircraft, but also the constraints used to compute the various operation points
- DIMS:provides the size of computation arrays, while also telling the minimum and maximum allowed size
- UALT: toggle which unit is used for aircraft altitude specification (kilometer or kilofeet)
- NAME: changes the name used for current geometry
- QUIT: leave the program (Ctrl+C also works), make sure you save your data first

1.3.5 NODE Menu

The NOTE menu allows to display the location of nodes along each beam, the position of zero-length intervals, joints, sensors, ground location, engines and point masses. It also allows visualization of minimum and maximum Fourier series used for the aerodynamic simulations along the beams of the aircraft. A typical plot obtained from the NODE menu is shown in Figure 1.13.

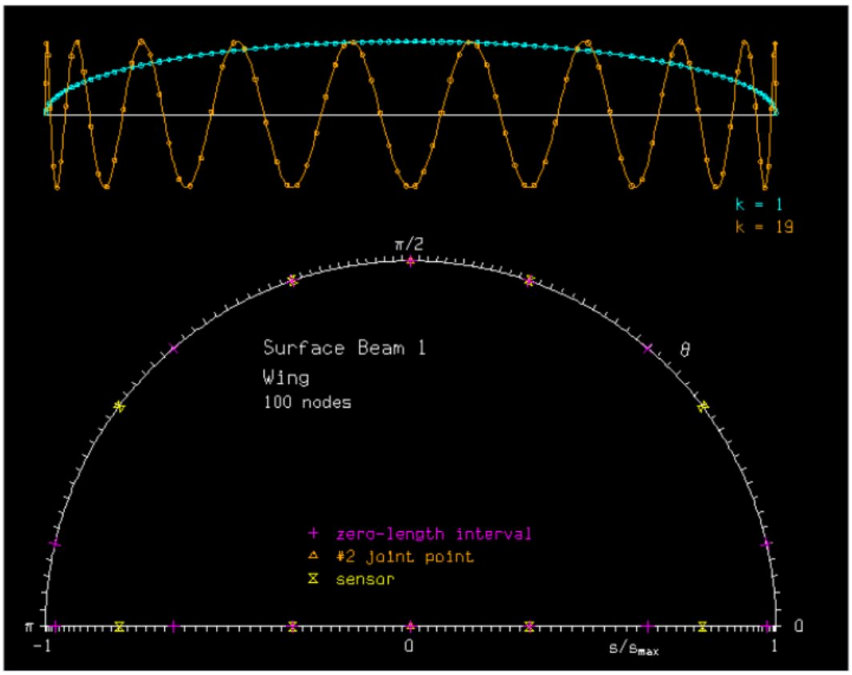
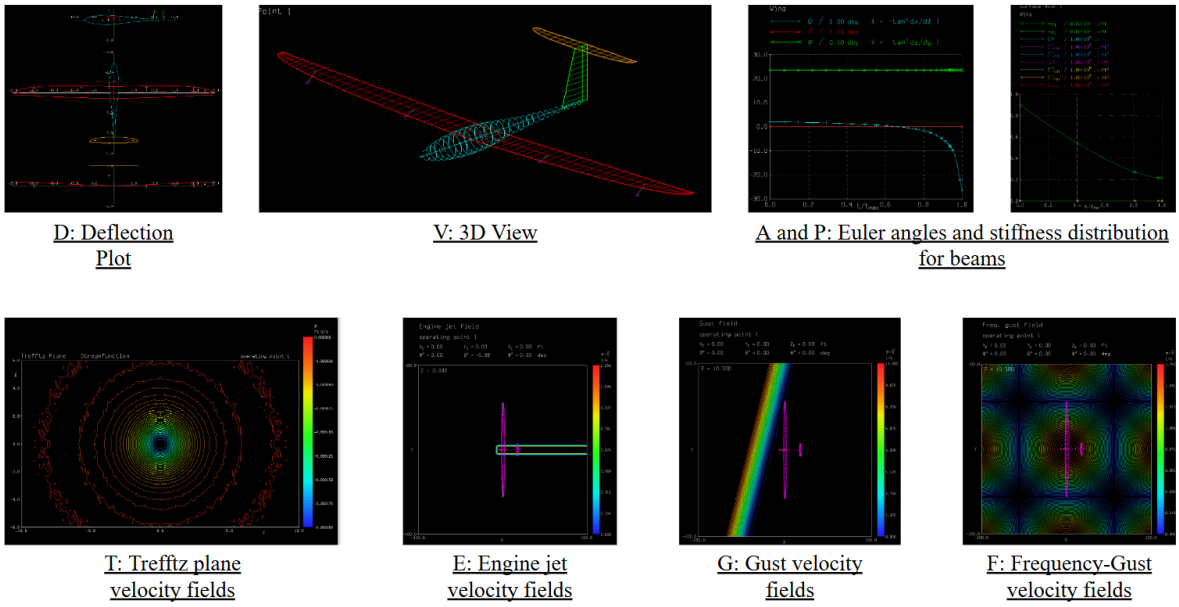


Figure 1.13: Typical plot obtained from the NODE menu. On the bottom, the interest points are plotted along the span of the beam (in this case, beam called "Wing", counting 180 nodes). On the top, minimum and maximum Fourier signals are shown (in this case, k=1 and k=19). The points location is indicated either via the ratio with-respect-to the maximum beam length, and via the angle θ (not the pitch, defined such that $s = s_{max} \cos(\theta)$)

1.3.6 PLOT Menu

The plot menu allows you to produce the imagesand access the menus shown in Figure 1.14

Plot options are also reachable with the O sub-menu, granting access to graphics settings (grids on/off, axis on/off, what is shown, settings for unsteady simulation movies...)



D: Deflection Plot | V: 3D View | A and P: Euler angles and stiffness distribution for beams

T: Trefftz plane velocity fields | E: Engine jet velocity fields | G: Gust velocity fields | F: Frequency-Gust velocity fields

Figure 1.14: Various images generated from the PLOT menu

1.3.7 OPER Menu

The OPER menu is used to find operating point performances. It includes the following sub-menus:

- Operation Point Specifics (State Variables)
  - !: modify constraint parameters (for current operating points
  - !!: change one parameter for all operating points
  - +-: add/delete operating points
  - --: delete all points except the current one
  - Q: operation point specifications
  - #: number of currently edited operation point[2]
- Operation Point Specifics (Sensor Variables)
  - @: modify constraints on sensors
- Simulation Parameters
  - . : transient or steady-state simulation?
  - N: change the load factor
  - A: change the altitude (air density), gravity
  - G: change the ground effect specifics
  - F: modify flap deflections
  - E: modify engine forces
- Simulation
  - T: toggle constraints
  - K: simulation settings
  - X: run simulation
    - xx: run on all operation points

[2] type the number, not "#"

    * `x input_file`: airplane can have state variables changing with time (programmed aileron deflection for example). In the ASWING User Manual[2] this is referred to as Open-Loop forcing. If simulation goes beyond the last specified parameter(t), the parameter is set to 0. The parameter is interpolated according to specified points written in `input_file`. Refer to the User Manual to understand the format of `input_file`.
  - `R`: retain current parameters (make the parameters found with previous operation point being fixed for next simulation)
  - `:`:set the current aircraft position as the new jig (undeformed) shape, but does not write it in the `.asw` file, but keeps such configuration in the RAM memory
  - `I`: initialize the solution to the initial state from the `.asw` file

- Operation Points Reporting
  - `=`:list operation points solutions
  - `M`: display force derivatives matrices
  - `W`: writes operation points specifics to a file. It does not save the state

Within the `OPER` menu, once the solution is computed, it is also possible to generate plots:

- `L`: loading plot
- `S`: structural plot
- `D`: deflection plot
- `C`: curvature-strain plot
- `V`: 3D view (steady simulation) or 3D Movie (unsteady simulation)
- `O`: plot options

Among the `OPER` options available from `K`, normally saved in the `.set` file, there are the following:

- `I`: maximum Newton iterations
- `S`: stride (20), meaning only plot every 20th point when doing transient simulations
- `F`: fast/slow mode. For fast, the vortex influence matrices are kept to the undeformed case (moving wake)
- `L`: include or not unsteady effects. Useful to evaluate the importance of flow unsteadiness on the solution
- `N`: node plotting or not
- `T`: terse listing, meaning not listing all parameters on the terminal when typing the `=` command, but only focusing on the aircraft-average parameters. The spanwise parameters are not shown
- `V`: what variables must be reported on the terminal once the computation is run?
- `D`: adding offset to the reported parameters on the terminal, for more compact view
- `P`: what parameters to show while computing the solution on the terminal
- `B`: what beams to show while computing the solution on the terminal
- `G`: plot ground reaction loads (in theory 0 for free flight)
- `E`: shows better engine specs, higher detail
- `R`: setting the reference location. $(0, 0, 0)^T$ or another reference?
- `A`: use the Earth reference frame to do calculations
- `C`: stability and control derivatives reference frame

  - 0: body axes
  - 1: stability axes (x fwd, z down)
  - -1: rev. stability axes (x aft, z up)
  - 2: wind axes: x windward, z up

- M: defines the Mach number, either by specifying it, or by M=V/a
- W: vortex core radius: value of the vortex core/local chord ratio. If a smaller w is obtained from node spacing, that is used. The default W is 0.25. **Note that changing this number does not improve the convergence of Newton iterations if they are diverging due to an excessively small timestep.**
- Y: involves ROM generated in the MODE menu
- K: selecting the inertial data measured by aircraft sensors: absolute (with-respect-to Earth frame) or relative (with-respect-to Body frame)

1.3.8 BODE Menu

The BODE menu includes the following sub-menus:

- Performing Forced Oscillations Simulations
  - T: toggle constraints menu
  - =: list operation points solutions (states)
  - #: set which operation point to use[3]
  - Q: what are the specifics of the various operation points?
  - N: new frequency-range analysis (i.e. from 0.1 to 10 hz forcing what happens)
  - E: compute more points, outside current range (from 10 to 20Hz for example)
  - A: add more points in current reviewed range (make finer frequency discretization)
- Defining the Forcing Parameter
  - F: select which forcing parameter is used (more than one can be selected, but the cross influence of the input parameters will not be evaluated, example flap1, flap2). Gust forcing parameters can also be present if those have been enabled in the I menu
  - I: gust forcing parameters (instead of flap for example, accounting for spanwise and flow-wise gust frequencies). Enabling and modifying this setting will make gust forcing parameters appear in the F menu
  - R: response parameters (again, more than 1 can be selected, for example lift and pitch)
- Outputs Available once a Frequency Analysis is Finished
  - G: gain plot generation
  - P: phase plot generation
  - D: allows to overlay bode plot data locally stored using W to current bode plots
- Option Sub-menu
  - !: selecting if one wants the bode plots of Response/Forcing or 1-Response/Forcing
  - $: not using the actual laws of motion of the aircraft, but a Reduced Order Model, created in the Mode menu
  - O: options (more or less the same as in the plot menu)
  - K: simulation settings (similar to the OPER menu)
- Time Domain Simulation for a Specified Frequency

[3] type the number, not "|"

  - X: examine a specific frequency response in time domain
  - V or M: viewing 3D view
  - L: loading response plots (loaded by default after executing X)
  - S: structural response plots ($\mathbf{F_c}$, $\mathbf{F_s}$, $\mathbf{F_n}$, $\mathbf{M'_c}$, $\mathbf{M'_s}$, $\mathbf{M'_n}$). The difference between $\mathbf{M_i}$ and $\mathbf{M'_i}$ is explained in the ASWING Technical Description, [2]
  - C: curvature-strain response plot

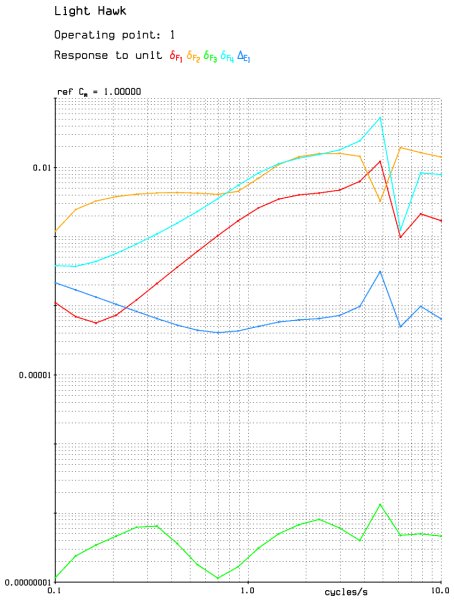
- Export Data
  - W: write the bode plot data, for each response parameter specified in R (Lift magnitude and phase wrt unit frequency for example)

Concerning bode analysis that include different inputs (say δ_{F_1} and δ_{F_2}) and several outputs (say L and θ), the analysis only manages to provide transfer function plots for the following values, around equilibrium position previously computed in OPER:

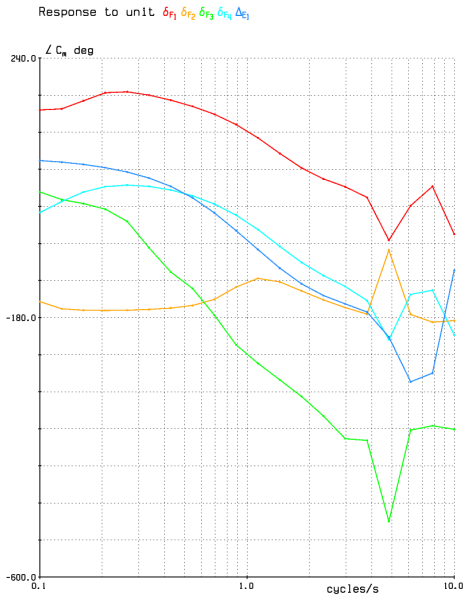
- $L(\delta_{F_1})$
- $L(\delta_{F_2})$
- $\Theta(\delta_{F_1})$
- $\Theta(\delta_{F_2})$

Note that the transfer functions assume unit harmonic inputs.

Examples of bode plots generated within the BODE menu are provided in Figure 1.15, while time-domain plots from single-frequency analysis within the BODE menu are provided in Figure 1.16



(a) C_m transfer function Magnitude bode plots

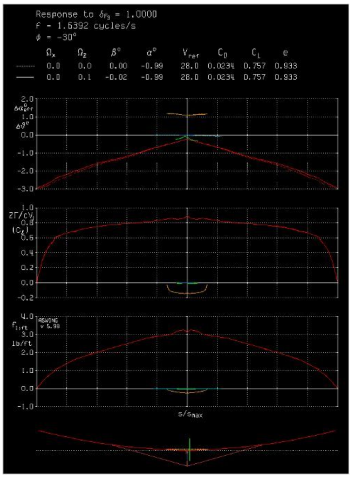


(b) C_m transfer function Phase bode plots

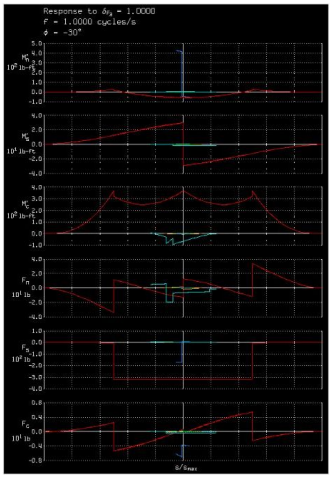
Figure 1.15: Examples of Bode plots of C_m over various different inputs like control surface $\delta_{F_1}, ..., \delta_{F_4}$ or engine power Δ_{E_1}. The plots were obtained from the BODE menu

1.3.9 MODE Menu

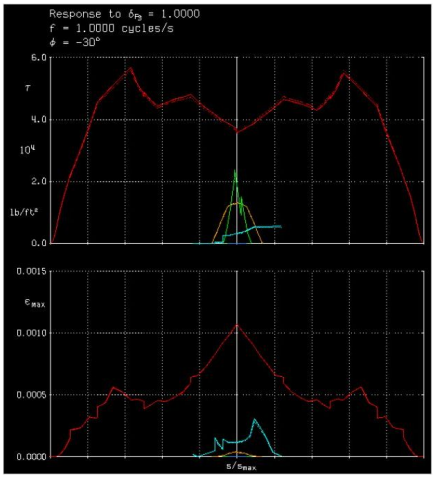
The MODE menu is used to compute eigenmodes (free oscillations) of the aircraft around a trimming point, previously computed with OPER (and/or saved and reloaded using HSAV and TGET).



(a) Response plots (obtained with L)



(b) Structural response plots (obtained with S)



(c) curvature-strain response plot (obtained with C)

Figure 1.16: Plots generated from time simulation corresponding to single-frequency analysis (BODE menu)

The idea of MODE is to compute a certain number of eigenvalues, around a certain location. For instance, in case one were to look for flutter modes, the idea would be to start looking for eigenvalues around $s = 0 + \omega_{f0} i$, with ω_{f0} being an estimate of the flutter angular frequency.

A typical root-locus plot of a rigid aircraft is displayed in Figure 1.17.

Sigma [1/s]	Omega [rad/s]	Mode
3.18E-02	0.00	Spiral
-1.84e-02	3.64e-01	Long-Period (Phugoid)
-7.26e-01	2.08	Dutch-Roll
-11.3	0	Pitch +Up/Down
-9.01	6.84	Short-Period
-15.6	9.18	Roll+Yaw
-22.3	0	Pitch +Up/Down
-23.8	0	Roll mode

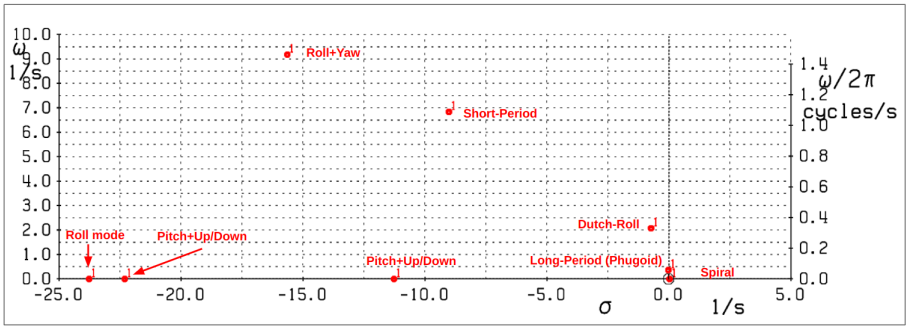


Figure 1.17: Roots and correspondent modes of a rigid aircraft, called Hawk (the aircraft is available in the free documentation of ASWING [2]), around a trimming point of 80 ft/s, leveled flight. More explanations on what the different modes correspond to are available in Phillips [9]

Modal analyses performed for the same aircraft, made flexible, will result in the following particularities:

- The trimming point may change because of airframe flexibility impact on the aerodynamic surfaces
- Flexibility will change and/or modify rigid-aircraft modes
- Flexible modes due to airframe bending will appear

An example of a root-locus plot for the same aircraft as in Figure 1.17, at the same trimming point (leveled flight, 80 ft/s) but flexible is displayed in Figure 1.18. Note the different impact of flexibility on each mode, as well as the appearance of airframe deformation modes.

To identify which pole corresponds to which mode, viewing the mode behavior in time domain is essential. Screenshots of eigenmode views and videos are visible from menus V and M are visible in Figure 1.19

While doing analyses in the MODE menu for the case of a rigid Hawk aircraft, the appearance of additional modes on the real axis, as shown in Figure 1.20 was noticed. Investigation on that point are needed, since, from the ASWING manual ([2]), only 8 modes should be present for a rigid aircraft:

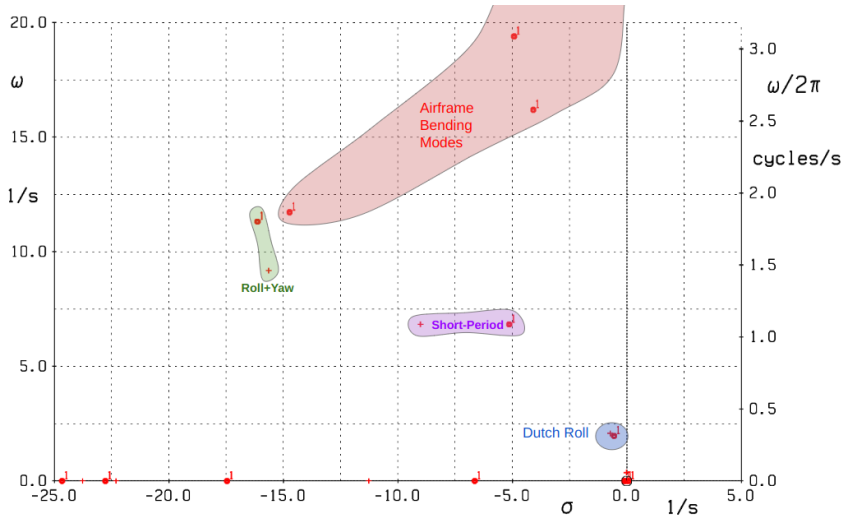
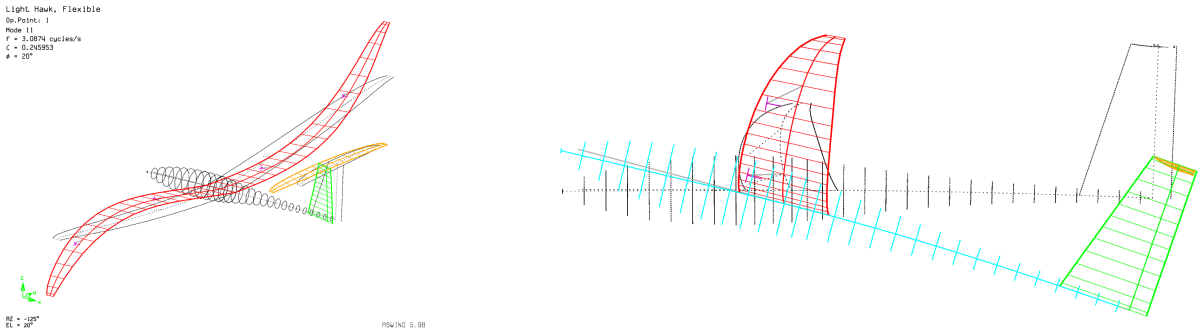


Figure 1.18: Root-Locus plot of the modes of the flexible Hawk aircraft (round dots) compared to the poles of the Rigid Hawk aircraft (previously shown in Figure 1.17



(a) Flexible Hawk airframe flexibility mode snapshot, of frequency of 3.09 Hz

(b) Short-Period mode snapshot for the flexible hawk aircraft, seen from the side

Figure 1.19: Time domain snapshots of some flexible Hawk aircraft modes

from the ASWING manual [2] *If the aircraft is perfectly rigid, at most 12 eigenvalues/eigenmodes will exist. These correspond to the 12 degrees of freedom present (6 rigid-body motions, and their rates), and 4 of these eigenvalues will be zero (X,Y,Z,heading rigid-body modes have no effect on dynamics). It is important not to request more eigenvalues than are present, else the ARPACK Arnoldi algorithm will fail.*

Sigma [1/s]	Mode
-27.7	Spiral and pitch
-34.6	Pitch+Up/Down
-34.8	Pitch+Up/Down
-37.7	Roll
-40.9	Pitch+Up/Down
-43.4	Roll
-46.5	Pitch+Up/Down
-49.4	Spiral and Pitch
-52.3	Pitch+Up/Down
-53.0	Spiral+Pitch

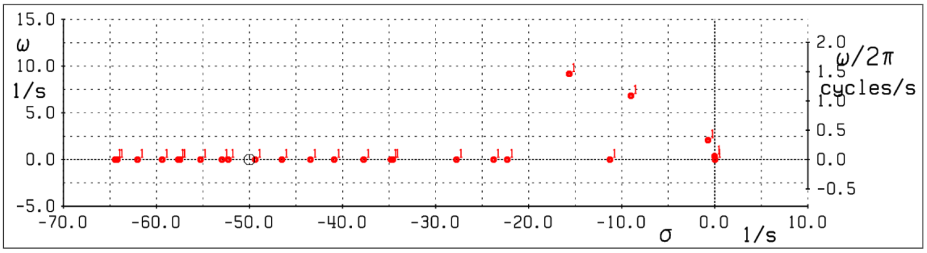


Figure 1.20: Eigenmodes on the real axis of the rigid Hawk. Both images were obtained from the MODE>X menu

The MODE menu includes the following sub-menus:

- Operating Point Creation and Selection
  - T: toggle constraints. Note that the command ! to change the values of constraint values can be used

  - `=`: list point solution: what are the states of the steady-state solution we are starting from
  - `#`: point number select. If several operation points are available, select which operation point we are using to do the mode calculation around
  - `Q`: query point specs: very simplified view of the operation point, not including all the states

- Eigenmode Computation
  - `N`: new eigenvalue calculation. Will ask how many eigenvalues per operation-point. If rigid, maximum 12 (6DOF+their rates, 4 of them are 0). `N` to compute only for current op-n point, `NN` to compute for all op-points.
  - `A`: additional eigenvalues (allows to add eigenvalues calculations to current computation)
  - `F`: free-body (vacuum) toggle IDK what it does
  - `R`: root locus plot

- Eigenmodes Output
  - `P`: plot-zone radius (more or less a zoom on a certain region of the previously viewed root-locus plot). The zoom method specifies the center of a zoom cercle we are interested in (σ_0 and ω_0) and the research radius we want to check out R_0
  - `-`: list stored eigenvalues (self explanatory, will display every eigenvalue number, sigma and omega in a table)

- S-Domain Output
  - `W`: write eigenvalues to a text file (useful to build better plots with Matlab for example)
  - `D`: data file overlay toggle
  - `G`: generate Reduced Order Model (will first require you to select the eigenmodes you want to include in the study)
  - `Y`: output eigenvector (outputs one eigenvector describing how the geometry varies)

- Time Analysis (For a single Eigenmode)
  - `X`: examine (time-domain) an eigenmode. To do so, ASWING will ask the user to click on the root-locus plot on the root to be analyzed. If the user clicks outside the graph region, but within the figure, ASWING will ask the user to provide the approximate σ and ω of the root, after which ASWING will find the closest root and show the time-transient behavior of the aircraft eigenmode (linearized aircraft behavior)
  - `V`: 3D-view of aircraft eigenmode
  - `M`: 3D-movie (real-time view from t=0 of the aircraft eigenmode oscillation, according to a previously set view on V)

- Time Analysis Outputs (as in `BODE`)
  - `L`: loading eigenmode plot
  - `S`: structural response plots ($\mathbf{F_c}$, $\mathbf{F_s}$, $\mathbf{F_n}$, $\mathbf{M'_c}$, $\mathbf{M'_s}$, $\mathbf{M'_n}$)
  - `C`: curvature-strain eigenmode plot

- Options
  - `F`: free-body (vacuum) toggle
  - `O`: options for plots
  - `K`: control toggles, settings
  - `E`: eigenmode output select
  - `I`: impose gust-forcing modes

1.3.10 EDIT Menu

The `EDIT` menu allows to change the structural spanwise performance of the beams composing the aircraft. The following commands are accessible:

- Selecting the Correct Variable
  - `L`: list the available variables one can change
  - `B#`: selecting the beam number #[4]
  - `#`: selecting variable number #[5]
- Available Modifications
  - `S`: scale by a constant
  - `A`: add a constant
  - `M`: modify using mouse cursor and current plot (click to modify the plot shape)
  - `I`: inserting new points on the curve (the property curve is built by interpolation of the values at those points, not by the points created with `M`)
  - `D`: delete curve points
  - `C`: create a corner in the curve (in theory, the first derivative is continuous at the point locations, this command allows discontinuities)
  - `R`: refresh the curve interpolation, by accounting of both the interpolation points, and the desired values, specified with `M`
  - `O`: overlay (show) additional data, from an external file, previously written with the `W` command, to the current plot
- Other
  - `T`: tangent endpoint (idk what it is)
  - `Y`: is the beam properties symmetric or not
  - `E`: Epsilon/Tau plot toggle
- Output
  - `W`: write current points to a text file

An example of the plots displaying the structural property value throughout the selected beam is shown in Figure 1.21

[4] type the number, not "#"
[5] type the number, not "#"

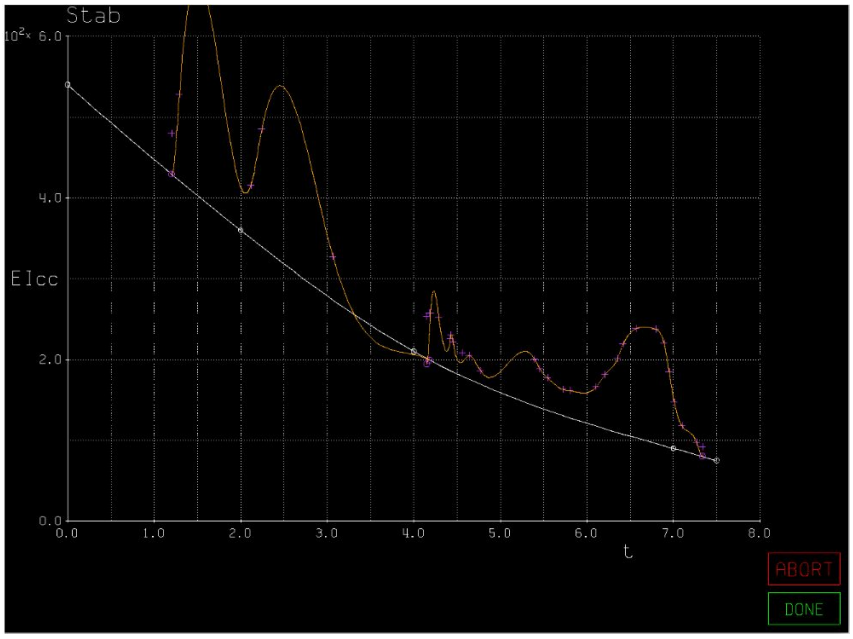


Figure 1.21: Plot of the EI_{cc} stiffness of the Stab beam. The initial stiffness profile is shown in white, while the plotted required profile is shown in orange. The latter was created with the `M` function by clicking and creating the purple points. This profile will serve as guide for the new white distribution (the one that will be followed by the software) that will depend on the positioning and value of the white points

Chapter 2

Official MIT Documentation

This paragraph reproduces the official ASWING documentation from MIT [2], with minor formatting changes, added comments, and a revised layout.

2.1 General Description

2.1.1 Introduction

```
ASWING 5.97  User Guide                          last update  20 July 12

Mark Drela
MIT Aero & Astro


ASWING is a program for the aerodynamic, structural, and control-response
analysis of aircraft with flexible wings and fuselages of high to moderate
aspect ratio.  Static, time-domain dynamic, and frequency-domain dynamic
analyses can be +textbfperformed+, allowing predictions of divergence, control
reversal, flutter, trajectory perturbations, structural strains and
stresses, structural load resultants, etc.

The structural model consists of nonuniform nonlinear connected beams,
and allows arbitrarily large deflections, with arbitrary loads.  The
aerodynamic formulation is a lifting-line model, with wind-aligned
trailing vorticity discretized by a vortex-lattice type approach
(effectively an enhanced Weissinger method).  Volume effects of slender
bodies are modeled with source+doublet lines.  Compressibility
is treated via the Prandtl-Glauert transformation in wind-aligned axes.
Arbitrary gust velocity fields can be imposed.

The aerodynamic and structural formulations are fully coupled, with
the aero loads causing structural deflections, and the structural
deflections and twists affecting the aero loads.  The overall coupled
formulation is solved by a full Newton method.  Eigenmode analyses
```

```
are performed using the complete Jacobian matrix from the Newton
solver.  The theory is described in the separate PostScript document,
and will not be expounded here.
```

2.1.2 Versions, Publications and Install notes

```
ASWING 5.x is a major upgrade from the 4.x series, which was a major
upgrade from the ASWING 3.x series (software always expands).  The
applicability of each series is:

3.x  Single surface, nonlinear in z but not in x (small sweep angles only),
     small yaw angles, incompressible flow

4.x  Single surface, fully nonlinear, compressibility corrections,
     flow-aligned VL formulation (large yaw angles OK), yaw and roll
     effects only

5.x  Same as 4,x, PLUS:  Multiple surface/fuselage configuration,
     roll, yaw, pitch rate effects all included in complete dynamical
     representation, structural and control response matrix calculations,
     eigenmode analyses, on-screen flight "movies", improved graphics
     in general, and a bunch of other goodies.

The rather lame 3.x formulation is described in the following reference.

 Drela, M.
 "Method for Simultaneous Wing Aerodynamic and Structural Load Prediction",
 Journal of Aircraft, 27(8), Aug 1990.  (AIAA Paper 89-2166).

The most recent publication appears in the April'99 SDM Conference:

 Drela, M.
 "Integrated Simulation Model for Preliminary Aerodynamic,
        Structural, and Control-Law Design of Aircraft",
 AIAA Paper 99-1394
 AIAA 40th SDM Conference, St Louis, April 1999.

The latest formulation is described in the PostScript files

 tex/asw.ps       basic formulation
 tex/aswu.ps      unsteady extension
 tex/dataflow.ps  visual layout of file and RAM data flow paths

The FORTRAN implementation closely follows the equations in the
asw,aswu documents, so they should be consulted if source code changes
are being considered.
```

The eigenmode analysis module of ASWING (.MODE menu) employs the ARPACK sparse eigenpackage, developed by Sorensen and co-workers at Rice U. This package must be obtained if it is necessary to perform eigenmode analyses in ASWING --- aircraft stability modes, flutter, etc.

Alternatively, dummy ARPACK routines which are included here can be used for linking if eigenmode analyses are *not* required. Using the dummy routines will only disable the .MODE menu. The frequency-response analyses (.BODE), and static and time-domain simulation analyses (.OPER) will remain fully functional.

The ARPACK library, with literature and documentation, is available
by anonymous ftp from: ftp.caam.rice.edu
or via the WWWeb from: ftp://ftp.caam.rice.edu/pub/software/ARPACK

2.1.3 Input/Output files

ASWING uses the following Input/Output files.

file	type	origin	contents
xxx.asw	I/O	user,ASWING	Configuration definition
xxx.con	I	user	Control-Law data
xxx.pnt	I/O	ASWING	Oper. Point definitions
xxx.set	I/O	ASWING	Settings, Parameters

All these files will be read automatically if ASWING is started
with an argument:

```
% aswing xxx
```

Other input files are read in via runtime keyboard commands.

ASWING also creates a number of miscellaneous listing files on demand. But these are output-only, and are not important for program execution.

The file names above are the defaults, with "xxx" being some arbitrary descriptive name common to all of them (e.g. glider.asw, glider.con, etc). Although arbitrary filenames can be used, the default forms considerably simplify ASWING execution, and so should be used if possible.

Only the input file xxx.asw is required for running ASWING, and initially must be provided by the user. Once this file is generated, typically via a text editor, its contents can be interactively

```
modified in ASWING, and the resulting new configuration can be
written out as a new yyy.asw file.
```

2.2 Code-Related Information

2.2.1 Changing flap,engine dimension limits

```
The maximum number of flaps and engines is set by the NFLPX and NENGX
dimensioning parameters.  To change these limits, perform the following:

 In file DIMEN.INC :  change  NFLPX, NENGX  dimension parameters.

 In file INDEXP.INC:  add/remove KPFLP* and KPENG* indices to match NFLPX,NENGX
                      add/remove IPFLP* and IPENG* indices to match NFLPX,NENGX
                      increase/decrease KPTOT,IPTOT to match new list sizes

In addition, if NFLPX was changed, perform the following:

 In file INDEXB.INC:  add/remove DATA statement lines for dCLdF,dCMdF...
                      increase/decrease JBTOT to match new number of data lines

 In file DIMEN.INC :  increase/decrease JBX to be at least as big as JBTOT

 In file iosubs.f  :  set parameter JDIM to be the same as JBTOT (2 places)


Important!!!
The KPFLP1,KPFLP2... KPENG1,KPENG2...  parameters and data quantities must
appear contiguously.  DO NOT simply add additional parameters to the end of
the list.
```

2.2.2 Considerations for Data Discontinuities

```
A slope break or discontinuity in the input data, specified via
a doubled-point as described previously,  will normally be "smeared"
when it is interpolated onto the computational node distribution.
The exception is for variables which have KBREAK=1 declared in the
INDEXB.INC variable-declaration file.  If any of these variables
has a doubled input point (i.e. slope break or discontinuity),
then a zero-length interval will be set up in the computational
node distribution as well, so that the discontinuity will be
captured perfectly rather than smeared.  These break-enabling
variables are currently the following:
```

```
      x
      y
      z
    twist

    Dmgcc
    Dmgnn
     Dmg
    DCcg
    DNcg

    chord
    alpha

   dCLdF1
   dCMdF1
   dCLdF2
   dCMdF2
     .
     .
```

Any other variable can have KBREAK=1 declared if appropriate.
It must be mentioned that for these variables one should avoid
declaring doubled points for no reason, since the resulting
zero-length computational interval will "waste" one grid point.
In the ideal situation, many such variables have doubled points
at common t locations, as in the following example.

```
   t      y    chord    dCLdF1   dCMdF1
 0.0    0.0    1.0       0.0      0.0
 7.0    7.0    1.0       0.0      0.0
 7.0    7.0    1.0       0.05    -0.02
10.0   10.0    0.6       0.05    -0.02
```

All four variables y, chord, dCLdF1, dCMdF1 have a doubled point exactly
at t = 7.0, and so their discontinuities are all captured on the same
zero-length computational interval. By the same token, one should avoid
doubled points which are "almost" co-located, as in the following case.

```
   t      y    chord
 0.0    0.0    1.0
 7.0    7.0    1.0
 7.0    7.0    1.0
10.0   10.0    0.6
```

```
   t    dCLdF1   dCMdF1
  0.0    0.0      0.0
  7.01   0.0      0.0
  7.01   0.05    -0.02
 10.0    0.05    -0.02
```

This will give zero-length intervals at t = 7.0 and also at t = 7.01 .
This is not just a waste of points, but will also likely produce
rather irregular computational node spacings. The NODE menu allows
display of the zero-length interval locations.

Variables which do not enable slope breaks (those not listed above)
are assumed to be continuous. This means that function discontinuities
should be avoided for these variables, since the spline evaluation at
a function discontinuity is then uncertain (both nodes at the zero-length
interval can take on the value on either side of the discontinuity
in an unpredictable way influenced by machine roundoff. Discontinuities
in these variables are detected and a warning is issued. e.g.

```
 * Discontinuity in      mg     on beam           4
```

Such discontinuities should be removed, or KBREAK = 1 can be set for
that variable in INDEXB.INC. Note that this pertains only to function
discontinuities. Slope discontinuities are inconsequential.

2.2.3 Warning and error messages

ASWING puts out info messages in three levels of severity, preceded
by "*", "**", and "***", respectively.

```
   * Warning.  Unexpected results may be produced.  May also be benign.
  ** Severe warning.  Program may crash if solution is attempted.
 *** Error.  Program crash almost inevitable if solution is attempted.
```

Sometimes you might see a cryptic message preceded by a "?".
For example:

```
   ? IISET:  IITOT, IIX:'     205     201
```

This indicates an internal program error (bug). If you see this,
please send a nastygram to drela@mit.edu , and I'll try to track
it down if time permits. Sending the xxx.asw, xxx.set, xxx.pnt,
files will help immensely.

2.2.4 Graphics

ASWING uses the Xplot11 graphics package, which drives X terminals
and can generate PostScript files on demand. The screen graphics
are most effective in reverse-video, which is the default.
Normal video can be selected by setting the shell variable

```
 % setenv XPLOT11_BACKGROUND white
```

before running ASWING. The default reverse video can be restored with

```
 % unsetenv XPLOT11_BACKGROUND
```

This shell variable has no effect on PostScript output.

ASWING uses color X-Windows graphics, with color PostScript
as the default. These defaults can be changed by setting
the IDEV and IDEVRP variables appropriately in
SUBROUTINE INIT (in init.f). Color Postscript can also
be toggled on/off in several of the submenus.

2.3 Geometry File Description

2.3.1 Global and Beam Blocks

A 4.x input file will need only a few minor changes to be usable with
ASWING 5.x.

The format of xxx.asw consists of a number of definition "blocks" which
come in two species:

i) Global information blocks for specifying units, parameters,
    beam joints, point masses, etc.

ii) Beam definition blocks, which contain a number of distribution
    sub-blocks which define beam properties along each beam’s axis.

A beam can be a lifting "surface" (chord distribution IS specified)
or a non-lifting "fuselage" (chord distribution IS NOT specified).

2.3.2 Global Information Blocks

Types of Global Information Blocks

```
The following types of global information blocks are recognized,
each beginning with one of the keywords

 "Name"        (required)
 "Unit"        (required)
 "Constant"    (required)
 "Reference"   (1st line is required)
 "Weight"      (optional)
 "Sensor"      (optional)
 "Engine"      (optional)
 "Strut"       (optional)
 "Joint"       (optional)
 "Jangle"      (optional)
 "Ground"      (required)

and each ending with "End".  Blank lines are ignored.  All characters
after and including a "!" are ignored, to allow in-line comments.
A comment-only line can be placed anywhere, and can begin with
"#" or "%", in addition to "!".

A typical global information block is shown below.

Weight
# Nbeam  t    Xo   Yo   Zo   Weight CDA  Vol   Hxo  Hyo  Hzo  Ixx Iyy
Izz Ixy Ixz Iyz  ! comment line
+        0.   0.   1.5  0.    100.  0.   0.    0.   0.   0.   ! adder line
*        1.   1.   1.   1.   1000.  1.   1.    1.   1.   1.   ! multiplier line
   1    12.0 -2.5  0.0 -1.5  2.50   0.0  0.01  250. 0.0  10.0 ! data line
   3    34.0  0.5  0.0  0.0  0.50   0.1  0.02  0.0  0.0  0.0  !  "    "
   4    -2.0  2.0  0.0  0.5  8.20   0.0  0.0   0.0  0.0  0.0  !  "    "
End

A multiplier line beginning with "*" can be placed anywhere in the data,
and gives scaling factors which are immediately applied to all data below
(no multiplier is specified for the integer beam index, however).  Another
multiplier line can be placed again to change these factors.  If such
multiplier lines are absent, the factors default to unity.

 NOTE: It does not seem that the multiplier factors are applied both, but only one

An adder line beginning with "+" can likewise be placed in the data,
and gives constants which are added AFTER the multiplier is applied.
If such adder line is absent, the constants default to zero.  The
```

final data value is:

```
F  =  F_input * multiplier  +  adder
```

The various quantities specified in the global information blocks are assumed to have the following user-specified units (described shortly):

Variable	Unit	
Area	L^2	reference area
Chord	L	reference chord
Span	L	reference span
X,Y,Zref	L	reference location
t	-	parameter along beam, in arbitrary units
Xo,Yo,Zo	L	spatial location in body axes
Weight	F	point mass * g
CDA	L^2	drag area
Vol	L^3	volume
Hxo,Hyo,Hzo	FL^2/T	angular momentum * g
Ixx,Iyy ...	FL^2	moments of inertia * g
F/Peng	F	propulsive force per unit power setting
M/Peng	FL	propulsive moment per unit power setting
Txyz	-	thrust axis vector
Rdisk	L	propulsor disk radius
Omega	1/T	prop rotation rate
cdA	L^2	prop profile drag area
dLo	L	added slack length
EAw	F	extensional stiffness (= load/strain)

The format of each type of global information block is shown below:

Example Geometry: Global Information Blocks Only

```
Name
FLEXY AIRPLANE   Model 1A
End

Unit
L 1.0 ft
T 1.0 s
F 1.0 lb
End

Constant
#  g      rhoSL     VsoSL
  32.18  0.00238   1115.0
```

```
End

Reference
# Area   Chord  Span
 1200.0  22.0   180.0
#
# Xmom  Ymom  Zmom   (reference point for moments)
  0.20  0.0  -0.05
#
# Xacc  Yacc  Zacc   (reference point for accelerations)
  0.0   0.0   0.0
#
# Xvel  Yvel  Zvel   (reference point for velocities, flow angles)
 -2.50  0.0   0.10
End

Weight
# Nbeam  t    Xo   Yo   Zo    Weight   CDA   Vol    Hxo    Hyo    Hzo
+        0.   0.   1.5  0.     100.    0.    0.     0.     0.     0.
*        1.   1.   1.   1.    1000.    1.    1.     1.     1.     1.
   1    12.0 -2.5  0.0 -1.5   2.50     0.0   0.01   250.0  0.0    10.0
   3    34.0  0.5  0.0  0.0   0.50     0.1   0.02   0.0    0.0    0.0
End

Sensor
# Ksens  Nbeam  t    Xo   Yo   Zo    Vx   Vy   Vz     Ax   Ay   Az
+               0.    0.   0.   0.    0.   0.   0.     0.   0.   0.
*               1.    1.   1.   1.    1.   1.   1.     1.   1.   1.
   1      1     10.0 -1.0  8.0 -0.5   1.0  0.   0.     0.   0.   1.0
   2      1    -10.0 -1.0 -8.0 -0.5   1.0  0.   0.     0.   0.   1.0
End

Engine
# Keng IEtyp Nbeam  t    Xo   Yo   Zo    Tx  Ty  Tz    dFdPe  dMdPe
Rdisk Omega   cdA    cl     CLa    S0     C0     S1     C1     S2
C2     S3     C3
*                  1.   1.   1.   1.    1.  1.  1.     1.    -0.1
1.    100.  0.01    1.     1.     1.     1.     1.     1.     1.
1.     1.     1.
    1   0     1    12.0 -2.5  8.0 -1.5  -1.0 0.0 0.02  10.0    0.5
    0.4   4.5   0.05
    2   1     1    12.0 -2.5  8.0 -1.5  -1.0 0.0 0.02   2.0    0.1
    0.4   4.5   0.05
    1   0     1   -12.0 -2.5 -8.0 -1.5  -1.0 0.0 0.02  10.0    0.5
    0.4   4.5   0.05
    2   1     1   -12.0 -2.5 -8.0 -1.5  -1.0 0.0 0.02  -2.0   -0.1
    0.4   4.5   0.05
```

```
    3   2     4   -12.0 -5.0  0.0  0.0  -1.0 0.0 0.02  20.0    1.0
    0.8   2.5   0.20    0.5    6.0    0.3054 0.3486 0.1552 0.2073
    0.0923 0.1379 0.0614 0.0991
End

Strut
# Nbeam  t      Xo    Yo    Zo     Xw    Yw    Zw     dLo   EAw
*        1.     1.    1.    1.     1.    1.     1.    1.    0.5
   2   -18.0    0.0 -10.0  -0.25   0.2   0.0   -6.0   0.2   2.50E+06
   2    18.0    0.0  10.0  -0.25   0.2   0.0   -6.0   0.2   2.50E+06
End

Joint
# Nbeam1  Nbeam2    t1     t2    [ KJtype ]
   1       2        0.0    5.0       0
   1       3       10.0    0.0       1
   1       3        2.0    5.0       3
End

Jangle
# Njoint  hx   hy   hz
  2       1.0  0.2  0.
#
#  Momh    Angh
*  1.0e5   1.0
+  0.      0.
  -15.0  -15.0
  -10.0   -5.0
   -5.0   -1.0
    0.0    0.0
    5.0    1.0
   10.0    5.0
   15.0   15.0
End

Ground
# Nbeam   t     KGtype
   1      0.0     0
End
```

"Name" block (optional)

```
This has only one significant data line, giving the name of the
geometry in 80 characters or less.  If more than one data line is
present, the last line before the "End" keyword will be taken
as the input, and the previous lines will be ignored.
```

"Unit" block (required)

ASWING makes no assumptions about the units it uses, so that any set of consistent units can be chosen for the input, and a consistent (possibly different) set of units can be chosen for the output. The Units block simultaneously specifies the units used for the input file,
and also allows selection of the output units. It must have three or four lines, each beginning with one of the four capital letters L T F M,
denoting Length, Time, Force, and Mass. The number and text string following each letter is the magnitude and name of that unit used in the entire input file. If four consistent units are specified, they must satisfy the identity FT^2 = ML. Since one unit is "redundant", it is simplest to specify only three units, and the missing one will be derived.

All output will be given in the unit names specified in the Units block.
Hence, another set of output units can be easily chosen by substituting
the appropriate magnitude and name for any unit. The following three Unit Blocks are equivalent for an English-unit input file

```
L 1.0   ft          L 0.3048 m          L 30.48   cm
T 1.0   s           T 1.0    s          T 1.0     s
F 1.0   lb          F 4.450  N          F 445000. dyn
```

but will produce English, Metric SI, and Metric CKS output units, respectively. For an SI-unit input file, these same output units would be selected with

```
L 3.2808 ft         L 1.0    m          L 100.0   cm
T 1.0    s          T 1.0    s          T 1.0     s
F 0.2247 lb         F 1.0    N          F 100000. dyn
```

For some really bizarre output, you can even specify

```
L 0.00497     furlong
T 0.827e-6    fortnight
F 0.01605     stone
```

for an SI-input file.

Comedy aside, ASWING has the following dictionary of units

```
L:  m  mm  cm  in  ft
T:  s  sec  min
F:  N  kg  g  oz  lb
M:  kg  g  oz  lb  slug
```

and will make appropriate conversions where possible to obtain simpler unit names. For example, if m,s,N are specified, then "N-s^2/m" will be replaced with the much nicer "kg". But if cm,s,lb (ugh!) are specified, then "lb-s^2/cm" will be used as the name of the mass unit, since there is no alternative standard name for it.

If a unit type is specified more than once, the last one will be taken. For example,

```
L 3.2808 ft
L 1.0 m
```

is equivalent to

```
L 1.0 m
```

"Constant" block (required)

This specifies the acceleration due to gravity, the sea-level air density, and the sea-level speed of sound, all in the assumed input units. ASWING has its own tables for the US Standard Atmosphere, which are used to set the air density for an interactively-specified altitude. The specification of the sea-level density is needed to scale the tabulated density to the user's own units. When the program starts up, the gravitational acceleration and sea-level air density are displayed in the output units as a useful check on the Unit specification block.

If more than one data line is present, the last one will be taken as input.

"Reference" block (first line required)

This block contains one to four lines

The first line gives the reference area, chord, and span used to calculate the overall CL, Cm, Cn, CDi, span efficiency, etc. This line is required.

The next three lines are optional, and give the reference x,y,z locations for

 moments
 accelerations
 velocities (and corresponding flow angles)

in that order. Omitted lines default to 0,0,0

The moment reference location will typically be the aircraft C.G. location
for static stability calculations. The acceleration reference location will
typically be the accelerometer position. The velocity reference location
will typically be the location of the air data probe (pitot, alpha+beta vanes).
If air data is taken from calibrated ports rather than a probe, then 0,0,0
is probably the appropriate reference location.

Note that for an ASWING model which represents an entire aircraft in flight
and which is to be trimmed for zero net applied + inertial-reaction moment,
the moment reference location is immaterial. Similarly, the acceleration
and velocity reference locations are significant only if the angular
velocities or angular accelerations are nonzero.

"Weight" block (optional)

This allows specification of "point-masses", or point-objects to be more
precise. Each point-mass possesses mass, and also angular momentum,
aerodynamic drag-area, and aerodynamic volume. Each is cantilevered from
a surface or fuselage beam via a massless rigid pylon. The pylon attachment
point is at location t on the reference axis of beam number "Nbeam".
The other free end of the pylon is specified by the cartesian
coordinates Xo, Yo, Zo, which is where the point-mass is located
in the undeformed state (i.e. when the beam has its jig shape).
The pylon is assumed to be rigid (infinitely stiff), and will wave
the point-mass around as the beam section at t moves and rotates
during deformation. The angular momentum vector's direction will
also change accordingly.

One can specify fewer than all the possible data items on one line.
The missing values will default to zero.

Note: The weight (not the mass) of the point-mass is given by the number
"Weight". This convention of specifying weight rather than mass was chosen
because of the traditional use of English pound-mass rather than the slug to
define mass. This is hardly an inconvenience if using SI units (kilograms).
Just include a factor of g (9.81) in the weight's multiplier line.

The aero drag force vector $\bar{D}$ on the point mass is calculated from the

specified drag area CDA as

```
 _              _
 D = 0.5 rho V V  CDA
```

where $\vec{V}$ is the net local relative velocity vector, and $V = |\vec{V}|$.

In some cases, it is desirable to give the point-mass some nonzero inertias.
This can be achieved by replacing the single physical point-mass into
a number of masses of the same total mass, distributed in space appropriately.
If the point-mass has mass "m", and the desired inertia tensor is diagonal,

```
     |Ixx 0  0 |
 I = | 0 Iyy 0 |
     | 0  0 Izz|
```

then the simplest equivalent configuration is six masses each with mass m/6,
arranged in three dumbbells with spans +/-x, +/-y, +/-z. These spans
are computed by

```
 x  =  sqrt[ (3/2) (Iyy + Izz - Ixx) / m ]  =  (rgy + rgz - rgx) sqrt(3/2)
 y  =  sqrt[ (3/2) (Izz + Ixx - Iyy) / m ]  =  (rgy + rgx - rgy) sqrt(3/2)
 z  =  sqrt[ (3/2) (Ixx + Iyy - Izz) / m ]  =  (rgz + rgy - rgz) sqrt(3/2)
```

where rgx,rgy,rgz are the radii of gyration. The dumbells can be
conviently implemented by using the multiplier and adder lines.
If the single point-mass is:

```
Weight
# Nbeam  t    Xo   Yo   Zo    Weight   CDA    Vol    Hxo    Hyo    Hzo   [ Ixx
 Iyy   Izz   Ixy   Ixz   Ixz ]
*        1.   1.   1.   1.    1.       1.     1.     1.     1.     1.
   1     2.5 -2.5  0.0 -1.5   2500.0   0.1    0.01   0.0    0.0    0.0
```

and the dumbell spans are determined to be x,y,z = 0.5, 0.3, 0.1 using
the above relations, then the modified Weight block which implements this is:

```
Weight
# Nbeam  t    Xo   Yo   Zo    Weight   CDA    Vol    Hxo    Hyo    Hzo
*        1.   1.   1.   1.    0.1667   0.1667 0.1667 1.     1.     1.

+        0.   0.5  0.   0.    0.       0.     0.     0.     0.     0.
   1     2.5 -2.5  0.0 -1.5   2500.0   0.1    0.01   0.0    0.0    0.0
+        0.  -0.5  0.   0.    0.       0.     0.     0.     0.     0.
   2     2.5 -2.5  0.0 -1.5   2500.0   0.1    0.01   0.0    0.0    0.0

+        0.   0.   0.3  0.    0.       0.     0.     0.     0.     0.
```

```
   3     2.5 -2.5  0.0 -1.5   2500.0   0.1    0.01   0.0    0.0    0.0
+        0.   0.  -0.3  0.    0.       0.     0.     0.     0.     0.
   4     2.5 -2.5  0.0 -1.5   2500.0   0.1    0.01   0.0    0.0    0.0

+        0.   0.   0.   0.1   0.       0.     0.     0.     0.     0.
   5     2.5 -2.5  0.0 -1.5   2500.0   0.1    0.01   0.0    0.0    0.0
+        0.   0.   0.  -0.1   0.       0.     0.     0.     0.     0.
   6     2.5 -2.5  0.0 -1.5   2500.0   0.1    0.01   0.0    0.0    0.0
```

For the more general case where the inertia tensor is not diagonal, it is probably simplest to first obtain the principal axes of this inertia tensor, and then set up the three dumbells along those axes.

"Sensor" block (optional)

This allows specification of "sensors", which measure various quantities at their locations or their pylon anchor points. Like a point-mass described above, each sensor is cantilevered from a surface or fuselage beam by a rigid pylon. The quantities measured by the sensor can be plotted, written out to files, or used in the user-supplied control routines described later. The sensor has three sensor directions V,A,B, used to sense the airspeed and flow angles. These would correspond to the pitot direction, and the beta,alpha vane pivot axes. The V,A,B vectors are defined by the two vectors Vspec and Aspec in the sensor specification line shown above.

```
  V = Vspec
  B = Aspec x Vspec
  A = V x B
```

so that B is perpendicular to the plane defined by Vspec,Aspec, and A lies in the plane defined by Vspec,Aspec, and is perpendicular to Vspec. The magnitudes of these vectors are be arbitrary, and only their directions are significant. If the Vspec,Aspec vectors are omitted from the sensor specification data line, the usual V,beta,alpha directions are used as the defaults.

```
  V = 1,0,0
  B = 0,1,0
  A = 0,0,1
```

These specified or default directions correspond to the undeformed geometry. The sensor will be waved around by its anchor pylon

```
as the structure deforms, changing the sensor's position, velocity,
and sensing directions accordingly.  The sensor V,alpha,beta data
is also affected by the induced velocity of all the surfaces and
volumes present, as well as the gust field.  In summary, it
behaves just like a normal V,beta,alpha pitot/vane unit.

Below are the quantities delivered by each sensor.

Scalars:
- - - -
  V        velocity        | In sensor axes
  Beta     sideslip        | (simulates TAS pitot, alpha,beta vanes)
  Alpha    angle of attack |

  Phi      roll angle      | In sensor axes
  Theta    elevation angle | (simulates position gyros)
  Psi      heading angle   |

 Int[V   -Vc   ]dt         | Error integrals in sensor axes
 Int[Beta-Betac]dt         |
 Int[Alph-Alphc]dt         |
 Int[Phi -Phic ]dt         |
 Int[Thet-Thetc]dt         |
 Int[Psi -Psic ]dt         |


Vectors:
- - - -
 dU/dt   absolute acceleration     | in sensor axes (simulates accelerometers)
 dW/dt   absolute ang.acceleration | in sensor axes (simulates rot.accelerom.)
  W      absolute rotation rate    | in sensor axes (simulates rate gyros)

  Re     position in earth frame |  In earth axes  (simulates GPS pos.sensor)
  Ue     velocity in earth frame |  In earth axes  (simulates GPS vel.sensor)

  F      resultant beam load   at pylon anchor   |  In csn axes
  M'     resultant beam moment at pylon anchor   |  In csn axes

  r      relative position       (in aircraft axes, for monitoring deflections)
  u      relative velocity dr/dt (in aircraft axes, for monitoring defl.vels.)

 Int[W-Wc]dt    rotation rate error integral in sensor axes
```

"Engine" block (optional)

Engines behave like point masses which have propulsive forces and moments applied to them. Normally, the force and moment are along the engine axis, although off-axis components can be defined to represent P-factor loads by using the Engine model type 2 described farther below.

An "Engine" is declared separately from a "Weight" so that the number of engines can be declared independently of the number of weights present. The declared number of engines needs to be known to allow calculation of engine-power response curves in the frequency-domain analyses.

An "Engine" does not have any mass, drag area, volume, or angular momentum. These can be provided by declaring a "Weight" with these attributes at the same location as the Engine.

One can specify fewer than all the possible Engine data items on one line. The missing trailing values will default to zero. The two engine lines below are equivalent:

```
# Keng IEtyp Nbeam  t    Xo   Yo   Zo    Tx  Ty  Tz  dFdPe dMdPe  Rdisk Omega
cdA

    1    0    1    12.0 -2.5  8.0 -1.5  -1.0 0.0 0.1  10.0

    1    0    1    12.0 -2.5  8.0 -1.5  -1.0 0.0 0.1  10.0   0.    0.    0.
    0.
```

The engine forces and moments are specified interactively during ASWING execution via an arbitrary unit "PengK", such as "%-power" or "throttle position". The engine force and moment is computed in SUBROUTINE ENGINE (engine.f). This routine can implement multiple types of engine models. The type of model is given by the IEtyp index on each Engine definition line. The engine model type can also be changed at runtime via the "E" command from the OPER menu.

Currently, SUBROUTINE ENGINE contains three engine types:

type 0: A simple proportional engine model. Peng is an arbitrary control variable,
and thrust and torque are set using the two gains:

$$T = dFdPe * Peng$$
$$Q = dMdPe * Peng$$
$$F = (Tx , Ty , Tz) * T$$
$$M = (Ty , Ty , Tz) * Q$$

```
 type 1: An extended actuator-disk model, with thrust and torque set by the
 model
            T = f( Peng , rho , V , Rdisk )
            Q = Peng/Omega

 type 2: Same as type 1, with added P-factor terms for prop whirl prediction

The Steady theory document asw.ps describes types 0 and 1.
The Unsteady theory document aswu.ps describes the added terms in type 2.
Additional models could be implemented as needed.  The data items
dFdPe...cdA  can then contain other information which might be
useful for passing to the routine.  Note that these will still be put
into the user's force and moment units!

textbfEngine model type 0 This is the simplest model, which assumes the engine force
and moment are simply proportional to the power setting "PengK",

  F = (dFdPe) * PengK
  M = (dMdPe) * PengK

where "K" is the engine-variable index given by the Keng number
in each Engine definition line.  The same Keng can appear
in more than one line, which indicates that this Keng controls
more than one engine.  In the example block shown above,
Peng1 controls collective thrust and Peng2 controls differential thrust
between the left and right engines.  Each left and right engine is
modeled by superimposing two point engines, each driven by one PengK.

The thrust force direction is given by the Tx,Ty,Tz engine-axis vector,
whose magnitude is not significant.  For an undeformed geometry,
the thrust force and moment will be

  Fxyz  =  F * Txyz / | Txyz |
  Mxyz  =  M * Txyz / | Txyz |

For a flexible configuration, this vector will be "waved around"
as the structure holding the engine deforms.

F and M are defined as the net airload force and moment applied to
the engine/propeller.  For a typical righthanded prop, with Txyz
pointing in the flight direction, dFdPe will be positive,
and dMdPe will be negative.
```

Engine model type 1

```
This is an extended actuator-disk model, with swirl loss and profile drag loss
```

```
contributions included.  The power control variable PengK is taken as the
shaft power, in whatever units are accepted by SUBROUTINE ENGINE.
The current implementation assumes Peng is in the same output units
as all the other dimensional user runtime inputs.

This model requires the definition of the "engine axis" unit vector,
denoted as x_e in the documentation.  This is defined using the Tx,Ty,Tz
components in the Engine data line.  For the undeformed geometry,
the engine axis vector is taken to be:

 x_e0  =  -Txyz / |Txyz|

This vector will be "waved around" as the structure holding the engine deforms,
which then defines the actual engine axis vector x_e.

Besides Tx,Ty,Tz, the engine model type 1 requires the following additional
data items on each Engine line:

 Rdisk =  propeller or fan disk radius
 Omega =  approximate baseline prop rotation rate
 cdA   =  total blade profile drag area   =  Nblades * Rdisk * chord * cd
```

Engine model type 2

```
This is the same as type 1, but also adds "P-factor" forces and moments
in the prop disk plane, along y_e and z_e engine-axes vectors, both
perpendicular to x_e

These P-factor forces and moments are produced by transverse velocities
and prop-mount rotation rates.  Besides the data for type 1, the type 2
engine model requires the following ten additional data items on each
Engine line:

 cl  = typical blade section cl     (this value has only a minor contribution)
 CLa = blade section 2D dCL/dalpha  (this value dominates the results)
 S0 =  B Int [ c/R sin(phi)         ] d(r/R)
 C0 =  B Int [ c/R cos(phi)         ] d(r/R)
 S1 =  B Int [ c/R sin(phi)  r/R    ] d(r/R)
 C1 =  B Int [ c/R cos(phi)  r/R    ] d(r/R)
 S2 =  B Int [ c/R sin(phi) (r/R)^2 ] d(r/R)
 C2 =  B Int [ c/R cos(phi) (r/R)^2 ] d(r/R)
 S3 =  B Int [ c/R sin(phi) (r/R)^3 ] d(r/R)
 C3 =  B Int [ c/R cos(phi) (r/R)^3 ] d(r/R)

The radial blade integrals S0..C3 can be approximated as suggested
in the Unsteady theory document.
```

Note that the P-factor forces and moments can be easily turned off
just by changing IEtyp=2 to IEtyp=1, with no other effects.
This may be necessary in cases in time-marching cases which
may have flutter instability present, but which cannot be
economically resolved in time.

textbfEngine jet

Whatever the engine model is used, ASWING will model the resulting
propulsive jet, as described in the documentation. This jet model
requires only the propulsive disk radius Rdisk, specified in the
Engine data line (and also used in engine models 1 and 2).

Specifying a negative Rdisk will disable the propulsive jet model,
but still allow engine model type 1 and 2 to be used as usual
(SUBR. ENGINE is unaffected by the sign of Rdisk).

The engine jet lies in the same direction as the trailing vortices
of the vortex-lattice model. This is either along the freestream
velocity, or along the x-axis if in "fast" mode. The fast mode
is toggled on and off in OPER, K submenu, item F.

An engine jet is visible to surface and fuselage beams,
to point weights, and to sensors.

An engine jet is NOT visible to other engines by choice. Accounting
for jet interactions would require simultaneously solving for the thrust
and torque of all engines including their jet parameters. The would
require detailed vortex/blade-element prop analyses, which in turn
require voluminous input of the detailed propeller blade geometry.
This is impractical for the modeling level used in ASWING.

"Strut" block (optional)

This allows the specification of struts or wires, each of which is attached
at the end of a rigid pylon cantilevered from the specified t location
of beam number "Nbeam". Xo, Yo, Zo, give the location for the other pylon
endpoint just like for the point-masses. Here, this other endpoint is
where one end of the strut is attached.

Xw, Yw, Zw give the location of the anchor at the other end of
the strut. This other end is assumed to be grounded (i.e. fixed
in the x,y,z coordinate system).

The pylon mount for the strut is provided to simulate a strut attachment

```
which may not be exactly at the beam axis.  For example, a typical strut
attached to the bottom flange of an I-beam spar is in effect mounted
on a short pylon of length equal to half the spar depth.  This pylon
offset produces a small but likely non-negligible moment load at the
strut attachment.

dLo is the slack length added to the strut length, in effect modifying
its unloaded length which is defined as follows.

                     2          2          2
 Lo  =  sqrt( (Xo-Xw)  + (Yo-Yw)  + (Zo-Zw) )  +  dLo

This is intended to simulate strut or wire slack or preload in
a typical braced-wing application.

EAw is the extensional stiffness of the strut.  It can be made huge
to simulate an effectively-rigid strut.

As in the case of point-masses, multiple struts at the same location
have their contributions superimposed.  The net effective EA is simply
the sum of the individual EA values.
```

"Joint" block (optional)

```
This allows specification of some number of "joints" between
pairs of beams.  The joints are specified as follows.

Joint
#  Nbeam1  Nbeam2    t1    t2    [ KJtype ]
*                    1.    1.
    1       2        0.0   0.0      0
    1       3       10.0   2.0      1
    2       3        2.0   5.0      3
End

A joint is a rigid pylon linking the two beams at the specified t locations.
The optional KJtype indicator specifies the type of joint:

KJtype = 0:  Both translation and rotation matched    (rigid  joint)
         1:  Only translation matched                 (pinned joint)
         2:  Only rotation matched                    (roller joint)
         3:  translation matched, two angles matched,
             one remaining angle depends on moment    (sprung-hinge joint)

If the KJtype value is omitted on the data line, then KJtype=0 is assumed,
which defaults to previous ASWING versions.
```

```
In the above example,

For joint 1 (1st line):
Location t=0 on beam 1 and location t=0 on beam 2 must move as a rigid unit.

For joint 2 (2nd line):
Location t=10.0 on beam 1 and location t=2.0 on beam 3 are separated
by a rigid distance, but are free to rotate relative to each other.

For joint 3 (3rd line):
Location t=2.0 on beam 2 and location t=5.0 on beam 3 are separated
by a rigid distance, and have a hinge angle dependent on the local
structural moment.  The hinge axis direction and angle(moment) function
must be specified in a separate "Jangle" block, described below.


It is essential NOT to connect two perfectly rigid beams with more than
one joint, as one might be tempted to do in a twin-boom configuration.
Physically, the two joints impose a strain on a perfectly rigid structure,
which gives an ill-posed problem.  If such a solution is attempted, the
system Newton matrix will be singular and an arithmetic fault will likely
result.  Multiple joints can be used if the two beams are compliant in
the appropriate deformation modes.  This requires that the extensional,
bending, and torsional stiffness distributions ---  EA, EIcc, EInn, GJ
--- have finite values.  And even if the values are finite, if they
are too large then the matrix will be ill-conditioned and the solution
may fail to converge.
```

"Jangle" block (required if KJtype=3)

```
"Jangle" block (required if joint with KJtype=3 is declared in "Joint" block)

This gives the information necessary to implement any KJtype=3
joints declared in the "Joint" block.  One such "Jangle" block
must be present for each KJtype=3 joint declared.

Jangle
# Njoint  hx   hy   hz
  2       1.0  0.2  0.
#
#  Momh    Angh
*  1.0e5    1.0
+  0.       0.
  -15.0  -45.0
  -10.0   -5.0
  -10.0   -5.0
```

```
   -5.0   -2.0
    0.0    0.0
    5.0    2.0
   10.0    5.0
   10.0    5.0
   15.0   45.0
End


NJoint is the joint index (numbered sequentially in the "Joint" block)

hx,hy,hz   is the hinge-axis vector for the jig geometry.
This moves appropriately as the structure deforms.

Momh  is the hinge moment about the hinge-axis vector
Angh  is the resulting hinge angle, in degrees.

The Angh(Momh) data is splined.  Two successive identical Momh values
enable a slope break at that point.  In the example data above,
there are slope breaks at Momh=-10.0 and at Momh=10.0 .
If a segment of the curve has only two points, the spline
will reduce to a straight line interpolation between these two points.

The Momh data must be monotonic, with Momh
strictly increasing (except for the doubled points).
The  dAngh/dMomh  slope must have the appropriate sign in order
to produce a restoring hinge moment for any hinge angle deflection.
Otherwise the joint will be statically unstable and the solution
will blow up.

To check:

If a positive structural joint moment about hxyz
results from a positive beam #2 rotation about hxyz,
then dAngh/dMomh > 0 must be specified.

Otherwise, dAngh/dMomh < 0 must be specified.



If the structural moment exceeds the Momh data range, then
the spline curve will be extrapolated.  For this reason, it is
prudent to have a two-point segment at each end of the range,
which is straight and hence will always extrapolate safely
and preserve monotonicity.
```

"Ground" block (required)

This allows specification of some number of "ground" points at specified
beam locations, in the following format:

```
Ground
#  Nbeam    t    [ KGtype ]
    1       0.0     0
    2      10.0     2
End
```

A ground point imposes various kinematic constraints on the beam at
the ground location. Three types of ground points can be specified:

```
KGtype = 0:  Both translation and rotation prevented  (rigid  mount)
         1:  Only translation prevented               (pinned mount)
         2:  Only rotation prevented                  (roller mount)
```

If the KGtype value is omitted on the data line, then KGtype=0 is assumed.

At least one translation ground and one rotation ground, at the same
or different locations, must be specified for each distinct group of
the configuration to restrain the overall rigid-body modes. A group
is defined as one isolated beam, or a number of beams interconnected
by joints. It is recommended that the rigid mount (KGtype=0) be used
if possible. It is less likely to cause ill-conditioning difficulties
in cases with large deformations.

2.3.3 Beam Definition Blocks

Types of Beam Definition Blocks

The definition of each surface-beam or fuselage-beam consists of the
the keyword "Beam" followed by the beam number, and an optional
physical beam index. The beam number is simply for the user's
reference and can be any integer. The physical beam index is
described later.

The keyword line is then followed by another line specifying the beam name,
which is then followed by a number of distribution sub-blocks,
in the following format.

```
\subsubsectionBeam Definition Block Examples

Beam 1  20              ! beam 1 definition begins, optional physical index 20
Main wing                     ! beam name string
#                             ! comment line
   t     chord    alpha       ! distribution block for chord,alpha begins
+ 0.     0.       3.          ! optional additive-constant line
* 1.     1.       1.          ! optional scaling-factor line
  0.0    1.3      3.0         ! data
 20.0    1.2      2.8         !  "
 20.0    1.2      2.8         !  "
 31.0    0.8      2.0         !  "
 34.0    0.7      1.7         !  "
#
   t      x      y     z      ! new distribution block for x,y,z begins
  0.0    0.0    0.0   0.5     ! data
 17.0    5.0   15.0  -0.2     !  "
 34.0   10.0   30.0  -2.0     !  "
End                           ! end of beam 1 definition



Beam 2                        ! new beam 2 definition begins
Fuselage                      ! beam name string
#                             ! comment line
   t      x      y     z      ! new distribution block for x,y,z begins
  0.0    0.0    0.0   0.5     ! data
 10.0    5.0    0.0   0.6     !  "
 20.0   10.0    0.0   0.7     !  "
End                           ! end of beam 2 definition


Whether a beam is a "surface" or "fuselage" depends on whether
a chord distribution for that beam is present.
```

Logical and Physical indices

```
Logical index:
This is the number immediately after the "Beam" keyword.
The example above has logical index = 1.  The logical index
is used to identify or select the beam in question, and is
simply an alternative to the beam's name.


Physical index:
The optional physical index is the second number after the "Beam" keyword.
```

```
The example above has physical index = 20.  The purpose of the physical index
is to allow a single aerodynamic object, like a wing + winglet,
or fin + stab of a T-tail, to be defined as separate logical beams.
The two beams are assumed to be the same physical beam
if they share the same physical index. e.g.

Beam 1  20
Main wing
.
.

Beam 5  20
Winglet
.
.

Any two beams which share a physical index like this are assigned
a zero vortex core radius when their mutual vortex lattice influences
are computed.  This improves accuracy in prediction of near-field
effects such as circulation carryover from wing to winglet,
and the endplating effect of a stab on the fin of a T-tail.
However, care must be taken that the trailing vortex legs from
one such beam do not come close to the control points of the other beam.
The absence of a finite core radius will then likely cause serious
numerical noise.

If the physical index is absent, it is assumed to be the same
as the logical beam index.  e.g. the following two declarations
are equivalent:

Beam 1
Beam 1  1

Therefore, one must be careful to avoid using a physical index
which is the same as another beam's logical index.
```

Distribution blocks

```
As the comments above indicate, each Distribution Block for a beam begins
with a specifier line which declares which variables are being defined
in that Block.  The first column is the arbitrary spanwise parameter "t",
not necessarily the arc length.

The data is interpolated in t using cubic splines, which normally produce
slope and curvature continuity across the data points.  A discontinuity
can be specified with two successive identical t values, which will produce
```

a separate interpolating spline on each side. In the first block above, two separate splines are used over the two intervals:

```
 0.0    1.3      3.0    |  interval 1
20.0    1.2      2.8    |

20.0    1.2      2.8    |  interval 2
31.0    0.8      2.0    |
34.0    0.7      1.7    |
```

If only two points are used, as in interval 1, then the cubic spline reduces to linear interpolation over that interval.

The spline interpolation requires that t be monotonically increasing. If any distribution has decreasing t values, the t values and associated data will be reversed in order. This allows specifying a left-side surface from a right-side surface by simply giving all its t values a negative multiplier. Other data, such as the y coordinates, will also need a negative multiplier in this case.

- - - - -

For surface beams only:

Circulation is defined positive by righthand rule in the direction of increasing t. Hence, **t should increase from aircraft left to right.** This will give the standard sign conventions for the airfoil section properties alpha, CL, Cm, etc. **Cm>0 if the aircraft pitches up**

Left/right symmetry is assumed for a surface beam if the first t value for the chord distribution is **identically zero.** A negative-t image side of the surface will be created by direct copying of the chord:

```
chord(-t) = chord(t)
```

As with the chord distribution, **every other variable of a surface beam will be tested for symmetry as well**. Again, if an variable's first t value is zero, its distribution will be copied to the corresponding negative-t side, e.g.

```
x(-t) = x(t)
z(-t) = z(t)
alpha(-t) = alpha(t)
 .
 .
```

The exception is the image y value definition, which consists of

a reflection about the t=0 value:

y(-t) = 2*y(0) - y(t)

These symmetry conventions are intended to streamline the specification
of a typical y-symmetric aircraft configuration. It should be noted
that symmetry is individually defined for each variable based on its
first t value. Hence, a wing with symmetric x,z, chord, twist, etc,
can still have an asymmetric aileron derivative dCLdF, but this must
then be defined via negative and positive t values across the whole span.

No symmetry conventions are assumed for point-masses and struts.
Each one must specified individually.

- - - - -

The following variable names must be used in the variable specifier line
of each Distribution Block. Each variable is input in the unit shown.
If the variable is omitted, the indicated default values are used.
As defined in the theory document, c,n are the airfoil-plane coordinates.
Hence, c/chord = "x/c" and n/chord = "y/c" in airfoil jargon.

Variable	Unit	Default	
x	L	0	zero-load x location
y	L	0	zero-load y location
z	L	0	zero-load z location
twist	deg	0	zero-load twist angle about s-axis
EIcc	FLL	infinity	bending stiffness about c-axis (out-of-plane bend)
EInn	FLL	infinity	bending stiffness about n-axis (in-plane bend)
EIcn	FLL	0	bending cross-stiffness
EIcs	FLL	0	n-bending/torsion coupling stiffness
EIsn	FLL	0	c-bending/torsion coupling stiffness
GJ	FLL	infinity	torsional stiffness
EA	F	infinity	extensional stiffness
GKc	F	infinity	c-shear stiffness
GKn	F	infinity	n-shear stiffness
mgcc	FL	0	weight-inertia/span about c-axis
mgnn	FL	0	weight-inertia/span about n-axis
mg	F/L	0	weight/span
Ccg	L	0	c-location of section mass centroid
Ncg	L	0	n-location of section mass centroid
Dmgcc	FL	0	additional weight-inertia/span about c-axis
Dmgnn	FL	0	additional weight-inertia/span about n-axis
Dmg	F/L	0	additional weight/span
DCcg	L	0	c-location of additional-mass centroid

```
 DNcg      L      0       n-location of additional-mass centroid
 Cea       L      0       c-location of elastic axis
 Nea       L      0       n-location of elastic axis
 Cta       L      0       c-location of tension axis
 Nta       L      0       n-location of tension axis
 tdeps     T      0       normal strain damping time
 tdgam     T      0       shear  strain damping time
 Cshell    L      0       c-Cea  where strain is evaluated (for display only)
 Nshell    L      0       n-Nea  where strain is evaluated (for display only)
 Atshell   L^3    0       A*T for shear stress evaluation (display only),
                             where A is the torsion shell enclosed area,
                                   T is the torsion shell thickness.
 radius    L      0       beam cylinder radius (for fuselage beam only)
 Cdf              0       section profile friction drag coefficient
 Cdp              0       section profile pressure drag coefficient

 chord     L     ---      wing chord
 Xax              0.5      distance/chord of s-axis from leading edge
 alpha    deg     0       angle of zero-lift line above c-axis
 Cm               0       section pitching moment coefficient about chord/4
 CLmax           2.0      section maximum lift coefficient
 CLmin          -2.0      section minimum lift coefficient
 dCLda   1/rad   2 pi     section lift-curve slope
 dCLdF1  1/flap   0       dCL/dFlap1 derivative ("flap" is in any units)
 dCMdF1  1/flap   0       dCM/dFlap1 derivative
 dCDdF1  1/flap   0       dCDp/dFlap1 derivative
 dCLdF2  1/flap   0       dCL/dFlap2 derivative
 dCMdF2  1/flap   0       dCM/dFlap2 derivative
 dCDdF2  1/flap   0       dCDp/dFlap2 derivative
  .
  .
  .
```

NOTE: flap units are random; but keep in mind that the unit used in the .asw file is the same as the unit of the flap deflection indicated by the solver

The variables starting with "chord" pertain only to a surface beam, and are ignored for a fuselage beam. The "radius" variable is ignored for a surface beam. Data for any variable can be omitted, with the exception of "chord" for a surface, and at least one of "x", "y", or "z".

The infinity defaults produce **perfect rigidity** for the corresponding deformation. **The variables with infinity defaults can also be specified as zero, which is taken to be infinity as well.** This simplifies changing of a normally-flexible structure to be rigid, since a variable can be temporarily set to zero via its multiplier.

Note that some number of separate flaps can be defined among all the
surfaces present. These can then be independently deflected during
interactive execution. A flap extending over part of the span is
specified by setting its derivatives nonzero over that part of the span,
and zero elsewhere:

```
   t    dCLdF1   dCMdF1
  0.0    0.       0.
 25.0    0.       0.
 25.0    0.08    -0.02
 30.0    0.08    -0.02
 30.0    0.       0.
 32.0    0.       0.
```

Together with the interactively-specified "Flap1" deflection,
these coefficients will be used to modify the net local 2-D section
zero-lift angle, and the net local Cm, as described in the theory
document:

$$\alpha_{net} = \alpha + Flap1 * dCLdF1/dCLda$$

$$Cm_{net} = Cm + Flap1 * dCMdF1$$

An aileron is defined by specifying non-symmetric distributions
of opposite sign across the entire span:

```
   t    dCLdF2   dCMdF2
-32.0    0.       0.
-30.0    0.       0.
-30.0   -0.08     0.02
-25.0   -0.08     0.02
-25.0    0.       0.
  0.0    0.       0.
 25.0    0.       0.
 25.0    0.04    -0.01
 30.0    0.04    -0.01
 30.0    0.       0.
 32.0    0.       0.
```

Note that in this case the left aileron will have twice the effective
deflection of the right aileron for a given specified "Flap2" variable.
This simulates variable up/down aileron scheduling, although this
can be valid only if "Flap2" is always of one sign. Also, the frequency

```
response calculations use a linearized formulation, and require that
the aileron flap derivatives are equal and opposite to properly model
the usual geometrically-symmetric left/right ailerons.


In general, the influence of all flap variables is always superimposed.
The relation for Cm which is actually implemented is

 Cm     =  Cm  +  Flap1*dCMdF1  +  Flap2*dCMdF2  +  Flap3*dCMdF3 ...
   net

where the series includes all flap variables defined in xxx.asw,
and likewise for alpha_net.  Hence, control surface mixing can
be represented.  In a V-tail, Flap1 might be "elevator", and Flap2
might be "rudder", with both variables contributing to both tail
halves.

Similarly, the dCDdF derivatives augment the local pressure drag coefficient.

 Cdp    =  Cdp  +  Flap1*dCDdF1  +  Flap2*dCDdF2  +  Flap3*dCDdF3 ...
    net

This is intended mainly to model the drag of a spoiler or drag-type yaw device.
Conventional trailing edge surfaces like ailerons have a profile drag rise
with is roughly quadratic with surface deflection, so this linear model is
not well suited for this case.

A "Flap" variable does not have to be associated with only one
surface.  For example, consider a case with two surfaces present,
with a dCLdF1 distribution is specified for each surface,

Beam 1
Wing
  t    dCLdF1
-20.0  -0.10
-10.0  -0.10
-10.0   0.0
 10.0   0.0
 10.0   0.10
 20.0   0.10
End

Beam 2
Rudder
  t    dCLdF1
 2.0   -0.30
 5.0   -0.30
```

```
End

The "Flap1" variable will actuate the ailerons and the rudder
simultaneously, in a 1:3 proportion.
```

2.4 Program Execution

2.4.1 Starting ASWING

```
ASWING is executed with

 % aswing

which immediately brings up the top-level ASWING menu:

 ======================================================

   LOAD f Read  geometry data file
   SAVE f Write geometry data file
   CASE f Change default filename prefix  ( hawk )

  .NODE   Plot/change grid node distributions
  .PLPA   Plot-page options

  .PLOT   Plots
  .OPER   Calculate operating point(s)
  .MODE   Calculate eigenmodes
  .BODE   Calculate frequency responses
  .EDIT   Edit structural distributions

   GGET f Read gust-parameter file
   GMOD   Modify gust parameters

   PSAV f Write operating-point save file
   PGET f Read  operating-point save file

   SSAV f Write settings file
   SGET f Read  settings file

   CGET f Read control-law file

   DIMS   Report current array usage and max dimensions
   UALT   Toggle units used for altitude
   NAME s Change case name
```

```
   QUIT   Exit program

 ASWING   c>
```

The commands preceded by a "." simply enter another menu (the "." is not typed). The other commands are executed immediately.

The lowercase letters "f,s" after some of the commands indicate that these commands expect arguments. "f" indicates that a filename is expected, and "s" indicates and arbitrary character string. If an argument is not given, a prompt will result.

Input/Output

The LOAD command is used to read and process an input file at any time. This can be bypassed if ASWING is executed with

```
 % aswing xxx
```

where "xxx" is an arbitrary optional Unix command argument.
After it starts, ASWING will immediately try to read and process the input file xxx.asw, so then the LOAD command can be skipped.

The SAVE command will generate an input file which can be read into ASWING later. This is useful if the structural and/or aerodynamic description has been modified via the EDIT menu (described later).

2.4.2 Input Checking

Immediately after the input file is processed, a number of integrated quantities like total weight, area, etc. are displayed for each beam and for the entire configuration as a check on correctness. The PLOT menu also allows visual examination of the geometry and some of the primary structural parameter distributions.

Some quantities like chord, EIcc, GJ, etc, absolutely cannot be allowed to be negative at any location. Even if these are positive at all input t locations, it is quite possible that they can be negative in between these t locations due to spline overshoots. All quantities in the PLOT and EDIT menus are displayed exactly as splined, so that

such problems can be quickly diagnosed. Overshoots can be fixed by adding more input t locations, and/or using doubled points.

ASWING performs a number of sanity checks on the input. When a new case is being run, the voluminous output which appears at program startup should be carefully checked for reasonableness, and also for any warnings or errors which might appear.

2.4.3 Saving/recalling Settings, Operating Points

The SSAV command can be issued at any time to save the current settings in a file. Only settings which pertain to all operating points go into this file. Setting and parameter values for the individual operating points go into another point-save file, which is written with the PSAV command.

The PSAV command at top level will write all the current operating-point definitions to a specified file. The default file is "xxx.pnt", which is used if just a <return> is pressed at the prompt.

The operating-point definitions can then be read back into ASWING later with the PGET command, so that they do not have to be laboriously entered again. If ASWING is started with the "xxx" case argument, it will try to read the default "xxx.pnt" operating-point file automatically. In this case it is not necessary to read it with PGET.

The default filenames are xxx.set and xxx.pnt, which are assumed if just <return> is pressed at the filename prompt. Any other filename can be specified, however. Use of the settings save file is essential in practice, since it is quite impractical to remember and type in all the necessary settings for a given case every time the program is started. The point save file is also useful for the same reasons.

When ASWING is started with argument "xxx", it looks for the following files and reads them if possible.

```
 xxx.set
 xxx.pnt
 xxx.con
```

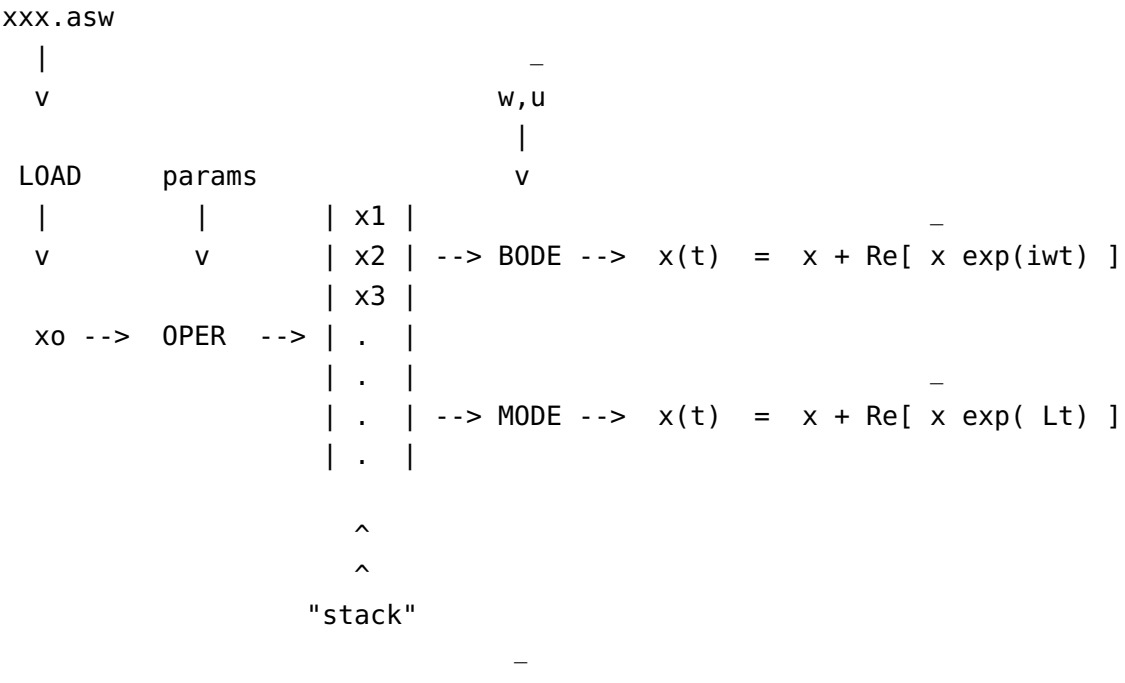
Hence, issuing SSAV and PSAV before exiting ASWING will ensure that the program state will be recovered when it is run again. The xxx.con file is currently read-only and is not written from ASWING.

If LOAD is used to read an "xxx.asw"-type input file, then the files above will have to be manually read with SGET, PGET, CGET. Using the default file name suffixes and the Unix command argument xxx is clearly simpler.

2.5 Steady-State Simulations

2.5.1 Overview

The primary activity which occurs during ASWING execution is the computation and analysis of one or more "Operating Points". Each Operating Point consists of a state "x" which is the result or solving all the nonlinear governing aero/structural equations r(x,u) = 0. The point solution sequence "stack" x1 x2 x3... consists of either isolated quasi-static solutions, or as a result of a time-march calculation. In the latter case, the point index p is essentially a time variable. Both the quasi-static and the time-march point sequences are generated from the OPER menu (described shortly), using the undeformed "jig" state xo defined in the xxx.asw file, and a handful of interactively specified parameters as inputs. These operating point states can then be further analyzed in the frequency domain in the MODE and BODE menus. The data flow is as sketched below. A more detailed version is in the PostScript file tex/dataflow.ps



The complex perturbation state $\bar{x}$ is either a forced-response solution (BODE), or an eigenmode (MODE). The forced-response solution is associated with a forcing variable u (flap,engine), driven at a specified frequency w. The eigenmode is associated with an eigenvalue L.

In each case, the output is a modified state x(t) which can be displayed

in a number of ways in the BODE and MODE menus. The forced-response modes in BODE are purely harmonic, with a constant amplitude. In contrast, the eigenmodes in MODE have their amplitudes decay or grow in time according to the magnitude and sign of Real(L), which indicates stability or instability of that mode.

2.5.2 Analysis – OPER menu

Once an input file has been read, processed, and checked, the configuration can be "flown" from the OPER menu to generate the point solutions x1,x2...

```
 ------------------------------------------------------------
  T oggle constraints (% for keybd) R etain current parameters
 eX ecute point calculation         -  delete point(s) on stack
  !  parameter-change prefix        +  add new point into stack
  @  sens.par.-change prefix        :  set new jig shape
  .  point/time-march toggle        =  list point solution

  N  load factor (Lift scale)       I nitialize point solution
  A ltitude(air density), gravity   F lap deflections
  G round effect, normal vector     E ngine forces

  L oading plot                     H ardcopy current plot
  S tructural plot                  J ot annotations on plot
  D eflection plot                  Z oom  (B for no distort)
  C urvature-strain plot            U nzoom

  V iew geometry                    O ptions for plots
  P arameter seq. plots, output     K ontrol toggles, settings

  W rite point to file              M atrix (force derivatives)

  #  point number select           Q uery point specifications
 ------------------------------------------------------------

 .OPERs (target point:  1)   c>
```

NOTE: the derivatives matrix M is only computed after computing at least one trimming point

The current stack of Operating Points can be listed any time with the "Q" command. In addition to the large number of structural and aerodynamic variables defined along each beam, a Point has the following associated quantities:

2.5.3 Parameters

```
Explicitly-specified parameters:
--------------------------------
 N         Load factor
 Mach      Mach number (for Prandtl-Glauert correction only)
 Alt       Altitude

Free parameters (solution variables):
-------------------------------------
 dUx       x acceleration
 dUy       y acceleration
 dUz       z acceleration
 dWx       Roll  acceleration
 dWy       Pitch acceleration
 dWz       Yaw   acceleration
 V         True airspeed
 Alpha     Angle of attack
 Beta      Sideslip angle
 Wx        Roll  rate
 Wy        Pitch rate
 Wz        Yaw   rate       _
 Xe        Earth X-location  | Used only to interrogate specified
 Ye        Earth Y-location  |   gust velocity field.
 Ze        Earth Z-location  |
 Psi       Heading angle     |  _
 Theta     Elevation angle   |   | Used only to determine the gravity
 Phi       Bank angle       _|  _|   vector direction in body axes
 Flap1     Flap1 deflection
 Flap2     Flap2 deflection
 Flap3     Flap3 deflection
 Flap4     Flap4 deflection
 Peng1     Peng1 power setting
 Peng2     Peng2 power setting  _
 Err_Vi    Int (Vi-Vic) dt       | State-error integrals used by
 Err_al    Int (al-alc) dt       | feedback control law routine
 Err_be    Int (be-bec) dt       |
 Err_Ph    Int (Ph-Phc) dt       |
 Err_Th    Int (Th-Thc) dt       |
 Err_Wx    Int (Wx-Wxc) dt       |
 Err_Wy    Int (Wy-Wyc) dt       |
 Err_Wz    Int (Wz-Wzc) dt      _|

also for each sensor...         _
 Err_Vis   Int (Vi-Vic) dt       | State-error integrals used by
```

```
 Err_als   Int (al-alc) dt       | feedback control law routine
 Err_bes   Int (be-bec) dt       |
 Err_Phs   Int (Ph-Phc) dt       |
 Err_Ths   Int (Th-Thc) dt       |
 Err_Wxs   Int (Wx-Wxc) dt       |
 Err_Wys   Int (Wy-Wyc) dt       |
 Err_Wzs   Int (Wz-Wzc) dt      _|



Computed parameters (dependent on solution variables):
------------------------------------------------------
 VIAS      Indicated air speed
 Fx        Total x Force
 Fy        Total y Force
 Fz        Total z Force
 Mx        Total x Moment  (roll)
 My        Total y Moment  (pitch)
 Mz        Total z Moment  (yaw)
 Lift      Aero Lift
 Di        Induced drag

also for each sensor, in sensor axes...
 Phi       bank angle
 Theta     elevation angle
 Psi       heading angle
 V         airspeed
 beta      sideslip angle
 alpha     angle of attack

 Int[Phi -Phic ]  Phi error integrator
 Int[Thet-Thetc]  Theta error integrator
 Int[Psi -Psic ]  Psi error integrator
 Int[Vinf-Vinfc]  velocity error integrator
 Int[beta-betac]  beta error integrator
 Int[alph-alphc]  alpha error integrator

 dUx Lacc
 dUy Lacc   linear acceleration (includes gravity vector)
 dUz Lacc
 dWx Aacc
 dWy Aacc   angular acceleration
 dWz Aacc
  Wx rate
  Wy rate   rotation rate
  Wz rate
```

```
 Int[ax -axc]  x-acceleration error integrator
 Int[ay -ayc]  y-acceleration error integrator
 Int[az -azc]  z-acceleration error integrator
 Int[Wx -Wxc]  x-rotation error integrator
 Int[Wy -Wyc]  y-rotation error integrator
 Int[Wz -Wzc]  z-rotation error integrator

also for each sensor, in earth axes (mimics GPS data)...
  RX pos
  RY pos   earth position
  RZ pos
  UX vel
  UY vel   earth velocity
  UZ vel

also for each sensor, in body axes...
  rx pos
  ry pos   location
  rz pos
  ux vel
  uy vel   relative velocity
  uz vel

also for each sensor, in beam csn axes...
  Mc‘ mom
  Ms‘ mom   bending moment
  Mn‘ mom
  Fc  for
  Fs  for   force
  Fn  for

also for each sensor...
  Gamma     local circulation


"Flap" corresponds to delta_F , and "Peng" corresponds to Delta_e in
the ASWING theory document.

The Xe,Ye,Ze earth position coordinates and the Psi,Theta,Phi Euler angles
are free solution variables only in time-marching and frequency-domain
calculations.  In static solutions they must be explicitly specified.

Error-integrator variables are automatically enabled when the
Control-Law toggle and the Unsteady-case toggle are both set.
Only those Err_* variables required by the Control-Law routine
will be enabled.
```

The equations which are solved to generate the Point states are controlled by user-chosen Constraints, which are equations added to the system to determine the above free parameters. These Constraints come in two flavors -- Steady and Unsteady -- although some of the "Steady" constraints can certainly be used in time-dependent computations (e.g. the commanded-flap equation Flap1 = Flap1_spec).

Steady			Unsteady		
DUx_ref	=	DUx_spec			
DUy_ref	=	DUy_spec			
DUz_ref	=	DUz_spec			
DWx	=	DWx_spec			
DWy	=	DWy_spec			
DWz	=	DWz_spec			
VIAS_ref	=	VIAS_spec	d(Ux)/dt	=	DUx
Alpha_ref	=	Alpha_spec	d(Uz)/dt	=	DUz
Beta_ref	=	Beta_spec	d(Uy)/dt	=	DUy
Wx	=	Wx_spec	d(Wx)/dt	=	DWx
Wy	=	Wy_spec	d(Wy)/dt	=	DWy
Wz	=	Wz_spec	d(Wz)/dt	=	DWz
Xe	=	Xe_spec	d(Xe)/dt	=	Ux
Ye	=	Xe_spec	d(Ye)/dt	=	Uy
Ze	=	Xe_spec	d(Ze)/dt	=	Uz
Psi	=	Psi_spec	d(Psi)/dt	=	K1.W
Theta	=	Theta_spec	d(Theta)/dt	=	K2.W
Phi	=	Phi_spec	d(Phi)/dt	=	K3.W
Flap1	=	Flap1_spec			
Flap2	=	Flap2_spec			
Flap3	=	Flap3_spec			
Flap4	=	Flap4_spec			
Peng1	=	Peng1_spec			
Peng2	=	Peng2_spec			
Fx	=	Fx_spec			
Fy	=	Fy_spec			
Fz	=	Fz_spec			
Mx_ref	=	Mx_spec			
My_ref	=	My_spec			
Mz_ref	=	Mz_spec			
Lift	=	N * L_spec			
			d(Err_Vi)/dt	=	Vi-Vic
			d(Err_al)/dt	=	al-alc
			.		
			.		

In general, the Steady constraints are used in static or quasi-static

flight point calculations. The Unsteady constraints are used in
time-marching calculations in OPER, and frequency-domain calculations
in BODE and MODE. The Steady/Unsteady flag is toggled explicitly in
the OPER menu with the "." command.

Moments, accelerations, and flow velocities (and corresponding angles)
are defined at their separate reference points Xref,Yref,Zref, which are
specified in the input file and can be changed from the "K ontrol" menu.

Rotation rates and accelerations are defined in body axes, while Lift
is defined in the conventional manner perpendicular to the freestream
and the y axis.

In the Steady mode, the primary purpose of the OPER menu is to compute
aero-structural solutions (i.e. to "operate" the aircraft) in response
to a wide variety of conditions, and immediately displaying the results.
Multiple operating points can be overlaid to permit checking of
the design throughout the operating envelope.

The number of available constraints considerably exceeds the number
of free variables, and so a great variety of aircraft flight situations
can be computed by appropriately combining a suitable subset of the
available constraints. For example, Alpha can be constrained directly
via Alpha_spec, or indirectly via the specified lift L_spec. The
flight speed V likewise can be constrained via VIAS_spec or via L_spec.
Of course, any constraint can be used at most once, so both Alpha
and V cannot both be constrained via L_spec.

In the Unsteady mode, the purpose of the OPER menu is to compute a time-
marching sequence, which is also stored on the stack as a relatively long
list of sequential operating points. For these cases the number of available
constraints is more restricted, since the state variables must evolve and
cannot be arbitrarily imposed.

Of course, one can conceive of unusual situations where some unsteady
motion is kinematically forced by external means, such as in a catapult
launch. In this case, the aircraft velocity would be imposed via the
"Steady" form U(t) = U_spec(t) rather than evolve according to the
"Unsteady" form dU/dt = F/m. Running such cases will require modification
of the Steady/Unsteady-flag logic

```
      IF(STEADY) THEN
        U - U_spec = 0
      ELSE
        dU/dt - Udot = 0
      ENDIF
```

so that the Steady form is used for the appropriate variable even though STEADY = false. U_spec is prescribed explicitly versus the time (Fortran variable: TIME(.)). This code appears in SUBROUTINE GCON (in gssubs.f).

More details on time-marching calculations will be given later.

Selecting Constraints (mouse)

The constraints to be used are selected in the visual constraint editor invoked by the "T" command (or with the "%" command, described shortly). A grid of variables versus constraints is displayed, with a colored marker indicating which constraint is currently associated with each parameter. The markers are moved (i.e. the constraints are changed) by clicking on the grid. If a constraint is selected more than once, its markers are displayed in red, indicating this is a physically-impossible specification. Each operating point has such a constraint/variable table associated with it.

The steady or unsteady constraints can be toggled with a mouse click in the "steady/Unsteady" button. The control law routines can be imposed in lieu of the explicitly-set flap and engine constraints by clicking on the "fixed-stk/cntrl-Law/" toggle button.

Up to three constraint-setting configurations can be saved by clicking on the "to A", "to B", or "to C" buttons. They can then be retrieved by clicking on the corresponding "from" buttons. These A,B,C settings get saved to the xxx.set file when SSAV is issued at top level.

The four default buttons in the lower-left corner impose commonly-used constraint configurations. The action of most buttons can also be conveniently executed by typing a keyboard character rather than by clicking on the button, according to the following equivalence list.

Key	=	Button	
<space>		DONE	
D		DONE	
X		DONE	(automatically followed by eXecute in OPER)
Q		DONE	(automatically followed by Query in OPER)
+		+	
-		-	

```
    .    steady/unsteady  toggle
    L    fixed-stk/cntrl-Law  toggle
    B    Body/earth axes toggle

    F    Free defaults
    C    freqCalc defaults
    A    Anchored defaults
    S    Static defaults

    P    all Pt.
```

The "all Pt." button (or typing "P") applies the displayed settings
to all operating points on the stack, which can save considerable
typing and mouse input if the stack is rather long.

Selecting Constraints (keyboard)

The "%" command is equivalent to "T", except that it requests input from the keyboard rather than from mouse clicks. This allows ASWING
to be executed via a script file containing keyboard inputs.
Like with the "T" command, the constraint table plot is shown,
but this is followed by a keyboard input prompt:

```
 Type indices of (col,row) to toggle,  or  ( ABCDFLPS.+- ):
```

Instead of clicking on a box, the coordinates of the box to be toggled
are simply typed. For example, entering

```
  8 12
```

results in toggling of the box in the 8'th column from left, and the
12'th row down from the top. A column coordinate of 0 is equivalent
to clicking on the constraint column, which allows changing the a constraint's
numerical value. A row coordinate of 0 is equivalent to clicking on
a parameter symbol, which toggles it on or off. Any other column or
row index results in no action.

Alternatively, one can type any one of the special characters

which will have exactly the same action as with the "T" command.

Parameters Modification

Any one of the explicit parameters listed above can be changed for
the current target point with the corresponding command. Typing "A"
produces the altitude prompt:

```
 Enter flight altitude (Kft):   30.000
```

The number displayed after the prompt is the current value of the parameter,
and is also the default input if just <return> is entered. This default
convention is used for all keyboard inputs.

Constraint parameters, such as Alpha_spec, VIAS_spec, etc, can be
changed in a number of ways:

a) By clicking on the constraint field in the visual constraint editor
b) By using the "!" prefix with or without a parameter key and value

The second method is invoked from the OPER menu. Typing

```
  !V  60
```

will change the indicated airspeed to the new value of 60.

Typing

```
  !V
```

will request the same data with a prompt

```
 Enter specified  V_indicated      :    0.5000E+01  ft/s
```

which displays the current value which can be changed by entering
a new value. Just typing <return> retains the current value.
Typing just "!" will produce the full list of parameters which
can be changed:

```
   Constraint parameter  Key  Current value (point  1)
   --------------------  ---  ------------------------
     x accel.@xyzref      Ax     0.00000E+00  ft/s^2
     y accel.@xyzref      Ay     0.00000E+00  ft/s^2
     z accel.@xyzref      Az     0.00000E+00  ft/s^2
     x angular accel.     Gx     0.00000E+00  deg/s^2
```

```
     y angular accel.     Gy     0.00000E+00  deg/s^2
     z angular accel.     Gz     0.00000E+00  deg/s^2
     V_IAS   @xyzref      V       40.000      ft/s
     alpha   @xyzref      A      0.00000E+00  deg
     beta    @xyzref      B      0.00000E+00  deg
     x rotation rate      Wx     0.00000E+00  deg/s
     y rotation rate      Wy     0.00000E+00  deg/s
     z rotation rate      Wz     0.00000E+00  deg/s
     X_earth position     RX     0.00000E+00  ft
     Y_earth position     RY     0.00000E+00  ft
     Z_earth position     RZ     0.00000E+00  ft
     Bank angle           Ex     0.00000E+00  deg
     Elevation angle      Ey     0.00000E+00  deg
     Heading angle        Ez     0.00000E+00  deg
     Flap  1 deflect.     F1     0.00000E+00  Flap
     Flap  2 deflect.     F2     0.00000E+00  Flap
     Flap  3 deflect.     F3     0.00000E+00  Flap
     Flap  4 deflect.     F4     0.00000E+00  Flap
     Peng  1 setting      E1     0.00000E+00  Peng
     Peng  2 setting      E2     0.00000E+00  Peng
     net x Force          Fx     0.00000E+00  lb
     net y Force          Fy     0.00000E+00  lb
     net z Force          Fz     0.00000E+00  lb
     net x Mom@xyzref     Mx     0.00000E+00  lb-ft
     net y Mom@xyzref     My     0.00000E+00  lb-ft
     net z Mom@xyzref     Mz     0.00000E+00  lb-ft
     Lift @ N=1           L      0.00000E+00  lb
     Climb angle          Ga     0.00000E+00  deg
     Rad.accel.           Ar     0.00000E+00  ft/s^2

 ..Select  parameter key and value   c>
```

In this menu loop the entry conventions are the same as from OPER, but the "!" prefix is omitted. For example, typing

```
V 50
```

will change the specified VIAS from 40 to 50. This is the most convenient way to quickly change many parameters. Typing just <return> will exit back up to the OPER menu.

Note that there is a distinction between specified constraint parameters and state variables, e.g. V_spec is not quite the same as V. In a logical and ordered world, V would become V_spec only when a solution is computed with V - V_spec = 0 as one of the equations. This logical approach is tiresome to use, however, since often it is desirable to change ∗variables∗ without going through the

```
motions of recomputing the solution.  Typically this occurs when
the intial state of an unsteady solution is being specified.
Hence, when a new velocity is specified with the command

  !V "value"

the action is:

 set  V_spec = "value"

 also set  V = "value"   IF  direct constraint  V - V_spec = 0  is active


In practice, the both actions would occur in a typical steady case
where V is constrained directly.  Only the first would occur in
a typical unsteady case where V evolves according to its rate equation.

Each flight situation has an appropriate set of constraints and free
parameters to simulate it.  The visual constraint editor (commands "T" or "%")
shows all the constraints for one operating point at a time.  A summary
listing for all the points can be obtained with the "Q" command, which
lists which quantities are being imposed, and which are left free.

The Rad.accel. constraint imposes the condition
  .
  UY - UX WZ  =  Ar

With the specified parameter Ar set to zero, this becomes the constraint
between velocity, heading rate, and centripetal acceleration in
a steady level turn.  It is useful for determining control settings
required for a trimmed steady turn.



Each sensor parameter also has a corresponding specified value
which can be set with the "@" prefix.  These specified values
are used only in the user-supplied control routine as needed.
```

Parameters Frames of Reference

```
The rotation rates Wx,Wy,Wz, rotational accelerations Gx,Gy,Gz,
and linear accelerations Ax,Ay,Az, can be specified in the body axes
or in the Earth axes.  The latter is useful for specifying certain
flight conditions such as steady circling flight at some specified
```

circling rate. In earth axes, setting the rotation rate is trivial:

```
 WX = 0 ,  WY = 0 ,  WZ = turn_rate
```

The corresponding rotation rates Wx,Wy,Wz in body axes are not simple, since they involve the bank and elevation angles, which may not be known a priori if the airspeed and climb rate are also being set implicitly. The Body/Earth axis specification flag is toggled with the "Spec.Ax" button. Body axes are the start-up default.

2.5.4 Computation of an operating point

Once the appropriate parameters and constraints for an operating point are selected, that operating point can be executed, or converged, at any time with the "X" command. This solves all the structural and aerodynamic equations together with the specified constraints, and immediately displays the results. If the solution converges properly, solution plots can then be generated with the "S", "D", and "C" commands. The "C" command will show stress and strain only if the Cshell, Nshell, and Atshell parameter distributions have been specified across the span. If convergence is not achieved before the iteration limit is reached, typing "X" again will continue the iteration.

For time-marching computations, the "X" command can accept a filename argument for specifying open-loop forcing versus time. More details will be given later in the time-marching section.

2.5.5 Defining multiple operating points on stack

An additional operating point can be declared at any time from the OPER menu with the command "+", which inserts a new point into the stack immediately after the current target point. The inserted point will then become the new default target point. The new point inherits all settings and parameters from the previous target point. The current target point is removed from the stack with the "-" command.

Any point on the stack can be designated as the target point by typing its number.

Most of the OPER commands can be applied to ALL the points on the stack
by issuing a double command letter. For example, the "AA" command allows
specification of the same altitude for all the operating points on the stack.
Likewise, "XX" will then converge all the points, and "DD", "CC", etc. will
plot all of them overlaid.

Special case: "--" will remove all points EXCEPT the current target point.

Issuing a double parameter-change prefix "!!" will make the input parameter
value be applied to **all current points.** For example,

```
  !!V 60
```

will result in ALL operating points receiving a specified VIAS = 60 ft/s.
If the numerical argument is omitted, a prompt will be given...

```
  !!V

 Enter specified  V_IAS   @xyzref     :      40.000      ft/s
  50.0
```

This is a bit more reassuring since it gives some confirmation
of what's being changed.

2.5.6 Specifying complex flight conditions

In general, any well-posed set of parameters can be specified for
any operating point. The multiple points can be used to sweep over
a parameter of interest to discern its effect. For example,
aileron reversal speed can be deduced with the following sweep
in airspeed (listed with the "Q" command):

```
 #    VIAS    N        Lift   Alpha    Mx     Wx     Wy     Wz   Flap1 Flap2
--  ------  ----  ---------  ------  ------ ------ ------ ------  ----- -----
 1   200.0   1.0   0.400E+6             0.0           0.0    0.0  5.0   0.0
 2   220.0   1.0   0.400E+6             0.0           0.0    0.0  5.0   0.0
 3   240.0   1.0   0.400E+6             0.0           0.0    0.0  5.0   0.0
 4   250.0   1.0   0.400E+6             0.0           0.0    0.0  5.0   0.0
```

The constant aileron (Flap1) deflection, together with the net
aero+inertial rolling moment Mx being fixed at zero, will result
in a steady-state roll rate Wx being calculated. Initially, Wx will
increase linearly with VIAS, and then round over and eventually go

```
back to zero at the reversal speed.  At each point in this sweep,
the Lift is held fixed, so that the resulting calculated Alpha will
decrease as VIAS increases.  This is clearly more representative of
an actual flight situation than fixing Alpha and allowing the Lift
to increase with VIAS.
```

2.5.7 Option selection

```
The "K" command in the .OPER menu puts up the following sub-menu and prompt:

 ------------------------------------------------------
  I teration limit          :    10
  S tride for point plots   :     1

  F reeze VL matrices (fast)   :  T
  L ag aero effects modeled    :  T
  N ode plotting               :  F
  T erse listing               :  F
  G round reaction listing     :  F
  R ef. parameter listing      :  F
  A bsolute (earth XYZ) listing:  F
  E ngine info listing         :  F

  C oordinates for matrix    : stability axes
  M ach (for P-G)            :    0.000
  W ake core_radius/chord    :    0.2500

  X Y,Z ref. for moments      :    0.000       0.000       0.000     ft
                accelerations:    0.000       0.000       0.000     ft
                V,alpha,beta :    0.000       0.000       0.000     ft
  D elta(t) for listing       :    0.000       0.000       0.000       0.000

  V ariables listed ...
  B eams displayed on plot ...
  P arameters output on plot ...

..Change what?
```

```
This allows the various parameters listed above to be changed as necessary.
All these parameters are saved in the xxx.set file when SSAV is issued.
Below is a more detailed description of some of these parameters.
```

Iteration Limit - "I"

The "I" command allows changing the number of Newton iterations which are attempted to converge each operating point. Note that one can type "X" or "XX" to get another set of iterations, so there is little reason to make this too large. Of course, there is no opportunity to continue converging an unsteady point, but these rarely require more than three iterations anyway, at least in normal flight.

Plot stride - "S"

The plot stride limit is to prevent an excessive number of point solutions to be simultaneously displayed during time-marching calculations. This would slow down the time-marching unacceptably. If a stride of 20 is used, than every 20'th point or time snapshot is plotted.

Fast,Slow Vortex-Lattice Model - "F"

The "F" command toggles the "fast" and "slow" options. In the "fast" option, the VL influence matrices are held frozen at their values based on the undeformed-state geometry and zero alpha and beta. This is roughly equivalent to the treatment used by most standard panel and vortex lattice methods with fixed wake geometries. Here, this "fast" option may give considerable speedup in the calculation compared to the "slow" option which uses wind-aligned trailing vorticity. The aligning consists of recalculation of the influence matrices based on the latest geometry, alpha, beta (and perhaps Mach number -- see below) after every Newton iteration. This requires significant extra CPU time. The recalculation also prevents true quadratic convergence, and hence requires somewhat more Newton iterations for any given situation.

The "slow" option, which is physically more realistic, might be appropriate if extreme deformations and/or flight conditions are being computed, or if there is significant nearby interaction between surfaces and wakes, as in a canard configuration. For more routine geometries, or for rough estimates in general, the "fast" option is typically adequate.

Lag aero effects modeled - "L"

In ASWING, unsteady aero effects are modeled with an additional

```
wing downwash velocity proportional to the circulation rate
of change dGamma/dt (see Unsteady Extension theory document).
This additional downwash is critical to accurate prediction
of unsteady airloads and flutter.  However, sometimes it's useful
to disable it, which is the reason for having the "L" toggle.

Disabling the dGamma/dt contributions allows one to determine
the importance of unsteady aerodynamics in any given application.
One situation is when doing MODE analysis.  The presence of
the dGamma/dt terms will introduce a large number of fast
highly damped modes along the negative real axis.  The modes
consist of rapidly decaying wing circulation perturbations,
and hence are mostly aerodynamic in nature, but may complicate
a modal analysis application.  These modes will disappear
if L is toggled False, leaving only the classical flight dynamics
and structural modes.

This flag is not saved in the .set file, and is always set to
its default (True) setting everytime ASWING is started.  This
is to prevent the situation where this flag is turned off,
and left off for all subsequent analyses, possibly corrupting
a large amount of work.
```

Node plotting - "N"

```
This allows toggling between line and line+symbol plot styles.
The line+symbol style is useful since it visually conveys the
discretization density.
```

Terse listing - "T"

```
Terse listing consists of only the overall operating parameters,
which is convenient in some applications.  The alternative (verbose)
listing also includes variables along all the beams. (only acts when
typing "=" and results are printed to the terminal)
```

Ground-Reaction Listing - "G"

```
This allows the display of force and moment load jumps across the
ground location declared in the xxx.asw input file.  These should
be zero if the free-flight constraints on acceleration and angular
```

```
acceleration are selected, since here the ground is fictitious.
This output then serves merely as a check.  For anchored cases,
the ground reaction gives the load applied by the configuration
to the support at the ground location.
```

Reference-Parameter Listing - "R"

```
Relative flow angles, velocities, and moments inside ASWING
are referenced to the origin of the body axes.  However, they
are specified at possibly some other reference locations
given in the "Reference" block of the input file.
The transformation of origin-referenced velocities U_o
and moments M_o to some other reference location r_ref is

 U_ref  =  U_o  +  Omega x r_ref
 M_ref  =  M_o  +  r_ref x F

The reference speed |U_ref| and reference flow angles
alpha_ref, beta_ref are defined from the U_ref components.
The "R" option toggles the listing of all these
reference-location quantities.
```

Absolute (earth) axes Vector Listing - "A"

```
This toggle enables the listing of various vectors in Earth axes
in addition to the usual body-axes listing.  In some situations,
the Earth-axes components are important, e.g. U_Z is the
aircraft climb or sink rate.
```

Engine info listing - "E"

```
This toggle enables the listing of engine-model quantities
in greater detail.
```

Coordinates for matrix - "C"

```
The stability and control derivatives generated with OPER's "M"
command can be output in a number ox axis systems.  Typing "C"
here lists the options and the input prompt:
```

```
  0  body axes
  1  stability axes:  x fwd, z down, (Etkin standard)
 -1  rev.stab. axes:  x aft, z up
  2  wind axes:  x windward, z up

 Enter new  coordinate system type:
```

Prandtl-Glauert Mach number - "M"

```
There are two ways with which the Prandtl-Glauert Mach number
can be defined.

 i)  M = M_PGspec
ii)  M = V/a

The type of definition is toggled with the "M" command.  Type i) simply
fixes the Mach number at the specified value (0.000 is shown above).
Type ii) defines the Mach number from the current true airspeed and
the atmospheric speed of sound.  Type i) should be used if possible,
since changing the Mach number requires recomputation of the vortex-lattice
AIC matrix each Newton iteration.  Normally, having the Prandtl-Glauert
Mach slightly "wrong" has little effect on the results, unless the
case is in the high subsonic range.  Using i) with M_PGspec = 0.0
is generally adequate for flight Mach numbers below 0.3 or so.

Note that local subsubsection Cm properties are scaled by the local
Prandtl-Glauert factor 1/sqrt(1 - Mn^2) where Mn is the Mach number
normal to the surface beam axis.  This is always computed, and is
independent of the Prandtl-Glauert Mach number described here.
```

Wake Vortex Core Radius - "W"

```
Each horseshoe vortex has a "core radius" so that it does not have
a singular velocity.  This is essential for stable calculations
with multiple interacting surfaces.  The "W" command allows
adjusting the core size relative to the chord where it is attached.
A value of r/c = 0.25 is recommended, although this can be varied
a bit to check its effect.  All of this is only an issue if a wake
impinges directly on a downstream surface or comes very close.
In the absence of close surface/wake encounters, the core size
has very little effect.  For single-surface configurations,
```

the core size has no effect at all.

For coarse node spacings, the core size may actually be determined from the local spacing (horseshoe vortex width) rather than the specified r/c value. The actual radius used is given by

```
r_core = max( (r/c)*chord , interval_width )
```

as described in the theory document. The reason for involving the interval width is to mostly eliminate non-physical discreteness in the wake's induced velocity field due to the individual filaments. Note that r/c has no effect for values below interval_width/chord. Hence, if it is necessary to use a small core size, sufficiently dense node spacing must be used (adjust in the NODE menu).

Displayed Variables - "V", "D"

The "V" command puts up a sub-sub-menu:

```
 1    x   !
 2 *  y   ! position in aircraft x,y,z axes
 3 *  z   !
 4    phi    !  dihedral Euler angle
 5 *  theta  !  twist    Euler angle
 6    psi    ! -sweep    Euler angle
 7 *  Mc'  !
 8 *  Ms'  ! resultant c,s,n moments
 9    Mn'  !
10    Fc    !
11    Fs    ! resultant c,s,n forces
12 *  Fn    !
13    ux     !
14    uy     ! velocities
15    uz     !
16    wc       !
17    ws       ! rotation rates
18    wn       !
19    e_max       ! maximum extensional strain
20    tau         ! maximum shear stress
21    chord         ! surface chord
22 *  cl              ! local cl_perpendicular
23    cm              ! local cm
24    f_aero            ! see below
25    f_reac            ! see below
26    da_eff            ! see below
```

```
 ...Select variable(s) to toggle:
```

which allows toggling of the variables (marked by "∗") to be listed versus span in the screen ASCII output. The variables are interpolated in t for each beam in increments displayed after the "D" command above. An increment of 0.0 indicates that the actual structural node values are to be output without any interpolation.

The f_aero variable is the overall aerodynamic lift-force/span normal to the wing, which reduces to

```
   f_aero  =  rho V Gamma  +  rho c Gamma_dot    (in n direction)
```

for a simple wing without swept flow. The second term is omitted if the unsteady aero lag terms are disabled with the L toggle command.

The f_reac variable includes gravity, acceleration, and apparent-mass forces in the normal direction. For a flat wing with only vertical acceleration this reduces to

```
   f_reac  =  mu(g - a)  -  rho pi (c/2)^2 a     (in n direction)
```

where g is the gravity vector (normally in the -z direction). The Unsteady Extension theory document gives the complete expressions for these applied forces.

The da_eff variable is the net effective angle of attack change as a result of elastic deformation. It is defined as

```
   da_eff  =  (n - n0) . V_tot / |V_tot|       (displayed in degrees)
```

where n is the surface-normal vector, n0 is the jig surface-normal vector, and V_tot is the total surface-relative velocity vector resulting from aircraft motion, deflection velocities, gusts, and induced velocities. The da_eff variable is useful in gauging the elasticity-induced change in the normal vector, n - n0, which must be compensated with a change in the circulation to maintain flow tangency.

Discrete Variable location

In the "Displayed Variables" list above, the first 18 beam variables

```
   1    x
```

```
   2    y
.
.
  17    ws
  18    wn
```

are state variables, which are located at the beam x,y,z nodes.
The remaining variables are derived quantities, and are located
at the midpoints between the nodes.

```
  19    e_max
  20    tau
.
.
  24    f_aero
  25    f_reac
  26    da_eff
```

For the interest of screen space, and to make the output easier to parse
by an external analysis program, the derived quantities are shifted
upward by 1/2 interval. So output which should strictly appear as

```
   Fuselage Beam  1:  Body
                                        2      -9       1
       t      y      z-z0   tw-tw0 Mc'/10  Ms'/10   Fn/10     cl   f_aero
   ------- ------- ------- ------- ------- ------- ------- ------- -------
      0.00    0.00    0.00    0.00  -2.073   0.000  -1.917
                                                            0.3041   0.033
      0.25    0.00    0.00    0.00  -1.979   0.000  -1.872
                                                            0.2621   0.083
      0.50    0.00    0.00    0.00  -1.886   0.000  -1.836
                                                            0.2207   0.110
      0.75    0.00    0.00    0.00  -1.795   0.000  -1.805
                                                            0.1802   0.117
      1.00    0.00    0.00    0.00  -1.705   0.000  -1.775
                                                            0.1608   0.115
      1.00    0.00    0.00    0.00  -1.705   0.000  -1.775
                                                            0.1402   0.108
      1.25    0.00    0.00    0.00  -1.617   0.000  -1.744
```

will actually be printed as follows.

```
   Fuselage Beam  1:  Body
                                        2      -9       1
```

t	y	z-z0	tw-tw0	Mc'/10	Ms'/10	Fn/10	cl	f_aero
0.00	0.00	0.00	0.00	-2.073	0.000	-1.917	0.3041	0.033
0.25	0.00	0.00	0.00	-1.979	0.000	-1.872	0.2621	0.083
0.50	0.00	0.00	0.00	-1.886	0.000	-1.836	0.2207	0.110
0.75	0.00	0.00	0.00	-1.795	0.000	-1.805	0.1802	0.117
1.00	0.00	0.00	0.00	-1.705	0.000	-1.775	0.1608	0.115
1.00	0.00	0.00	0.00	-1.705	0.000	-1.775	0.1402	0.108
1.25	0.00	0.00	0.00	-1.617	0.000	-1.744	0.0000	0.000

Note that the derived variables, or "cl" and "f_aero" in this case,
are shifted up by 1/2 line, and dummy zeros are put in the last line.

Plotted Parameters,Beams - "P", "B"

Similarly, the "P" command allows toggling of the parameters to be plotted
in the table appearing above the spanwise loading plot. Individual beams
can be "hidden" in the plots via the "B" command to reduce plot clutter.

The "P" command also allows toggling of sensor parameters for plotting
in the table form. Sensor parameter names are indicated by a sensor
index in brackets. For example, the aircraft's global angle of attack
at (x,y,z)_ref would be labeled as simply

```
 alpha  or  alpha_ref
```

while the angle of attack delivered by sensor 3 would be labeled as

```
 alpha [ 3]
```

This convention is used in all output.

Other Options

Atmospheric density, gravity

The "A" command allows changing of the ambient atmosphere density
from a Standard Atmosphere by specification of an altitude.
This command also allows changing of the gravity acceleration.
These changed quantities over-ride their values given in the input file.

Ground effect

```
The "G" command allows selection of ground effect from its sub-menu:

   0  no ground image
   1  solid ground
  -1  free surface

  Enter image flag:


The ground effect is implemented via the usual method of images,
described in the theory document.  The default flag is 0 ,
which specifies that there's no image (i.e. no ground effect).
A flag of +1 indicates the usual solid-ground image, and
a flag of -1 indicates an anti-image which models a free surface.

The image plane is assumed to contain the Earth-axes origin XYZ = (0,0,0).
The orientation of the image plane is specified by its normal vector
in XYZ axes, which can be given by the second sub-menu of the "G" command:

  Enter ground-normal vector: 0.000 0.000 1.000

The default (0,0,1) vector in the prompt corresponds to level ground.

The position of the aircraft relative to the ground image plane
is given by its Earth XYZ coordinates.  These can be changed
in the constraint table with command "T", or with the
!RX , !RY , and !RZ commands.  It is essential that the
Z position and aircraft Euler angles be such that all parts
of the aircraft are above the ground plane, otherwise the solution
will be nonsense.  The velocity field with ground image effects
can be viewed in the PLOT menu from Top Level, "T" command.

A separate image flag is defined for each operating point.
To change the flag for all the defined points, use the doubled
"GG" command as usual.

In contrast, the ground normal vector is common to all operating points.
```

2.6 Time-Marching Calculation

2.6.1 Overview

A time-marching calculation is generated in OPER by toggling to the unsteady mode with "." (or in the visual constraint editor), and then issuing the "X" command. This produces the prompt

```
 Enter delta(t), Nstep  (-Nstep to append steps)
```

which requests the time step interval and the number of time steps to be generated. The number of time steps is limited by the array dimension NPNTX declared in file DIMEN.INC, and can be increased if necessary. The current target point is used as the initial state, and is stored in slot 1 (if Nstep > 0), or is kept in its current slot (if Nstep < 0), and the subsequent N states are calculated and stored in the operating point arrays. Each time snapshot can then be examined by selecting it as the target point. For example, choosing

```
 Enter delta(t), Nstep  (-Nstep to append steps)
 0.1 50
```

will result a time history spanning 0.1 x 50 = 5 seconds (or whatever time unit is being used). Once the time-march calculation finishes, choosing the time index, e.g.

```
 .OPER (target point:  1)   c>  21
```

will allow the 21'st state at t = 0.1 x (21-1) = 2 seconds to be examined in the various plots and listings. Alternatively, repeated commands such as "SS" and "CC" will overlay distributions for the entire time history, allowing maximum bending moment or strain over the unsteady maneuver to be easily discerned, for example. The "WW" command will dump all spanwise distributions for the entire time history to a disk file for external digestion.

2.6.2 Open-Loop forcing during time march (*** NEW for v 5.41 ***)

The "X" command can accept a filename argument:

```
  .OPER (target point:  1)   c>  X filename
```

The file is expected to have the following format:

```
#
 time    param1    param2  . . .
#
* 1.      1.        1.
+ 0.      0.        0.
#
 0.0      1.0       0.0
 0.1      1.0       0.0
 0.1      1.0       5.0
 0.2      2.0       5.0
 0.3      3.0       5.0
  .
  .
```

Blank lines and lines beginning with "#" or "!" or "%" are ignored. The labels "param1", "param2"... are chosen from the column below. These labels are assigned to the CKEY(*) array, in SUBROUTINE INIT (src/init.f).

	param
x accel.@xyzref	Ax
y accel.@xyzref	Ay
z accel.@xyzref	Az
x angular accel.	Gx
y angular accel.	Gy
z angular accel.	Gz
V_IAS @xyzref	V
beta @xyzref	B
alpha @xyzref	A
x rotation rate	Wx
y rotation rate	Wy
z rotation rate	Wz
X_earth position	RX
Y_earth position	RY
Z_earth position	RZ
Bank angle	Ex
Elevation angle	Ey
Heading angle	Ez
Flap 1 deflect.	F1
Flap 2 deflect.	F2
Flap 3 deflect.	F3
Flap 4 deflect.	F4
Flap 5 deflect.	F5
Flap 6 deflect.	F6
Peng 1 setting	E1
Peng 2 setting	E2

```
     net x Force          Fx
     net y Force          Fy
     net z Force          Fz
     net x Mom@xyzref     Mx
     net y Mom@xyzref     My
     net z Mom@xyzref     Mz
     Lift @ N=1           L
     Climb angle          Ga
     Rad.accel.           Ar
```

These are the same parameters that can be specified manually with the "!" prefix in OPER. In effect, the parameter file saves a lot of manual typing of parameter values as the calculation proceeds. The file can also contain multiplier and adder lines anywhere after the first parameter name line.

In addition to the parameters listed above, it is also possible to specify commanded sensor parameters, which would typically be used in the control law routines. A sensor parameter is indicated by a suffix of brackets enclosing the sensor index. The sensor parameters are chosen from the following list:

```
                     param
                     -----
     x accel          Ax
     y accel          Ay
     z accel          Az
     x ang.accel      Gx
     y ang.accel      Gy
     z ang.accel      Gz
     V_IAS            V
     beta             B
     alpha            A
     x rot. rate      Wx
     y rot. rate      Wy
     z rot. rate      Wz
     X_earth pos.     RX
     Y_earth pos.     RY
     Z_earth pos.     RZ
     U_earth vel.     UX
     V_earth vel.     UY
     W_earth vel.     UZ
     Bank             Ex
     Elevation        Ey
     Heading          Ez
     c Force          Fc
     s Force          Fs
```

```
     n Force            Fn
     c Moment           Mc
     s Moment           Ms
     n Moment           Mn
     Circulation        Cr
```

An example file showing the format is given shortly.

Each parameter is splined versus time, and the value taken
from the spline is imposed explicitly during the time marching
calculation. Two consecutive time values will result in
separate splines on each side of the doubled point,
allowing a discontinuity in the value and/or slope of
the parameter. In the example above, param1 has a slope
break and param2 has a discontinuity at time=0.1

Often it is convenient to specify parameter increments
to their initial values (e.g. trim values). This is
indicating by prefixing the name of each such parameter
with a "+" in the first line. For example,

```
 time     F1        +F2     Az[2]
 0.0      1.0       0.0      0.0
 0.1      1.0       0.0      2.0
 0.1      1.0       5.0      2.0
 0.2      2.0       5.0      3.0
 0.3      3.0       5.0      4.0
```

would indicate that **the F2 data is to be added to
the initial F2 value** to form the actual specified values.
The 4th column gives the commanded acceleration Az_c,
associated with the control law for sensor 2.

For open-loop calculations, the param values will be imposed
explicitly. **It must be remembered that a specified parameter
will be ignored if that parameter's constraint is not active**
(i.e. it is not selected in the "T" table). For typical unsteady
open-loop cases, specifying only the flap and engine parameters
F1,F2..P2 is physically meaningful. The other parameters simply
do not appear in any of the unsteady constraints.

For closed-loop calculations, the control-law can also act
on other specified quantities such as airspeed or bank angle,
or any of the sensor-related quantities, so these can affect
the simulation. All the specified parameters will be received
by the control-law routine which can then do whatever

```
it wants with them.
```

2.6.3 Detailed display of time-marched calculation

```
In the Steady mode, the "V" command will allow the deformed geometry
of the target point (or time snapshot) state to be viewed in perspective.
In the Unsteady mode, the "V" command will produce a "movie" of the time
history, in real time if possible.

The viewing angle for the movie can be set first with the "V" command
in the Steady mode to get a suitable viewpoint.  The "O" command allows
a number of options to be changed to alter the view and movie display.
```

2.6.4 Time-marching calculation procedures

```
A time-marching calculation can simulate the dynamics of a fixed aircraft
(e.g. model bolted down in wind tunnel), or the dynamics of a free-flying
aircraft.  In each case, the aircraft's reference Earth coordinates
RX,RY,RZ evolve according to

 d   RX     UX    [    ]  Ux
 --  RY  =  UY  = [ Te ]  Uy
 dt  RZ     UZ    [    ]  Uz

where UX,UY,UZ are the earth-axis components and Ux,Uy,Uz are the body-axis
components of the aircraft's velocity.  The latter are simply opposite to
the usual relative freestream velocity vector

              Ux        -cos(a) cos(b)
 -Vinfinity = Uy  =  V          sin(b)    ,  a = alpha,  b = beta
              Uz        -sin(a) cos(b)

and V is the freestream speed.  The Euler angles and hence Te
also evolve via the rotation relation

 d   E1      [  -1][    ]  Wx
 --  E2   =  [ Ke ][ Te ]  Wy
 dt  E3      [    ][    ]  Wz

where Wx,Wy,Wz are the aircraft rotation rates, and the Ke matrix,
also a function of the Euler angles, relates the rotation rates
to the Euler angle rates.  The velocities Ux,Uy,Uz and rotation
rates Wx,Wy,Wz in turn evolve according to
```

```
 d   Ux     ax
 --  Uy  =  ay  - W x U
 dt  Uz     az

 d   Wx     Ax
 --  Wy  =  Ay
 dt  Wz     Az
```

where ax,ay,az and Ax,Ay,Az are the absolute (inertial-frame) linear
and angular accelerations. These can be either set explicitly
(typically to zero) to simulate a bolted-down aircraft:

```
   .  .  .             .  .  .
   Wx Wy Wz  Mx My Mz  Ux Uy Uz  Fx Fy Fz
   -- -- --  -- -- --  -- -- --  -- -- --
   *  *  *             *  *  *
```

or implicitly via force and moment equilibrium to simulate free flight:

```
   .  .  .             .  .  .
   Wx Wy Wz  Mx My Mz  Ux Uy Uz  Fx Fy Fz
   -- -- --  -- -- --  -- -- --  -- -- --
             *  *  *             *  *  *
```

The visual constraint editor displays all these constraints, which
should be checked before the time-marching calculation is generated.

2.6.5 Displaying time sequences

Examining each snapshot individually as described above provides
a wealth of information on the aircraft states, but is overwhelming
if only the overall behavior is of interest. Time trace plots of
key parameters are a concise way to view the time-march sequence.

The command "P" puts up the time-trace plot menu:

```
 ---------------------------------
   S elect parameters
   P lot parameter time traces
   L ist parameter time traces
   W rite parameters to file
   D ata file overlay toggle

 ..Select option   c>
```

```
Parameters which are to be plotted are toggled with the "S" option.
This includes the global parameter as well as the sensor parameters.
The P,L,W options then act only on the selected parameters.
The D option will overlay disk data written previously with W.
```

2.6.6 Running long time sequences

```
The maximum number of time snapshots (points on stack) is set
by the NPNTX dimension parameter.  This is necessarily limited
by the relatively large amount of storage needed to store the
full state vector at each time snapshot.  However, if only
a history of the parameters toggled in the "P" sub-menu is required,
arbitrarily long time sequences can be easily computed as follows.

 1) compute a time sequence up to the array limit NPNTX or less
 2) save whatever is needed from the sequence with
      the "W" command in the "P" sub-menu
 3) make sure current point is the last point in the sequence
      (this will normally be the case)

 1) compute new sequence with "X", using positive number of steps
 2)  ...
 3)  ...

Repeating the 1),2),3) steps and specifying "Append" for the
disk file output for step 2), will result in the data file
containing the total time sequence.
```

2.7 Additional Content for Time-Marching Simulations

2.7.1 ASWING Closed-Loop Control

```
Control methods can be run in ASWING time-transient simulations by
loading a control law file (XXX.con), and by specifying the solver
to use the control file as closed-loop control law when completing
the constraint table of the OPER menu.

The file specifies the various gain-scheduled parameters for the
controllers, linking one input to an output through a (scheduled)
gain g(par1,par2):

    output = g(par1,par2) * (input) + trimmed_output

The layout of the XXX.con file is organized as follows:
```

```
Each g_ij coefficient can be scheduled with up to
two operating parameters,

    g_ij = g_ij(par1,par2)

evaluated via bilinear interpolation from a lookup table.
The lookup table data is read from file FNAME (xxx.con),
and placed in the common blocks in CONLAW.INC.

The file has an arbitrary number of table blocks, each block
defining the table for one g_ij function.

Each block has the following format.  Blank lines, and lines
beginning with ! or # or $ are ignored.

    First line: which output, which input, which sensor (0 body
        frame), scale factor
    Second line: How many values and how many scheduled variables
        (in general V and rho, can be more)
    Third line: value for variable 1 (velocity)
    Fourth line: value for variable 2 (rho)
    Fifth line: Gain values from g(1,1), to g(np1,1)
     ...
    Nth line: Gain values from g(1,np2), to g(np1,np2)

A general file example is provided below

C#-------------------------------------------------------
i  j  s  gfac          ! identifies which g_ij function is being defined
C
np1  np2               ! number of scheduling-parameter values
C
par1(1) par1(2) par1(3) . . .  par1(np1)           ! parameter 1 values
par2(1) par2(2) par2(3) . . .         par2(np2)    ! parameter 2 values
C
g(1,1)   g(2,1)   g(3,1)   g(4,1)   . . .  g(np1,1)   ! table data
g(1,2)   g(2,2)   g(3,2)   g(4,2)   . . .  g(np1,2)
g(1,3)   g(2,3)   g(3,3)   g(4,3)   . . .  g(np1,3)
.        .        .        .        . . .  .
.        .        .        .        . . .  .
.        .        .        .        . . .  .
g(1,np2) g(2,np2) g(3,np2) g(4,np2) . . .  g(np1,np2)
C#-------------------------------------------------------
```

```
The quantities are defined as follows.
See SUBR.CONTROL header for i values to use for KD*
See INDEXC.INC for j values to use for KU*, KQ*

  i = 1..NDDIM   control variable D index KD*
  j = 1..NUDIM   state   variable U index KU*  (if s = 0)
  j = 1..NQDIM   sensor  variable Q index KQ*  (if s > 0)
  s = 1..NSEN    sensor index
  gfac = scale factor for all g values to follow

  np1 = dimension 1 of table (number of par1 values)
  np2 = dimension 2 of table (number of par2 values)

  par1(.)  par1 table values
  par2(.)  par2 table values

  g(..)    gain values

C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Two practical controller examples are shown below


  Table block example 1:
  Set gain for yaw damper, scheduled on true airspeed V.
  ( SUBR.CONTROL currently defines  par1 = V )
C
3  6  0  1.0           ! Delta(3) is rudder,  KUROTZ = 6 is Omega_z index
5  1                   ! Five velocities will define g(V) scheduled gain
30. 40. 50. 70. 90.    ! par1 values are velocities
0.0                    ! par2 value doesn't matter here (since np2 = 1)
1.00 0.87 0.77 0.65 0.58  ! gain values corresponding to par1 values
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Table block example 2:
  Set constant (unscheduled) gain for phugoid damper
C
2  8  0  1.0           ! Delta(2) is elevator,  KUELEV = 8 is Theta index
1  1                   ! no scheduling... use only one par1,par2 value each
0.0                    ! par1 value, doesn't matter if np1 = 1
0.0                    ! par2 value, doesn't matter if np2 = 1
0.4                    ! gain value
```

2.7.2 Gust Velocities

ASWING permits arbitrary gust velocities to be specified for both
quasi-steady and time-marching calculations. The gust consists
of a velocity vector field Vgust(X,Y,Z) embedded in the otherwise
still atmosphere. This vector function is computed in the user-supplied
SUBROUTINE VGUSTE (in vgust.f) which returns the gust X,Y,Z velocity
components as functions of the Earth coordinates X,Y,Z. The value of
X,Y,Z at some aircraft location x,y,z is

```
  X       RX      [    ]  x
  Y   =   RY   +  [ Te ]  y
  Z       RZ      [    ]  z
```

where RX,RY,RZ is the position of the aircraft's origin, and
Te(E1,E2,E3) is a transformation tensor depending on the aircraft's
(E1,E2,E3) = (bank,elevation,heading) Euler angles. The earth coordinates
X,Y,Z coincide with the body axes x,y,z for the case of zero E1,E2,E3
and at zero time, since in this case RX = RY = RZ = 0, and Te = I.
Hence, it is helpful to think of X,Y,Z as overlaying x,y,z when
the gust field is defined.

As time t increases, RX,RY,RZ and E1,E2,E3 all evolve, and so X,Y,Z
(passed in the XYZ(3) array) at each aircraft location changes,
resulting in the apparent gust velocity Vgust(X,Y,Z) changing as well.
Typically, the gust is defined "upstream" at negative X values so that
the aircraft will fly into it.

The total velocity seen by a point on the aircraft at location r is

```
 Vtotal(r)  =  Vinfinity  +  Vgust(r)  +  Vinduced(r)  -  W x r  -  u
```

where W is the aircraft rotation rate and u is the local beam
deformation velocity relative to the aircraft axes. Note that
a uniform gust velocity field is equivalent to changing Vinfinity
by the same amount. The definitions of CL, CD, etc, use the
velocity relative to the still atmosphere, so these may appear
to have unusual values. The physical lift and drag will still
be correct, however.

SUBROUTINE VGUSTE computes the gust velocity field using the
user-defined parameters passed to it via COMMON/COM_VG/ .
These can be initialized or read from a file in SUBROUTINE VGINIT,
which is called once at program startup. Another file can be read
interactively with the "GGET" command, and the parameters can
be interactively modified via the "GMOD" command. These commands

```
simply invoke the simple subroutines VGPGET and VGMODI which
actually read the file or allow parameter input.  Routines VGINIT,
VGMODI, and VGUSTE (all in vgust.f) can be modified as needed.

The gust field can be plotted on any slice plane in the PLOT menu
using command "G".
```

2.8 Distribution Variable Editing

```
The EDIT routine is useful for quickly changing the Distribution variables
without the need to regenerate the input file.  The EDIT menu is invoked
with the EDIT command:

 ----------------------------------
   B# number of beam     to edit
   #  number of variable to edit
   L ist variable numbers, beams

   S cale by constant
   A dd constant
   M odify via cursor
   T angent-endpoints toggle
  sY mmetry toggle

   I nsert point
   D elete point
   C orner add (double point)

   W rite current variable to file
   O verlay file-variable toggle

   Z oom
   U nzoom, refresh
   R efresh plot

   E psilon/tau plot toggle
   J ot on plot
   H ardcopy current plot
 ----------------------------------

 .EDIT (beam  2, variable  1)   c>


The L command gives a list of the beams and variables which can be modified,

    0  Fuselage
```

```
    1  Wing
    2  Right Stab
   -2  Left Stab
    3  Right Rudder
   -3  Left Rudder

     0  s                 21  Dmg               42  dCLda
     1  x                 22  DCcg              43  dCLdF1
     2  y                 23  DNcg              44  dCLdF2
     3  z                 24  Cea               45  dCLdF3
     4  twist             25  Nea               46  dCLdF4
     5  EIcc              26  Cta               47  dCLdF5
     6  EInn              27  Nta               48  dCLdF6
     7  EIcn              28  tdeps             49  dCMdF1
     8  EIcs              29  tdgam             50  dCMdF2
     9  EIsn              30  Cshell            51  dCMdF3
    10  GJ                31  Nshell            52  dCMdF4
    11  EA                32  Atshell           53  dCMdF5
    12  GKc               33  radius            54  dCMdF6
    13  GKn               34  Cdf               55  dCDdF1
    14  mgcc              35  Cdp               56  dCDdF2
    15  mgnn              36  chord             57  dCDdF3
    16  mg                37  Xax               58  dCDdF4
    17  Ccg               38  alpha             59  dCDdF5
    18  Ncg               39  Cm                60  dCDdF6
    19  Dmgcc             40  CLmax             61
    20  Dmgnn             41  CLmin             62

 .EDIT (beam  1, variable  1)   c>
```

which are simply all the Distribution variables listed earlier, indexed by integer. The s(t) variable is also included, but only for inspection. It cannot be edited since it is derived from the x,y,z(t) distributions. A variable is selected by typing its integer. A beam is selected by prefixing its beam number with "B". For example, the command

```
 B -2
```

would select the Left Stab as the beam to be modified. The command

```
 39
```

would then select Cm as the particular variable to be modified.

Once selected, a beam variable can be modified via the S,A,M,I,D commands. The M command allows a new distribution to be "drawn" by inputting new points on the window over any part of the distribution. The input points

```
are sorted, splined, and the resulting curve is grafted into the existing
curve between the leftmost and rightmost clicked points.  The endpoints
of the grafted curve can be made tangent to the existing curve if the
tangency flag is set (toggled via the T command).

As described earlier, a two identical t coordinates indicate a "corner",
which allows a discontinuity in the variable and/or its derivative.
A corner can be set at any existing point with the C command, and can
be removed with the D command.

When the EDIT menu is exited and some changes have been made, the modified
distribution variables are applied to all defined operating points,
and all the operating points are marked as unconverged.  The operating
points can then be immediately reconverged from the OPER menu.
```

2.9 Forced-Response Frequency Analysis – BODE

2.9.1 Overview

```
The frequency-response facility is invoked with the BODE command,
which produces the following prompt and menu:

 ---------------------------------------------------------------
  T oggle constraints (% for keybd) A dditional points in range
  N ew frequency range              E xtend frequency range

  G ain  plot                       F orcing  parameter select
  P hase plot                       R esponse parameter select
  !  broken-loop plotting toggle    I mpose gust modes

 eX amine selected frequency
  V iew of geometry response        M ovie (real-time view)

  L oading response plot            H ardcopy current plot
  S tructural response plot         J ot annotations on plot
  C urvature-strain response plot   Z oom  (B for no distort)
                                    U nzoom

  W rite Bode data to file          O ptions for plots
  D ata file overlay toggle         K ontrol toggles, settings

  #  point number select            Q uery point specification
 ---------------------------------------------------------------

 .BODE (point:  1 ,  forcing by: Flap01 )   c>
```

This menu allows calculation of linearized dynamic perturbation
solutions of the form

$$dx(t) = \bar{x}\, \exp(iwt) \quad , \quad x(t) = x\_base + Real[dx(t)]$$

taken about the aircraft state x_base corresponding to the current
(presumably converged) operating point. The perturbation dx is assumed
to be small, but x_base could have very large deformations.

The forcing is due to flaps, engine power settings, or gust field variables.

$$dFlap = \bar{F}\, \exp(iwt) \quad , \quad dPeng = \bar{P}\, \exp(iwt) \quad , \quad dGk = \bar{Gk}\, \exp(iwt)$$

The dFlap and dPeng variables are present only if defined in the xxx.asw file
via the dCLdF* variables, or in the Engine data block. The dGk variables
are enabled by issuing the "I" command here. The theory document aswu.ps
has more details on the gust perturbation formulation.

The first step in computing response solutions is to set the appropriate
constraint flags via the visual constrain editor invoked by the "T" command.
The appropriate constraints must correspond to the flight situation
being considered. These are likely to be *different* than the constraints
which were used to compute the operating point state. For example,
alpha could have been determined with the static specified-lift constraint
L = L_spec , but this constraint is probably inappropriate for a real
dynamical situation where lift is not fixed in time. Typically, the
frequency-response analysis constraints are the same as those which
would be used for a time-marching calculation. The Unsteady mode is
always be assumed in BODE, and it is not necessary to toggle it explicitly.

Command "N" computes response solutions for all forcing variables which
are defined. The forcing variable which is displayed is indicated
in the prompt. It is changed with the "F" command.

Once a frequency range is computed, additional points can be computed
with the command "A" to put more detail in the Bode and Phase plots.
The range can also be extended with command "E".

2.9.2 Bode response selection and plotting

- - - - - - - - - - - - - - - - - - -
The response variables to be plotted are selected via the "R" command.
This prints the following variable-selection menu and prompt.

```
   1   Phi   (bank)           20 * V
   2 * Theta (elev)           21   beta
   3   Psi   (head)           22 * alpha
   4   Flap 1                 23   VIAS_ref
   5   Flap 2                 24   beta_ref
   6   Flap 3                 25   alpha_ref
   7   Flap 4                 26   Lift
   8   Flap 5                 27   Drag
   9   Flap 6                 28   CL
  10   Peng 1                 29   CD
  11   Peng 2                 30   CDi
  12   Peng 3                 31   CM
  13   Peng 4                 32   span eff.
  14   Int(Vi-Vic)dt
  15   Int(be-bec)dt
  16   Int(al-alc)dt
  17   Int(Ph-Phc)dt
  18   Int(Th-Thc)dt
  19   Int(Ps-Psc)dt

  41   dUx lin.acc.           51   dWx ang.acc.
  42   dUy lin.acc.           52   dWy ang.acc.
  43   dUz lin.acc.           53   dWz ang.acc.
  61   Ux velocity            71   Wx rot.rate
  62   Uy velocity            72   Wy rot.rate
  63   Uz velocity            73   Wz rot.rate
  81   RX position            91   Int(Wx-Wxc)dt
  82   RY position            92   Int(Wy-Wyc)dt
  83   RZ position            93   Int(Wz-Wzc)dt
 101   Int(a-g-ac)dt         111   Xcg position
 102   Int(a-g-ac)dt         112 * Ycg position
 103   Int(a-g-ac)dt         113   Zcg position

 Select parameter(s) to toggle:
```

Each variable is toggled on/off by specifying its number from the list.
More than one number can be given to the prompt. In the example above,
variables 2, 20, 22, 112 are already toggle ON for plotting.
If there are no more changes, just typing <Enter> will exit the menu.

If there are any sensors present, a second sensor-variable selection menu
will then be printed:

```
      Sensors..
      1  2  3  4
   1              Phi
```

```
 2               Theta
 3               Psi
 4               V
 5               beta
 6               alpha
 7               Vi-Vic dt
 8               be-bec dt
 9               al-alc dt
10               Ph-Phc dt
11               Th-Thc dt
12               Ps-Psc dt
13               Gamma
14               dUx Lacc
15               dUy Lacc
16  *        *   dUz Lacc
17               dWx Aacc
18               dWy Aacc
19               dWz Aacc
20               Wx rate
21               Wy rate
22               Wz rate
23               UX vel
24               UY vel
25  *        *   UZ vel
26               RX pos
27               RY pos
28               RZ pos
29               Fc  for
30               Fs  for
31               Fn  for
32               Mc‘ mom
33               Ms‘ mom
34               Mn‘ mom
35               rx pos
36               ry pos
37               rz pos
38               ux vel
39               uy vel
40               uz vel
41               Wx-Wxc dt
42               Wy-Wyc dt
43               Wz-Wzc dt
44               ax-axc dt
45               ay-ayc dt
46               az-azc dt

Select  par.index, sens.indices  to toggle (? = help):
```

```
Typing "?" will show the various toggling command syntaxes available:


  Command line:    Ipar Isens Isens Isens ...

  Special cases:  Ipar  = 0   apply to all parameters
                  Isens = 0   apply to all sensors
                  Isens = -1  turn off for all sensors


For example, typing  either "5 1 2 3 4" or "5 0" will toggle
variable 5 (beta)  for all four sensors present.
Typing "0 -1" will turn all variables OFF.

Once the response variables are selected, their response gain and
phase plots are produced with the "G" and "P" commands.

The "!" command can be used to toggle Broken-Loop gain and phase plotting.
With U being a response parameter and Uc being a forcing parameter,
the standard gain and phase plots display the usual

  abs(x/u) ,  arg(x/u)

while the broken-loop gain and phase plots display

  abs(1 - u/x) ,  arg(1 - u/x)

The latter are only meaningful if x is the response parameter which
corresponds to the forcing parameter, e.g. u = Flap1, u = Flap1_spec.
```

2.9.3 Integrator-response plotting

```
For computational economy, ASWING will compute the response of control-error
integrators only if they are used in the user-supplied control law routine,
implemented in  src/conlaw.f :


  14   Int(Vi-Vic)dt
  15   Int(be-bec)dt
  16   Int(al-alc)dt
  17   Int(Ph-Phc)dt
  18   Int(Th-Thc)dt
```

```
 19   Int(Ps-Psc)dt

 91   Int(Wx-Wxc)dt
 92   Int(Wy-Wyc)dt
 93   Int(Wz-Wzc)dt

101   Int(a-g-ac)dt
102   Int(a-g-ac)dt
103   Int(a-g-ac)dt


 41     Wx-Wxc dt    (sensor variable)
 42     Wy-Wyc dt       "
 43     Wz-Wzc dt       "
 44     ax-axc dt       "
 45     ay-ayc dt       "
 46     az-azc dt       "
```

If these are not used in the control routine, but are selected for plotting nevertheless, their response will show zero gain.

Making all these variables available for plotting at all times would require assigning one additional righthand side for each variable, producing a 10-20% increase in computational time. This is deemed an unacceptable cost for such a small plotting luxury.

2.9.4 Detailed response display

A detailed response solution for the structural variables can be examined for one specific frequency with the "X" command, which requests that the frequency by selected by clicking on the Bode plot or by inputting the frequency directly.

The L S C V M commands can be used to view the structural-variable response or the geometry response, just like the same commands in the MODE menu. Here, the amplitude to the solution does not grow or decay in time, however.

2.9.5 Frequency-gust modes

Starting with ASWING 5.56, a general gust field can be expanded in streamwise and spanwise modes. A streamwise mode of wavenumber k

```
becomes and unsteady forcing with frequency w = U/k, while a spanwise
mode with wavenumber l remains a spatially varying a spanwise mode.
The spanwise modes are symmetric or antisymmetric.

The number of spanwise gust modes and the gust velocity direction
are both selected with the "I" command.  If 10 modes are selected,
the spanwise modes are

1, cos(pi eta), cos(2 pi eta), ...  cos(9 pi eta)    (symmetric)
   sin(pi eta), sin(2 pi eta), ...  sin(9 pi eta)    (antisymmetric)

BODE allows multiple forcing parameters to be specified,
which is intended for simultaneous display or output of
all the spanwise gust modes.  The multiple forcing parameters
are specified via the first and last indices, e.g.

F

    ---------------------------
      1    Flap01
      2    Flap02
      3    Flap03
      4    Flap04

      7    Peng01

     11    Vg_symm00
     12    Vg_symm01
     13    Vg_symm02
     14    Vg_symm03
     15    Vg_symm04
     16    Vg_symm05
     17    Vg_symm06
     18    Vg_symm07
     19    Vg_symm08
     20    Vg_symm09

     33    Vg_asym01
     34    Vg_asym02
     35    Vg_asym03
     36    Vg_asym04
     37    Vg_asym05
     38    Vg_asym06
     39    Vg_asym07
     40    Vg_asym08
     41    Vg_asym09
```

```
    Enter first [,last] indices of forcing parameter(s):   11 20

This selects all the symmetric modes Vg_symm00 .. Vg_symm09
as forcing variables.
```

2.10 Eigenmode Frequency Analysis – MODE

2.10.1 Overview

```
The eigenmode analysis facility invoked with the MODE command,
which has the following menu:

 ---------------------------------------------------------------
  T oggle constraints (% for keybd)  A dditional eigenvalues
  N ew eigenvalue calculation        P lot-zone radius
  F ree-body (vacuum) toggle         I teration limit

 eX amine eigenmode                  R oot locus plot
  V iew of eigenmode                 M ovie (real-time view)

  L oading eigenmode plot            H ardcopy current plot
  S tructural eigenmode plot         J ot annotations on plot
  C urvature-strain eigenmode plot   Z oom  (B for no distort)
                                     U nzoom

  W rite eigenvalue data to file     O ptions for plots
  D ata file overlay toggle          K ontrol toggles,settings

  G enerate Reduced Order Model      E igenmode output select

  =  list point solution
  #  point number select             Q uery point specs
 ---------------------------------------------------------------

 .MODE (point:  1)   c>


This menu permits the determination of vibration frequencies, damping
ratios, and corresponding mode vectors for perturbations taken about the
current target point, which is any nonlinearly-deformed operating point
state computed previously in OPER.  If no operating points were computed,
then the unloaded geometry at zero lift is used as the base state.

The eigenmode formulation is similar to the forced-response calculation,
```

except that self-excited motions (described by the eigenvalues and eigenmodes)
are now considered.

Negative damping, indicated by a positive real eigenvalue component,
indicates instability. The imaginary component (frequency) and the
eigenmode associated with that eigenvalue indicates the rate and type
of instability involved. Typical instabilities correspond to flight
dynamics instabilities (e.g. spiral instability), or structural
instabilities (e.g. flutter).

2.10.2 Eigenvalue/Eigenmode computation

Eigenvalue/Eigenmode pairs are computed by the "N" or "NN" commands.
As usual, "N" performs the computation for only the current target point,
and "NN" performs it for all available operating points. For the latter
case, it is necessary to set the appropriate constraints for all the points
beforehand. The "N" or "NN" command gives the following prompts:

```
 Enter number of eigenvalues per point:   16

 Enter shift  sigma,omega:   -0.010000    0.000000
```

The number of eigenvalues per operating point can vary anywhere from 1 up
to the maximum permitted by the array limit NEIGX. Between 5 and 20 is
typical in most situations. If the aircraft is perfectly rigid,
at most 12 eigenvalues/eigenmodes will exist. These correspond to the
12 degrees of freedom present (6 rigid-body motions, and their rates),
and 4 of these eigenvalues will be zero (X,Y,Z,heading rigid-body modes
have no effect on dynamics). It is important not to request more
eigenvalues than are present, else the ARPACK Arnoldi algorithm will fail.

The shift sigma,omega give the location in the eigenvalue plane in
whose vicinity the eigenvalues are to be computed. This shift location
will depend on which eigenvalues are being sought. Typically, the
algorithm will return the specified number of eigenvalues which
lie closest to the shift location.

If flight-dynamics modes are being sought, then a shift near 0,0 is
appropriate. These modes have the smallest frequencies or decay time
constants present (e.g. phugoid, spiral, dutch roll), and so will be
typically clustered near the origin. Well-damped modes such as roll
subsidence or short-period will lie farther left on or off the real
axis, but will still be relatively close to the origin.

NOTE: If an aircraft in free-flight is being analyzed, then a shift of
exactly 0,0 cannot be used, since the algorithm creates an LU-decomposition
of the shifted system Jacobian, which is singular at the origin due to
the presence of zero-frequency rigid-body motion modes. For restrained
configurations (e.g. wing clamped in wind tunnel), using the 0,0 shift
is permissible.

If structural dynamics modes are sought, then an appropriate shift is 0,w
with "w" being in the radian frequency in the frequency range of interest.
Bending/torsion flutter will likely occur near the first wing torsional
frequency, so this would be the appropriate shift value.

When the eigenmode analysis is performed the first time, the eigenmode
distribution is often not very well known. In this case one would choose
a relatively large number of eigenvalues to be computed (20-30, say),
and give a shift relatively near 0,0. This can be an expensive computation,
but will give an indication of what shifts are appropriate for what modes,
which can be used for subsequent analyses with fewer eigenvalues specified.

2.10.3 Eigenmode examination

When the eigenvalue computation finishes, the eigenvalue map is displayed.
It can be replotted with the "R" or "RR" commands. If there is an excessive
number of eigenvalues present and there is too much "clutter", only a region
of the map can be designated for plotting with the "P" command, which requests
the center and the radius of the zone on the map which is plotted.

The "X" command allows one particular eigenvalue to be selected by clicking
on it with the mouse. If necessary, a region of the map can be first enlarged
with the "B" command. The geometric part of the eigenmode for the selected
eigenvalue can then be examined via the "V" command, which shows the geometry
along with the following view-manipulation menu:

```
  ----------------------------
   L eft         R ight
   U p           D own
   Z oom         N ormal size
   I ngress      O utgress
   H ardcopy     J ot
   E uler toggle

  < > 0  mode phase
```

```
  - + 1  mode scale

 Type in plot window:  Command,  or  <space> to exit
```

The letter key commands move the location and distance of the observer.
The < and > keys recede or advance the phase of the eigenmode, and
the 0 key resets the phase to zero. The - and + keys change the scaling
factor of the eigenmode (which of course has an arbitrary magnitude), and
the 1 key resets this scaling factor to unity. The E key toggles between
viewing at the actual baseline Euler angles and zero baseling Euler angles.

The "M" command plays a "movie" of the eigenmode, in real time if possible.
The viewpoint and intial scale factor for the movie can be changed first
with "V" if necessary.

The structural part of the eigenmode can be displayed with the L,S,C commands,
which accept the same phase and mode-scaling commands as V.

In all cases, the eigenmode plots will decay or grow in magnitude with
increasing time or phase, depending on whether the mode is stable or
unstable.

2.10.4 Eigenmode output

The "E" command will output the eigenvector elements to a specified file.
The output is performed by SUBROUTINE EVCOUT (in mode.f). Since only
selected eigenvector quantities are typically of interest, the WRITE
statements in this routine can be easily modified to suit a particular
need.

2.10.5 Flutter Analysis

A typical use of the MODE facility is the determination of flutter speed.
This is most conveniently performed by first using OPER to generate
a sequence of quasi-static operating points over a range of flight speeds
which span the expected flutter speed. An eigenvalue calculation is
then performed in MODE using the "NN" command for all these operating
points. A shift of 0,w is used, with "w" being the expected flutter
radian frequency. About 10 eigenvalues per point is typically sufficient
even if the frequency is uncertain. The eigenvalue map will then display

the progression of each eigenvalue with operating point index (i.e. increasing flight speed). The flutter speed and actual flutter frequency is indicated by one of the eigenvalues crossing over from left to right of the imaginary axis.

Flight-dynamic instabilities (e.g. spiral mode) will also have eigenvalues on the right side of the imaginary axis, but these will typically be at a near-zero frequency, and will have completely different eigenmodes. The unstable eigenmode can be viewed with "V" as described above.

2.10.6 Reduced Order Model (ROM) generation

The "G" commands executes the ROM generation procedure. A suitable number of eigenmodes must have been generated previously with the "N" and/or "A" command.

The purpose of a ROM is to approximate the dynamics of the aircraft with a relatively small number of variables. These variables are mode coefficients z_k(t), with each coefficient being a weighting factor for a mode vector v_k. The full state is then a superposition of the individual modes, with the operating-point state u_0 serving as an additional bias.

```
  x(t) = sum_k ( v_k z_k(t) )  +  x_0
```

The approach here is to use specific eigenmodes as the mode vectors v_k. The eigenmodes selected are the ones whose eigenvalues are closest to the origin, since these are expected to dominate the dynamics. The Unsteady Extension theory document derives the overall procedure.

After the "G" command is issued, a listing of the available eigenmodes is presented:

```
    k   abs(lambda)    Re(lambda)      Im(lambda)
    1    0.1865        -0.1865           0.000
    3    0.3609        -0.4024E-01 +/-  0.3587
    5     1.387        -0.3732     +/-   1.335
    6     2.319         -2.319           0.000
    8     3.793         -2.767     +/-   2.594
   10     10.12         -3.769     +/-   9.393
   12     11.97         -9.195     +/-   7.659
   14     12.51         -9.778     +/-   7.807
   16     15.15         -3.284     +/-   14.79
   18     18.74         -2.818     +/-   18.52
```

```
 Enter number of ROM modes ( 18 max ) :   18
```

The user is asked to specify the number of modes to be used, up to the number available (18 in this case), which is also the default.
The ROM matrices are then written into a xxx.rom file.
Specifically, the file includes the following:

```
 k          mode index
 lambda_k   eigenvalue
 v_k        eigenvector
 B'_k       control-variable influence matrix row   =  w_k* B
 C'_k       output (sensor) variable dependence matrix column
              =  C v_k  +  Q v_k lambda_k
 D'         matrix of sensor outputs derivatives w.r.t control variables
              =  D  +  Q v_k w_k* B
```

Each eigenvector v_k and the C'_k column are in general fairly large, with much more information than is needed in practice. Hence, the mode file contains only those components which have been toggled for output. The toggling menu is entered with the "E" command, which is identical to the "R" command in the BODE facility.

Bibliography

[1] L. Avoni, M. Bronz, J.-P. Condomines, and J.-M. Moschetta. *Enhancing ASWING Flight Dynamics Simulations with Closed-Loop Control for Flexible Aircraft.* AIAA American Institute of Aeronautics and Astronautics, 2025.

[2] M. Drela. ASWING.

[3] R. Jan. Récupération d'énergie sur drone à voilure souple.

[4] R. Jan, J.-P. Condomines, and J.-M. Moschetta. Fast simulation model for control law design andbenchmark of high aspect ratio flexible uavs. In J. Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 101–108, Puebla, México, Nov 2021. Paper no. IMAV2021-12.

[5] R. Jan, J.-M. Moschetta, and J.-P. Condomines. Experimental validation of an aeroelasticity framework : ASWING. part i: Aerodynamics. Technical report.

[6] R. Jan, J.-M. Moschetta, and J.-P. Condomines. Experimental validation of an aeroelasticity framework : ASWING. part II: Propellers. Technical report.

[7] R. Jan, J.-M. Moschetta, and J.-P. Condomines. Experimental validation of an aeroelasticity framework : ASWING. part III: Structure. Technical report.

[8] R. Jan, J.-M. Moschetta, and J.-P. Condomines. Experimental validation of an aeroelasticity framework : ASWING. part IV: Aeroelasticity. Technical report.

[9] W. F. Phillips. Mechanics of flight, 2nd edition | wiley.

[10] T. Theodorsen. General theory of aerodynamic instability and the mechanism of flutter, 1934. NTRS Author Affiliations: National Advisory Committee for Aeronautics. Langley Aeronautical Lab. NTRS Document ID: 19930090935 NTRS Research Center: Legacy CDMS (CDMS).