NASA/CR-2004-212994

Modeling and Analysis of Large Amplitude Flight Maneuvers

Mark R. Anderson
Paper Pilot Research, Inc., Sterling, Virginia

August 2004

NASA/CR-2004-212994

Modeling and Analysis of Large Amplitude Flight Maneuvers

Mark R. Anderson
Paper Pilot Research, Inc., Sterling, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-02048

August 2004

Table of Contents

List of Illustrations

List of Tables

List of Symbols

a	section chord
A	acceleration
b	wing span
c	section aerodynamic coefficient
$\bar{c}$	mean aerodynamic chord
C	aircraft or surface aerodynamic coefficient
d	dimension
D	drag force
dp	pressure increment
e	linear cutting plane function
E	matrix of cutting planes
f	linear cell map function
F	cell mapping subspace matrix
g	cutting plane constant
h	section width, altitude
k	time index
L	lift force
M	Markov matrix, pitch moment
n	time index
p	body-axis roll rate, probability vector
p*	steady-state probability vector
q	body-axis pitch rate
Q	expected absorption time matrix
r	body-axis yaw rate, relative position
R	absorption probability matrix
s	quaternion
S	reference area
T	thrust force
u	body-axis forward velocity
v	body-axis lateral velocity
V	velocity relative to air mass
w	body-axis vertical velocity
x	state vector, forward position
y	subspace vector, lateral position
z	vertical position

List of Symbols (Continued)

α	angle-of-attack
β	sideslip angle
γ	induced velocity coefficient
Γ	circulation strength
δ	control deflection
ρ	atmospheric density
Λ	sweep angle
υ	Lorenz model coefficient
σ	vector function of differential equations
ω	rotational velocity

Superscripts

’	nondimensional
2D	two-dimensional
3D	three-dimensional
D	downwash
G	geometric
LE	leading edge
T	transpose

Subscripts

a	absorbing cell
A	aileron
d	delayed
D	drag force
E	elevator
l	roll moment
L	lift force
m	pitch moment
n	yaw moment
t	transient cell
R	rudder
Y	lateral force

1. Introduction

Analytical methods for stability analysis of large amplitude flight maneuvers have been slow to develop for two reasons. The first reason is that the aerodynamic models needed for analysis of highly dynamic flight regimes are not readily available without a great deal of effort and expense. Most existing models have been developed for military aircraft and are considered proprietary or have restricted availability. The second reason is that stability analysis techniques for nonlinear systems are often limited to a state-space dimension of less than three. The state-space dimension of most aircraft dynamic models exceeds ten, while complete models including a flight control system can easily approach thirty. The research described in this report addresses these issues by providing a new method to rapidly assess the stability properties of high-dimensional aircraft state-space dynamic models.

Fundamental flight dynamic investigations do not require models of the highest fidelity; however, it is imperative that the models possess the characteristics needed to represent the essential dynamics. The modeling method described herein provides this level of fidelity in a format that is easy to use and modify. The models are not expected to replace wind tunnel testing or more sophisticated methods in computational fluid dynamics. However, configurations can be defined in a matter of minutes and a wide variety of control devices or effects can be represented in the model. The modeling technique utilizes the nonlinear lifting line method first proposed by Piszkin and Levinsky in 1975.[1,2] This method is augmented with other existing approximation methods to handle wake roll-up effects and rotating flow. Details of the modeling technique are presented in Chapter 2 of this report.

The generalized cell-to-cell mapping method, introduced by Hsu[3], is used as the foundation for nonlinear system stability analysis. Hsu's cell mapping method cannot be applied to high-dimensional systems without considerable computational difficulty so an alternative method has been developed. The approach described in Chapter 3 is to create a cell map within a two-dimensional subspace of the higher dimensional state-space. The stability properties of the aircraft become apparent by studying the behavior near a series of several strategically placed subspaces. This method offers an intermediate solution that is more representative of nonlinear behavior than linearized local solutions and yet avoids the difficulties in creating a cell map for the entire state space.

An example analysis of a general aviation (GA) aircraft configuration is provided in Chapter 4. The GA configuration is modeled after the Grumman Yankee aircraft that was tested at NASA Langley in the early 1980's. In addition to wind tunnel tests, an extensive flight test program was undertaken to study the stall and spin characteristics of this airplane. A complete aerodynamic model is constructed for this airplane using the nonlinear lifting line method. Model characteristics are shown to compare favorably with previous wind tunnel test results. Cell-to-cell maps are then generated within critical subspaces near the dominant spin mode of the airplane. These maps reveal important information about the stall and spin behavior and are shown to be consistent with flight test experience.

2. Aerodynamic Modeling

There are several advantages of the Piszkin and Levinsky method that make it ideal for the study of aircraft dynamics and control in highly dynamic flight regimes. First and foremost, the method can provide aerodynamic modeling information for a very large angle-of-attack range. It can also provide reasonable solutions without expensive computers or a significant effort devoted to problem set-up, analysis, and post-processing. The method can readily include variations in airfoil section characteristics. Multiple control surfaces or spanwise changes in shape can be modeled by varying the airfoil section characteristics accordingly.

The method appears to predict several of the nonlinear dynamic phenomenon of interest including dynamic stall, pitch-up, nose slice, wing rock, wing drop, and control loss or reversal. Multiple solutions are also possible when the airfoil section characteristics have discontinuities. Piszkin and Levinsky postulated that these multiple solutions could yield randomly appearing phenomenon such as wing drop. The method introduces unsteady wake effects by using a discrete vortex system that trails behind the primary bound lifting line. The strength of each shed vortex is related to the corresponding bound vortex at an earlier point in time. Three-dimensional unsteady aerodynamic effects are therefore included in the development.

Two minor modifications have been added to the nonlinear lifting line method to improve its prediction characteristics. The first modification is to shift the wake vortices according to the induced velocity at each corner point of the ring vortex. This change adequately represents wake roll-up.[4] The other modification is to include the affect of rotating flow. The rotating flow approximation is intended to improve the predicted aerodynamics at high angular rates due to spinning motions.

2.1 Nonlinear Lifting Line Overview

The vortex system of the nonlinear lifting line method is constructed by first dividing the aerodynamic surface into spanwise sections of equal width. A ring vortex is bound to each section. One edge of the bound vortex is aligned with a line segment located at 1/4-chord distance from the leading edge. The streamwise edges of the bound vortex are aligned with the root chord. The downstream edge is placed at a user-defined distance from the forward edge. A control point is located midway between the spanwise edges and at a distance of 3/4-chord from the leading edge. This control point location is used to determine the effective angle-of-attack and sideslip of the section.

Figure 2.1 illustrates the complete vortex system. The aerodynamic surface is represented in gray. It has been divided into a total of sixteen sections. A parallelogram-shaped ring vortex has been attached to each section along with a control point. This example shows a wake structure consisting of five rows. A conventional, right-handed, (x,y,z) coordinate system is shown in the figure and is used for all distance measurements.

The Kutta-Joukowski theorem states that the lift per unit span L_i is related to the vortex strength Γ_i at section i by,

$$L_i = \rho V \Gamma_i \tag{2.1}$$

This expression can also be written as,

$$L_i = 0.5\rho V^2 \left[2\left(\frac{\Gamma_i}{V} \right) \right] \tag{2.2}$$

The total lift force L acting on the aerodynamic surface is the sum of the lift for each section, multiplied by the section width h_i,

$$L = \sum_i h_i L_i = 0.5\rho V^2 \sum_i h_i \left[2\left(\frac{\Gamma_i}{V} \right) \right] \tag{2.3}$$

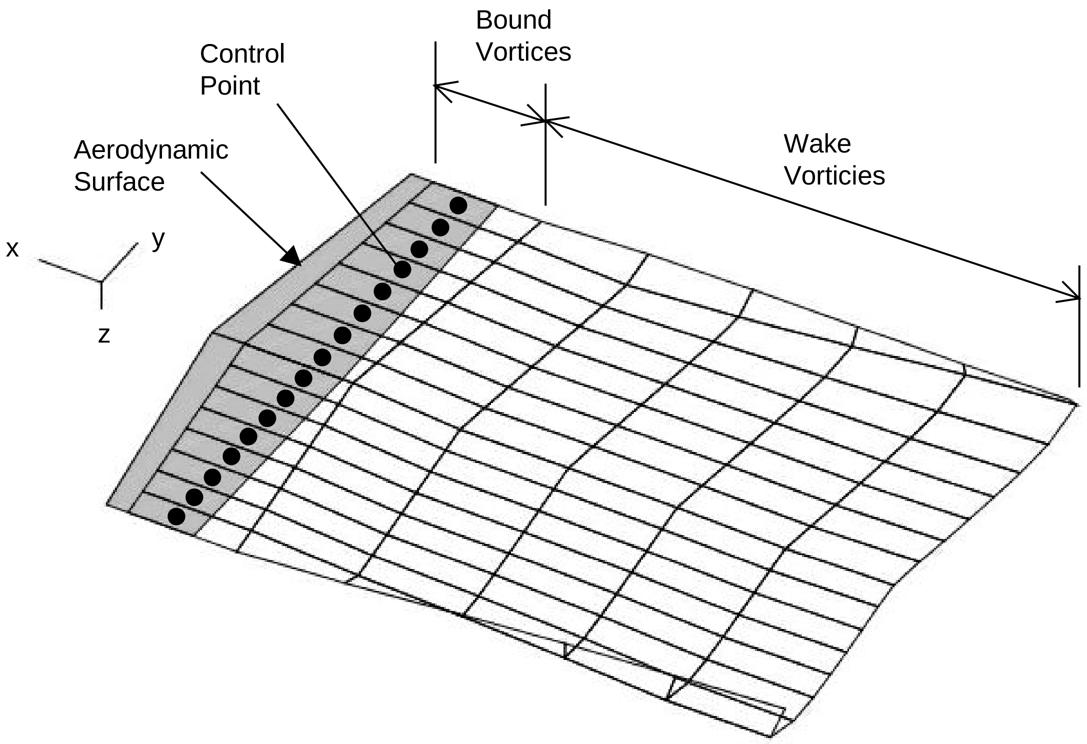


Figure 2.1 Vortex System

Following standard practice, the lift coefficient C_L for the aerodynamic surface is related to the total lift force,

$$L = 0.5\rho V^2 S C_L \tag{2.4}$$

where S is the reference area. Combining (2.3) and (2.4) leads to an expression for the lift coefficient,

$$C_L = \sum_i \left(\frac{h_i}{S}\right)\left[2\left(\frac{\Gamma_i}{V}\right)\right] = \sum_i \left(\frac{h_i a_i}{S}\right)\left[\left(\frac{2}{a_i}\right)\left(\frac{\Gamma_i}{V}\right)\right] \tag{2.5}$$

where a_i is the chord length of section i. Equation (2.5) can also be written as,

$$C_L = \sum_i \left(\frac{h_i a_i}{S}\right) c_{Li} \tag{2.6}$$

by defining the lift coefficient for section i as,

$$c_{Li} = \left(\frac{2}{a_i}\right)\left(\frac{\Gamma_i}{V}\right) \tag{2.7}$$

A linear solution would set (2.7) equal to a linear function of angle-of-attack. Piszkin and Levinsky, however, allowed the section lift coefficient to be written as a nonlinear function of angle-of-attack. In this case, (2.7) is written as,

$$c_{Li}(\alpha_i) = \left(\frac{2}{a_i}\right)\left(\frac{\Gamma_i}{V}\right) \tag{2.8}$$

The section angle-of-attack α_i consists of two components,

$$\alpha_i = \alpha_i^G - \alpha_i^D \tag{2.9}$$

The first component α_i^G is the "geometric" angle-of-attack that stems from the aircraft motion. The second component α_i^D comes from the downwash generated by the other vortices in the vortex system.

The Biot-Savart vortex filament laws are used to find the induced velocity at the control point resulting from the other elements in the vortex system. Details of this calculation can be found in most aerodynamics texts. The general form of the final expression is,

$$\alpha_i^{3D} = \sum_j \gamma_{i,j}\left(\frac{\Gamma_j}{V}\right) \tag{2.10}$$

where $\gamma_{i,j}$ is the velocity induced at the control point of section i, due to the vortex j.

Piszkin and Levinsky intended to represent the nonlinear section lift coefficient using two-dimensional airfoil data. Consequently, the theoretical two-dimensional downwash must be subtracted from the three-dimensional downwash found above. The complete downwash angle-of-attack becomes,

$$\alpha_i^D = \alpha_i^{3D} - \alpha_i^{2D} \tag{2.11}$$

where,

$$\alpha_i^{2D} = \frac{\Gamma_i / V}{a_i \pi \cos(\Lambda_i - \beta_i)} \tag{2.12}$$

In (2.12) above, Λ_i is the sweep angle of the bound vortex leading edge and β_i is the geometric sideslip at section i.

Some care must be used when evaluating (2.12) to insure that the signs of Λ_i and β_i are handled correctly. The computer software does not specifically evaluate the cosine term in (2.12). Instead, this term is found as a dot product of two vectors. One vector is aligned along the leading edge of the bound vortex. The other vector is defined as the projection of the velocity vector onto the plane defined by the bound ring vortex.

The solution method is relatively straightforward. The unknowns consist of the normalized strength Γ_i /V at each section. Solutions can be determined using an iterative Newton method. The objective is to determine Γ_i /V such that (2.8) is satisfied at every section. Multiple solutions are possible and convergence is not guaranteed for arbitrary section lift characteristics.

The control point is used to determine the effective angle-of-attack and sideslip at each section. Assume that the control point of section i is located at a position (x_i, y_i, z_i) from a reference point. The body-axis velocity components at the control point are:

$$u_i = V\cos\alpha\cos\beta + qz_i - ry_i \tag{2.13}$$
$$v_i = V\sin\beta + rx_i - pz_i \tag{2.14}$$
$$w_i = V\sin\alpha\cos\beta + py_i - qx_i \tag{2.15}$$

where p, q, and r are the angular velocity components.[5] The angular velocities are typically normalized such that: p' = pb/2V, q' = q$\bar{c}$/2V, r' = rb/2V (where b = span and $\bar{c}$ = mean aerodynamic chord). Using the normalized angular rates, the equations above become,

$$u_i / V = \cos\alpha\cos\beta + \left(2q'/\bar{c}\right) z_i - \left(2r'/b\right) y_i \tag{2.16}$$
$$v_i / V = \sin\beta + \left(2r'/b\right) x_i - \left(2p'/b\right) z_i \tag{2.17}$$
$$w_i / V = \sin\alpha\cos\beta + \left(2p'/b\right) y_i - \left(2q'/\bar{c}\right) x_i \tag{2.18}$$

Equations (2.16)-(2.18) above are evaluated using the following definitions,

$$\vec{V}_i = \begin{bmatrix} u_i / V \\ v_i / V \\ w_i / V \end{bmatrix} \qquad \vec{V} = \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix} \qquad \vec{r}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad \vec{\omega} = \begin{bmatrix} 2p'/b \\ 2q'/\bar{c} \\ 2r'/b \end{bmatrix}$$

With the definitions above, equations (2.16)-(2.18) are written as:

$$\vec{V}_i = \vec{V} + \vec{\omega} \times \vec{r}_i \tag{2.19}$$

The geometric angle-of-attack and sideslip at the control point of section i are found using the components of velocity at the control point. The required equations are:

$$\alpha_i = \tan^{-1}\left(\frac{w_i / V}{u_i / V}\right) \qquad (2.20)$$

$$\beta_i = \sin^{-1}(v_i / V) \qquad (2.21)$$

The acceleration of the control point i is needed to determine the pressure gradient resulting from rotating flow. The required expression is:

$$\frac{\vec{A}_i}{V^2} = \vec{\omega} \times \vec{\omega} \times \vec{r}_i \qquad (2.22)$$

The aerodynamics of a spinning airplane are evaluated experimentally using a rotary balance wind tunnel. The aircraft model is rotated about the freestream velocity vector to approximate the motions that are typical of a steady spin. In this case, the body-axis rotational rates are related to the spin rate Ω,

$$p = \Omega \cos\alpha \cos\beta \qquad (2.23)$$
$$q = \Omega \sin\beta \qquad (2.24)$$
$$r = \Omega \sin\alpha \cos\beta \qquad (2.25)$$

The normalized spin rate is defined as $\Omega' = \Omega b/2V$. Substituting for the spin rate in (2.23)-(2.25) yields,

$$p = V(2\Omega' / b) \cos\alpha \cos\beta \qquad (2.26)$$
$$q = V(2\Omega' / b) \sin\beta \qquad (2.27)$$
$$r = V(2\Omega' / b) \sin\alpha \cos\beta \qquad (2.28)$$

The control point velocity in the spinning case can therefore be computed using the following angular rate vector along with (2.19) and (2.22),

$$\vec{\omega} = (2\Omega' / b)\vec{V} = \begin{bmatrix} (2\Omega' / b) \cos\alpha \cos\beta \\ (2\Omega' / b) \sin\beta \\ (2\Omega' / b) \sin\alpha \cos\beta \end{bmatrix} \qquad (2.29)$$

Solution of (2.8) leads to the lift coefficient $c_{Li}(\alpha_i)$ and effective angle-of-attack α_i for each section. The effective angle-of-attack is then used to determine the drag and moment coefficients for the section, $c_{Di}(\alpha_i)$ and $c_{mi}(\alpha_i)$, respectively. The drag force vector is aligned in the direction opposite of the velocity vector at the control point, which is found by (2.19). The drag force, normalized by dynamic pressure, is therefore represented by the following vector expression,

$$\frac{\vec{D}_i}{0.5\rho V^2} = -a_i h_i c_{Di}(\alpha_i)\frac{\vec{V}_i}{|\vec{V}_i|} \tag{2.30}$$

The lift force vector is perpendicular to the plane defined by the velocity vector and the leading edge of the bound vortex. The lift force vector is found using the cross product of these two vectors,

$$\frac{\vec{L}_i}{0.5\rho V^2} = -a_i c_{Li}(\alpha_i)\left[\frac{\vec{V}_i}{|\vec{V}_i|}\times \vec{r}_i^{LE}\right] \tag{2.31}$$

where $\vec{r}_i^{LE}$ is the vector that defines the leading edge of the bound vortex of section i. Note that this vector will have length that is approximately equal to h_i. The total force, normalized by dynamic pressure, is the sum of the lift and drag force vectors.

The section moment is assumed to act about the leading edge of the bound vortex. The moment vector acting at section i is therefore,

$$\frac{\vec{M}_i}{0.5\rho V^2} = a_i^2 c_{mi}(\alpha_i)\vec{r}_i^{LE} \tag{2.32}$$

The lift and drag forces also contribute to the total moment acting at the section. It is assumed that the combination of these forces acts at a location midway across the leading edge of the bound vortex. The vector from the reference point to this midpoint defines the moment arm. This contribution to the total moment is computed as the cross product of the moment arm and the total force.

2.1 Rotating Flow

Separated flow conditions occur at angles-of-attack above stall. When the aerodynamic surface is rotating, the fluid in the separated flow region moves with the surface.[6] A pressure gradient is formed along the surface that is proportional to the rotational acceleration. This pressure gradient leads to additional forces and moments that should be included in the prediction method if the intent is to study flight conditions involving high angular rates and spinning motions. This solid body rotating flow effect is modeled by computing a pressure increment at each section. The increments are then summed across all sections of the surface to yield the pressure distribution caused by rotating, separated flow.

The acceleration at the control point of section i is found in (2.22). The pressure increment at section i, normalized by dynamic pressure, is given by,

$$\frac{dp_i}{0.5\rho V^2} = 2\left|\frac{\vec{A}_i}{V^2}\cdot \vec{r}_i^{LE}\right| \tag{2.33}$$

The dot product expression in (2.35) isolates the components of the control point acceleration in the direction of the bound vortex leading edge. The resulting pressure increment is proportional to the rotational acceleration in the spanwise direction only.

The total pressure at each section is computed by summing the pressure increments at each section. The summation is completed from the tip section to the root section to satisfy the boundary condition of zero pressure at the surface tip. Note also that a pressure increment is computed only for sections where the effective angle-of-attack is greater than a user-defined separated flow limit. Consequently, the aerodynamic surface can have sections with attached flow and sections with separated, rotating flow. The force produced by this effect is assumed to act in the direction perpendicular to the plane of the bound vortex.

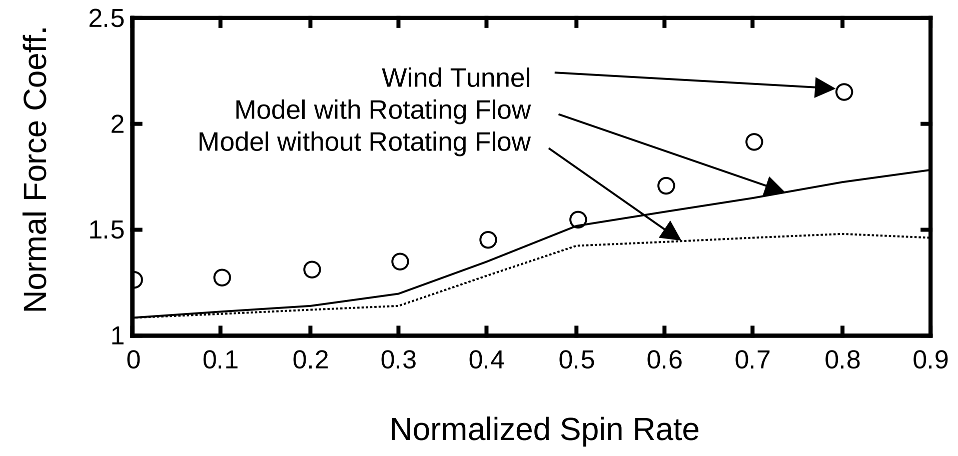
Figure 2.2 illustrates how rotating flows contribute to the aerodynamics of the typical aircraft configuration. The top of Figure 2.2 is the normal force coefficient plotted with respect to the normalized spin rate Ω' for a 35 deg angle-of-attack. The bottom half is the rolling moment coefficient. The open circles in these figures are rotary balance wind tunnel results. The dotted line is the coefficient predicted using the nonlinear lifting line program. The solid line includes the rotating flow effect. Notice that the rotating flow increases the amplitude of both the normal force and the rolling moment. However, it should also be noted that this effect does not appear to be significant for normalized spin rates of less than 0.5.

2.3 Airfoil Sections

The nonlinear lifting line program requires two-dimensional (infinite aspect ratio) airfoil section data for each aerodynamic surface. Lift, drag, and moment coefficients are required as a function of angle-of-attack. A very wide angle-of-attack range is needed so that post-stall and other highly dynamic flight regimes can be modeled. Traditional sources of this information, such as Reference [7], include data for only a few degrees beyond stall. Reference [8], produced in 1955, describes wind tunnel results of the NACA 0012 airfoil section for angles-of-attack up to 180 degrees. This data was used to define analytical functions representing the NACA 0012 airfoil section. Analytical functions were used, instead of tabular data, so that the functions could also serve as a template for other airfoil types.

Figure 2.3 shows the lift and drag coefficient data, represented by the open circles, for the NACA 0012 airfoil section. This data represents the case of a standard roughness and a Reynolds number of 1,800,000. An analytical function was fitted to the lift coefficient data. This function is marked by a solid line in Figure 2.3 and is defined as follows,

$$\begin{aligned}
&\alpha^* = \alpha + \frac{\partial\alpha}{\partial\delta}(\delta + \delta_O) \\
&h_N = \exp\left[-\mu(\alpha^* + \alpha_S)^2\right] \\
&h_P = \exp\left[-\mu(\alpha^* - \alpha_S)^2\right] \\
&c_L(\alpha,\delta) = p_{LO}(h_P - h_N) + p_{HI}\sin(2\alpha^*) + \frac{\partial c_L}{\partial\delta}(\delta + \delta_O)
\end{aligned} \tag{2.34}$$



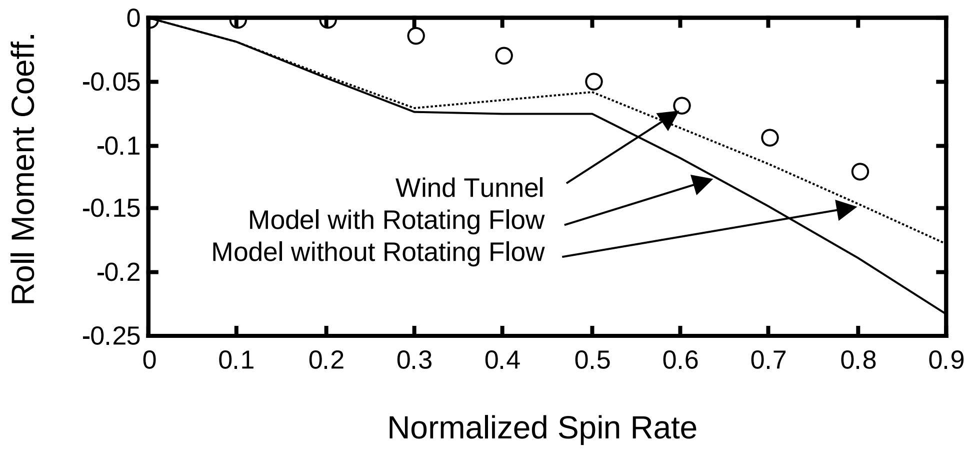


Figure 2.2 Rotating Flow Effect

The parameters that are used in the section lift coefficient function are defined to represent the shape characteristics of the resulting curve. For example, the curve consists of two peaks. The primary stall peak is located at an angle-of-attack equal to α_S and its magnitude is approximately equal to the parameter p_{LO}. The width of the primary stall peak is governed by μ. The amplitude of the second peak, always located at 45 deg angle-of-attack, is p_{HI}. The parameters $\partial\alpha/\partial\delta$ and $\partial c_L/\partial\delta$ define how the control parameter δ (in this case, a 20% chord mechanical flap) influence the angle-of-attack and section lift coefficient, respectively. The section drag coefficient, in all cases, is modeled by the following function,

$$c_D(\alpha,\delta) = \begin{Bmatrix} 0 & ,|\alpha| < 0.2 \\ 2.5\sin|\alpha| - 0.5 & ,|\alpha| \geq 0.2 \end{Bmatrix} \tag{2.35}$$

This function is also marked and shown in Figure 2.3 as a solid line.

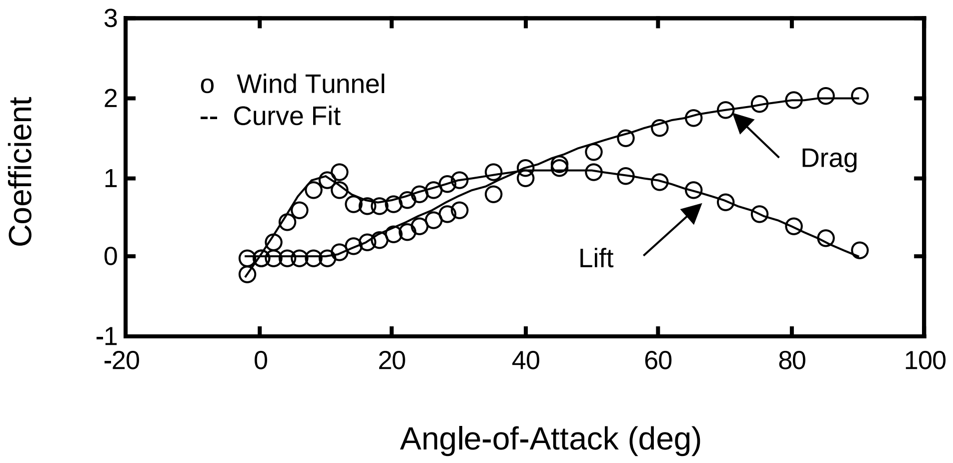


Figure 2.3 NACA 0012 Airfoil Section Lift and Drag Coefficients

Figure 2.4 depicts the section moment coefficient data for the NACA 0012 airfoil. This data was fit to the following analytical function shape:

$$c_m(\alpha,\delta)=\left\{\begin{array}{ll} c_{mo}+\dfrac{\partial c_m}{\partial\delta}\delta & ,|\alpha|<0.2 \\ c_{mo}+\dfrac{\partial c_m}{\partial\delta}\delta+\dfrac{\partial c_m}{\partial\alpha}(\alpha-0.2\sin\alpha) & ,|\alpha|\geq 0.2 \end{array}\right\} \qquad (2.36)$$

The function shown in (2.36) is depicted as the solid line in Figure 2.4.

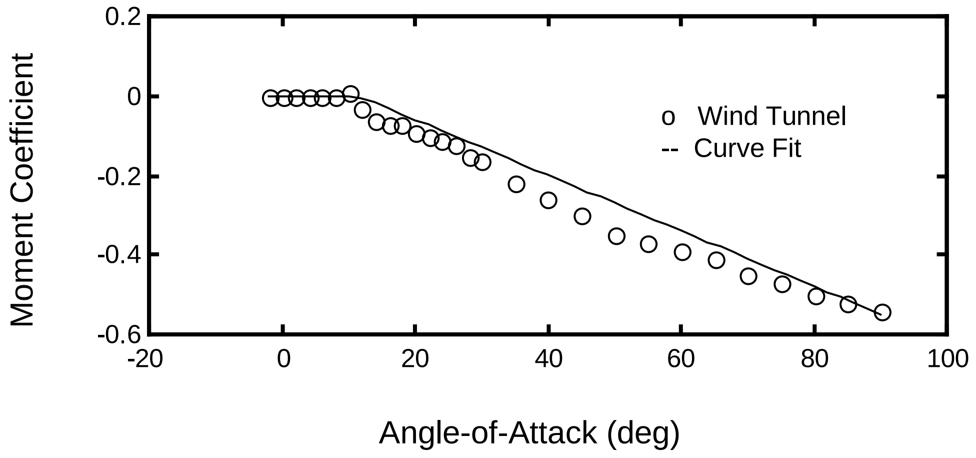


Figure 2.4 NACA 0012 Airfoil Section Moment Coefficient

The parameters of the lift and moment functions were adjusted to approximate the two different airfoil sections needed for the configuration results shown later in this report. The adjustments were made to approximate the coefficients shapes from Reference [7] whenever possible. Table 2.1 lists the section modeling parameters for the two airfoil sections. Angle-of-attack and the flap control parameter are in expressed in radians.

Table 2.1 Airfoil Section Modeling Parameters

Parameter	NACA 0012	NACA 64_2-415
α_S	0.15	0.20
p_{LO}	0.67	1.00
μ	100	50
p_{HI}	1.1	0.4
$\partial\alpha/\partial\delta$	0.1	0.1
$\partial c_L/\partial\delta$	0.86	0.86
δ_O	0.0	0.2
c_{mO}	0.0	-0.08
$\partial c_m/\partial\delta$	-0.22	-0.22
$\partial c_m/\partial\alpha$	-0.4	-0.3

2.4 Coefficient Expansion

The nonlinear lifting line program yields tabular force and moment coefficient data as a function of the following independent variables: angle-of-attack α, sideslip β, nondimensional roll rate p', nondimensional pitch rate q', nondimensional yaw rate r', and one or more control variables δ. The lift coefficient data, for example, is expressed as a function of the independent variables:

$$C_L = C_L(\alpha, \beta, p', q', r', \delta) \tag{2.37}$$

The functional notation depicted in (2.37) above will be used to represent the numerical results obtained from the lifting-line program. In other words, (2.37) represents the aircraft lift coefficient data that is obtained by a single "run" of the lifting line program using a given set of independent variables.

It is desirable to create a compact analytical representation of the aerodynamic data by fitting analytical functions to the tabular data. Multi-dimensional spline functions could be used to represent (2.37) directly. However, multi-dimensional data-fitting is computationally difficult above two or three dimensions so an extension of the more traditional non-dimensional stability derivative models will be used instead. The resulting form allows for ready comparison to other existing aerodynamic models.

The aerodynamic characteristics are dominated by variations in angle-of-attack. This behavior is captured by setting all of the independent variables equal to zero with the exception of angle-of-attack. Under these conditions, the lifting line program yields tabular data in the form:

$$C_{L0}(\alpha) = C_L(\alpha, 0, 0, 0, 0, 0) \tag{2.38}$$

where angle-of-attack α is specified by a set of discrete values.

For a conventional aircraft configuration, the effects of the remaining independent variables can be approximated using a series expansion. For example, adding a second term to the lift coefficient expression incorporates the effect of pitch rate variations,

$$C_L = C_{L0}(\alpha) + C_{Lq}(\alpha)q' \tag{2.39}$$

The coefficient data for $C_{Lq}(\alpha)$ is obtained by running the lifting line program while setting the independent variable q' to a single non-zero value and allowing angle-of-attack to vary. It is important to verify that variations in coefficient data are linearly related to variations in the independent variable. For variations in q', one should verify that,

$$C_L(\alpha, 0, 0, q'/2, 0, 0) \cong C_L(\alpha, 0, 0, q', 0, 0)/2 \tag{2.40}$$

where q' has been set to its maximum expected value. As long as (2.40) holds, the new coefficient term can be determined using a two-sided difference expression,

$$C_{Lq}(\alpha) = \frac{C_L(\alpha, 0, 0, q', 0, 0) - C_L(\alpha, 0, 0, -q', 0, 0)}{2q'} \tag{2.41}$$

In some cases, the variation in force or moment depends upon the amplitude of independent variable. For example, the lift coefficient variation with respect to roll rate typically satisfies the following condition,

$$C_L(\alpha, 0, p', 0, 0, 0) \cong C_L(\alpha, 0, -p', 0, 0, 0) \tag{2.42}$$

This roll rate variation is introduced by adding a new term that is a multiple of the normalized roll rate absolute value,

$$C_L = C_{L0}(\alpha) + C_{Lq}(\alpha)q' + C_{L|p|}(\alpha)|p'| \tag{2.43}$$

The roll rate coefficient data is found using the following two-sided difference,

$$C_{L|p|}(\alpha) = \frac{C_L(\alpha, 0, p', 0, 0, 0) + C_L(\alpha, 0, -p', 0, 0, 0) - 2C_L(\alpha, 0, 0, 0, 0, 0)}{2p'} \tag{2.44}$$

where p' is set to its maximum expected value and α is allowed to vary over its operating range.

The lifting line program can also be used to determine the effect of variations in angle-of-attack and sideslip rate. These unsteady effects are handled using a time-delayed angle-of-attack or sideslip to compute the strength of the vortices bound to each aerodynamic surface. The following discussion will describe angle-of-attack rate calculations. Sideslip rate variations are modeled in an obviously similar fashion.

The time-delayed angle-of-attack α_d is defined as,

$$\alpha_d = \alpha - \tau\dot{\alpha} \tag{2.45}$$

where τ is a constant representing the time delay. This time constant represents the total time that is required for the bound vortex circulations to reach new values after a large change in angle-of-attack. The delay is assumed to be caused by various phenomena such as flow separation, reattachment, and boundary-layer formation. The wake vortices and rollup are not delayed and are computed as before.

Normalizing the delay time constant such that $\tau' = 2V\tau/\bar{c}$ results in,

$$\alpha_d = \alpha - \tau'\dot{\alpha}' \tag{2.46}$$

where $\dot{\alpha}' = \dot{\alpha}\bar{c}/2V$ is the non-dimensional angle-of-attack rate. The angle-of-attack rate effect is added to the total lift coefficient expression as,

$$C_L = C_{L0}(\alpha) + C_{Lq}(\alpha)q' + C_{L|p|}(\alpha)|p'| + C_{L\dot{\alpha}}(\alpha)\dot{\alpha}' \tag{2.47}$$

The lift coefficient due to variations in angle-of-attack rate is then computed using,

$$C_{L\dot{\alpha}}(\alpha) = \frac{C_L(\alpha - \tau'\dot{\alpha}', 0, 0, 0, 0, 0) - C_L(\alpha + \tau'\dot{\alpha}', 0, 0, 0, 0, 0)}{2\tau'\dot{\alpha}'} \tag{2.48}$$

where $\dot{\alpha}'$ is set to close to its maximum expected value and α varies over its expected range.

Fan and Lutze used a delay time constant of $\tau' = 8.0$ to model a fighter aircraft configuration undergoing large pitching motions.[9] This time constant implies that the airflow must travel four chord lengths beyond the wing surface before the circulation reaches its new value. Citing results obtained in a water tunnel, Brandon suggests that it may take as long as thirty chord lengths to achieve a new steady-state.[10] Note that the delay time constant is only used in (2.48) and does not become part of the total lift coefficient expression (2.47). As a result, the lift coefficient due to angle-of-attack rate variations obtained in (2.48) can be compared directly to the stability derivative approximations that are generally used to model this effect.

A typical coefficient expansion model structure for a complete aircraft is shown in Table 2.2. This model has been found suitable for several conventional aircraft configurations and is also consistent with traditional linear stability derivative models. The general aviation aircraft example described later in this report will also rely on this model format.

The coefficients defined in Table 2.2 are fitted to polynomial forms to create a compact representation of the aerodynamic model. There are a number of algorithms available to accomplish this task. The results presented herein were obtained using the least-squares rank reduction method described in Reference [11]. This method uses the singular value decomposition of the data matrix to diagonalize the components of the least-squares objective function. Morelli achieves a similar result by building the data matrix with orthogonal functions.[12] The orthogonal function method avoids the singular value decomposition and therefore is well suited to problems involving large data tables. For this modeling application, the data tables rarely exceed thirty points so the singular value decomposition is used.

Table 2.2 Aerodynamic Model Coefficient Expansion

$$C_D = C_{D0}(\alpha) + C_{Dq}(\alpha)q' + C_{D\delta}(\alpha)\delta$$

$$C_Y = C_{Y\beta}(\alpha)\beta + C_{Yp}(\alpha)p' + C_{Yr}(\alpha)r' + C_{Y\delta}(\alpha)\delta$$

$$C_L = C_{L0}(\alpha) + C_{L\dot{\alpha}}(\alpha)\dot{\alpha}' + C_{L|p|}(\alpha)|p'| + C_{Lq}(\alpha)q' + C_{L|r|}(\alpha)|r'| + C_{L\delta}(\alpha)\delta$$

$$C_l = C_{l\beta}(\alpha)\beta + C_{lp}(\alpha)p' + C_{lr}(\alpha)r' + C_{l\delta}(\alpha)\delta$$

$$C_m = C_{m0}(\alpha) + C_{m\dot{\alpha}}(\alpha)\dot{\alpha}' + C_{m|p|}(\alpha)|p'| + C_{mq}(\alpha)q' + C_{m|r|}(\alpha)|r'| + C_{m\delta}(\alpha)\delta$$

$$C_n = C_{n\beta}(\alpha)\beta + C_{np}(\alpha)p' + C_{nr}(\alpha)r' + C_{n\delta}(\alpha)\delta$$

3. Nonlinear Stability Analysis

Analytical investigations of aircraft motion stability often focus on local solution regions surrounding equilibrium or critical points. These results are applicable to small perturbations in each of the motion variables. Global analysis, or analysis of large amplitude motions, is usually conducted using deterministic simulations or small-scale model experiments. This recipe has worked well for the vast majority of aviation applications. The major drawback of the procedure is that it can become expensive and time-consuming when the aircraft is expected to be highly maneuverable or to operate throughout a relatively large flight envelope.

Hsu's cell-to-cell mapping method divides the operating domain of the nonlinear system into a large number of smaller regions or "cells". The cell-to-cell map represents how the solution trajectories of the nonlinear system move from one cell to another. The power of the cell mapping theory comes from the fact that the map itself represents a linear Markov chain. Consequently, linear system stability analysis tools can be applied to the map and the results reveal how the underlying nonlinear system will behave.

As one might expect, the cell-mapping concept suffers from the curse of dimensionality. With some effort, cell-to-cell maps can be constructed for systems of up to about five dimensions using a typical modern workstation. A 5-dimensional state-space domain, with three dimensions having 20 divisions and two having 10 divisions, would lead to a total of 800,000 cells and would require about 4 megabytes of memory in 32-bit floating point format. However, a total of 20^{10} cells are needed to tessellate a 10-dimensional state-space such that each dimension has 20 divisions. The map itself would require over 40 terabytes of memory!

A six degree-of-freedom aircraft dynamic model requires a state-space dimension of at least nine. An automatic flight control system or human pilot model will likely add several more state variables. The most common approach to analysis is to first construct a high-fidelity model with a large number of state variables and then attempt to reduce the state-space dimension to a more manageable number. There are a number of methods available to reduce state-space dimension if the model is represented by a linear system. For nonlinear models, the analyst must use physical insight to eliminate state variables that are not expected to contribute to the motion under study. The analyst may need to be repeat this process several times in order to gain sufficient insight into the behavior of the nonlinear system.

The approach advocated by this research is to define a two-dimensional subspace within the state-space domain of interest. The cell-to-cell mapping method is then used to create a cell map on the two-dimensional subspace. This approach has many advantages. The total number of cells needed to represent a two-dimensional cell map will typically be less than one thousand. This relatively low value means that the statistical properties of the map can be readily computed using standard algorithms. Standard plotting routines can also be used to visualize the results.

The most significant advantage of using two-dimensional subspaces is that the cell map can be created in just a few seconds on a modest workstation. The concept is to offer an interactive environment wherein the analyst rapidly changes the subspace orientation and location within the state-space to gather an overall sense of the global stability characteristics. This procedure yields the same results as one would find by creating several reduced-order versions of the same high-fidelity model, but does so in much more systematic manner.

3.1 Cell Mapping Overview

Figure 3.1 illustrates a two-dimensional grid of cells that might be used to create a cell-to-cell map for a second-order nonlinear system. The state-space is divided into nine rectangular cells. The cell-to-cell map for this system would be represented numerically by a nine-by-nine matrix. In simple cell mapping, one trajectory is computed for each cell. The trajectory begins at a point within the cell and ends after a specified time period. The association between the starting cell and the endpoint cell determines one element of the cell-to-cell map.

Generalized cell-to-cell mapping considers the possibility of multiple trajectories starting from within each cell. This characteristic is illustrated in Figure 3.1, where nine trajectories are shown to start from within cell 5. Note that all trajectories starting from cell 5 will not necessarily end in the same cell. In Figure 3.1, two of the nine trajectories end in cell 4, four trajectories end in cell 1, two end in cell 2, and one remains in cell 5.

The generalized cell map can be interpreted in a probabilistic sense. If the initial condition of the nonlinear system is located within cell 5, then Figure 3.1 shows that the solution at the next time step will lie in cell 4 with probability 2/9, cell 1 with probability 4/9, cell 2 with probability 2/9, and cell 5 with probability 1/9. These probabilities form the elements of the generalized cell map. The fifth column of generalized cell map for this case would contain the following entries: M(1,5) = 4/9, M(2,5) = 2/9, M(4,5) = 2/9, and M(5,5) = 1/9.

The generalized cell map is most typically constructed by filling columns as each domain cell is considered in turn. The sum of the column entries will equal unity unless one (or more) of trajectories leaves the cell state-space. In this case, the trajectory is said to have entered the sink cell. The *sink cell* represents the entire region outside the domain of interest.

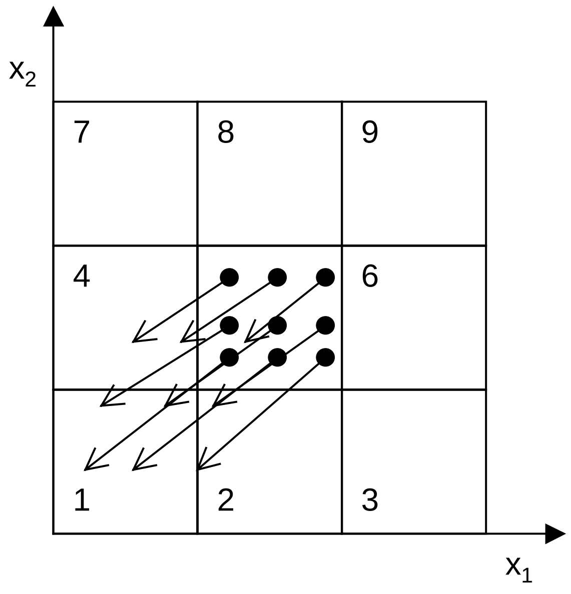


Figure 3.1 Generalized Cell Mapping

3.2 Markov Chains

A number of theoretical properties found in the literature of Markov chains can be applied to analysis of generalized cell-to-cell maps. Many of these properties have been adapted by Hsu to further develop the cell-to-cell mapping concept. This section will review the specific properties of Markov chains that are relevant to this application. The review will focus on the general case of acyclic (non-periodic) behavior.

The Markov chain that describes the generalized cell-to-cell map is specified by the map matrix M and a probability vector p. The Markov chain is represented by the following set of linear equations:

$$p(k+1) = M\, p(k) \qquad (3.1)$$

The one-step indexing from k to k+1 denotes the time period that the trajectories were followed when the cell-to-cell map matrix M was built. For example, if each trajectory was followed for three seconds, the one-step index represents a time increment of the same duration. Each element of the probability vector p is associated with a domain cell. The number of elements in p will equal the number of cells in the cell-to-cell map. The map matrix M will be square, with as many rows and columns as there are domain cells.

The numerical value of an element in p is defined as the probability that a solution of the nonlinear system lies within the associated domain cell. It is therefore possible to specify a probability distribution for the cell map domain by specifying a probability value for each cell. These probabilities form an initial probability vector p(0). By definition, the probability vector elements should sum to unity. The Markov model (3.1) can then be used to simulate the evolution of the probability vector as time increases. The probability distribution at step n becomes,

$$p(n) = M^n\, p(0) \qquad (3.2)$$

It is worth noting that M does not need to be re-computed for different initial probability distributions. As a result, (3.2) allows general stochastic simulations of the nonlinear system to be performed using the linear Markov chain model.

In analyzing cell-to-cell map properties, Hsu devised ways to organize and sort the domain cells using characteristics of solutions that enter or leave the cells. Two of the categories that are most relevant to this research are absorbing cells and transient cells. A *transient cell* has the property that once a solution leaves the cell, it does not return. An *absorbing cell* has the property that once a solution enters the cell, it does not leave. The sink cell, for example, is considered an absorbing cell.

It is possible to locate the absorbing cells by studying the long-term behavior of the Markov chain. Assume that the probability distribution reaches a steady state such that $p(n) = p(n+1) = p^*$ for large n. It is reasonable to expect that only absorbing cells will contribute to the steady-state probability distribution. From (3.1), one can see that the steady-state probability distribution will satisfy,

$$p^* = M\, p^* \qquad (3.3)$$

or,

$$[I - M]\, p^* = 0 \tag{3.4}$$

Equation (3.4) reveals that any non-zero steady-state probability distribution must also be an eigenvector of M. These special eigenvectors are associated with eigenvalues equal to unity. The non-zero elements of p* point to the absorbing cells in the map.

Once the absorbing cells have been found, the probability vector can be sorted into absorbing and transient cells. Let p_a represent the probability vector elements associated with absorbing cells and p_t represent those associated with transient cells. The Markov chain can then be written in the following manner,

$$\begin{bmatrix} p_a(k+1) \\ p_t(k+1) \end{bmatrix} = \begin{bmatrix} M_{aa} & M_{at} \\ 0 & M_{tt} \end{bmatrix} \begin{bmatrix} p_a(k) \\ p_t(k) \end{bmatrix} \tag{3.5}$$

The zero submatrix in (3.5) stems from the fact that once a solution enters an absorbing cell, it will not move back into a transient cell.

Studying the behavior of the transient cells as they enter the absorbing cells reveals the stability of the nonlinear system represented by the cell map. Using (3.2) and (3.5), one can derive the following expression for the transient cell probability distribution at step n,

$$p_t(n) = M_{tt}^{\,n}\, p_t(0) \tag{3.6}$$

The absorbing cell probability distribution at step n+1 is,

$$p_a(n+1) = M_{aa}\, p_a(n) + M_{at}\, p_t(n) \tag{3.7}$$

By combining (3.6) and (3.7), we have that,

$$p_a(n+1) = M_{aa}\, p_a(n) + M_{at}\, M_{tt}^{\,n}\, p_t(0) \tag{3.8}$$

The matrix expression $M_{at}\, M_{tt}^{\,n}$ represents the conditional probability that a solution starting in a transient cell will pass into an absorbing cell at time n+1. The probability that a solution starting in a transient cell will *eventually* pass into an absorbing cell is defined by the matrix R,

$$R = \sum_{i=0}^{\infty} M_{at} M_{tt}^{i} = M_{at} \sum_{i=0}^{\infty} M_{tt}^{i} \tag{3.9}$$

Recall that the unit amplitude eigenvalues of M serve to locate the absorbing cells. Therefore, M_{aa} will contain unit amplitude eigenvalues and the remaining eigenvalues, which are also the eigenvalues of M_{tt}, will have amplitude less than unity. This result means that the series shown in (3.9) will converge such that,

$$R = M_{at}\, (I - M_{tt})^{-1} \tag{3.10}$$

The list of transient cells that have a high probability of moving into an absorbing cell determine the *basin of attraction* for the absorbing cell.

Transient cells located near the edge of an attraction basin would presumably take a longer time to reach the absorbing cells. The expected absorption time can be used to quantify this relationship. The expected absorption time matrix Q is computed by

$$Q = M_{at} \sum_{i=0}^{\infty} i\, M_{tt}^{i} \tag{3.11}$$

The matrix series in (3.11) will also converge and the result is given by,

$$Q = M_{at}\,(I - M_{tt})^{-2} \tag{3.12}$$

The (i,j) elements of Q represent the expected time for solutions from transient cell j to pass into absorbing cell i.

Table 3.1 outlines the recommended method for stability analysis using generalized cell maps. The cell map matrix M will be large and will have relatively few nonzero entries. Initial results have shown that the density of M (the number of non-zero elements divided by the total number of elements) is typically less than five percent. Fortunately, there are several sparse matrix software libraries available including those in MATLAB.[13]

Table 3.1 Cell Map Analysis

Step 1:	Find the unit amplitude eigenvalues of M as well as their associated eigenvectors p*. The non-zero elements of p* point to the absorbing cells.
Step 2:	Sort the probability cell vector such that M is decomposed into the form of (3.5). This manipulation defines M_{at} and M_{tt}.
Step 3:	Compute the probability of absorption matrix R and the expected absorption time matrix Q using (3.10) and (3.12), respectively.

3.3 Cutting Planes

A Poincare' section is a hyperplane located within a state-space. The hyperplane essentially divides the state-space into two regions. Stability analysis using Poincare' sections considers a large number of trajectories that are started on the hyperplane and followed until they return to the hyperplane. The point of intersection is called the first return. Patterns that emerge from plots of the first return points are often used to describe the global stability properties of the nonlinear system.

Levitas, Weller, and Singer proposed an analysis method whereby simple cell mapping is applied to Poincare' sections.[14] A simple cell map is created on the Poincare' section hyperplane instead of the entire state-space. This manipulation means that the dimension of each cell is reduced by one and a significant computational savings is achieved. This fundamental concept is extended further in this research by considering multiple intersecting hyperplanes to yield even greater reductions.

A hyperplane will be called a *cutting plane* in this report because its intent is to cut or divide a space.[15] A single cutting plane in a d-dimensional state-space x is described by the equation,

$$e_j^T x = g_j \tag{3.13}$$

where e_j^T is a row vector and g_j is a scalar constant. The intersection of cutting planes satisfies a set of equations,

$$E^T x = g \tag{3.14}$$

where,

$$E^T = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \end{bmatrix} \qquad\qquad g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \end{bmatrix}$$

The intersection of cutting planes will produce a subspace of reduced dimension as long as the rows of E^T are independent. This condition is readily checked by making sure that E^T has full rank. For example, if E^T has d independent rows, than the solution to (3.14) represents a unique point x in d-dimensional space.

Figure 3.2 depicts a three-dimensional space with two cutting planes. The two cutting planes are defined as:

$$e_1^T x = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [0] = g_1 \tag{3.15}$$

$$e_2^T x = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [1] = g_2 \tag{3.16}$$

The equation for the intersection between the two planes is given by,

$$\begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.17}$$

By solving the system of equations above, one can see that the intersection between the two cutting planes is represented by the one-dimensional line, $x^T = (x_1, 1, 1)$.

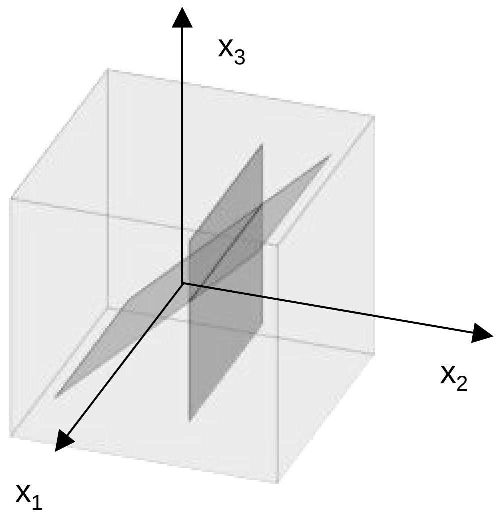


Figure 3.2 Intersection of Two Cutting Planes

When matrix E^T contains d-2 independent rows, it creates a 2-dimensional subspace within a d-dimensional state-space. A generalized cell-to-cell map will be created in this two-dimensional subspace. The coordinates of the two-dimensional subspace will be denoted y_1 and y_2. These coordinates will be related to the state vector x such that,

$$f_1^T x = y_1 \tag{3.18}$$

$$f_2^T x = y_2 \tag{3.19}$$

or,

$$F^T x = y \tag{3.20}$$

with,

$$F^T = \begin{bmatrix} f_1^T \\ f_2^T \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

By combining (3.14) and (3.20), the following system of equations results,

$$\begin{bmatrix} E^T \\ F^T \end{bmatrix} x = \begin{bmatrix} g \\ y \end{bmatrix} \tag{3.21}$$

where E^T is a (d-2)×d matrix, g is a (d-2)×1 column vector, F^T is a 2×d matrix, and y is a 2×1 column vector.

Equation (3.21) gives the definition of the fundamental subspaces that will be used in cell-to-cell map construction and analysis. E^T and g serve to create a two-dimensional subspace within the d-dimensional state-space. F^T defines the coordinates in the two-dimensional subspace that will be used to define the cell map.

In general, any E^T and F^T can be chosen as long as the rows of combined matrix formed in (3.21) are independent. This condition implies that E^T and F^T define a unique one-to-one transformation of the state vector x to another space represented by g and y. In this new d-dimensional space, d-2 coordinates are fixed by the elements of g and the remaining 2 coordinates (defined by y) are allowed to vary.

Cell-to-cell map construction is simplified if E^T and F^T are chosen to form an orthogonal transformation. If E^T and F^T represent an orthogonal transformation, then the following conditions will also hold,

$$\begin{bmatrix} E^T \\ F^T \end{bmatrix} \begin{bmatrix} E & F \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{3.22}$$

$$\begin{bmatrix} E & F \end{bmatrix} \begin{bmatrix} E^T \\ F^T \end{bmatrix} = I \tag{3.23}$$

An orthogonal transformation essentially restricts the transformation matrix to rotations and permutations. This restriction may seem limiting at first, but it helps to provide a framework for stability analysis. The elements of b and y are restricted to rotations or permutations of the physical variables in the state vector x so their domains are much easier to establish.

Using (3.23), one can solve (3.21) in terms of the state vector x. The result is given by,

$$x = Eg + Fy \tag{3.24}$$

Equation (3.24) above illustrates how trajectories are started from within the domain cells. A point inside a given 2-dimensional domain cell specifies y. Equation (3.24) is then solved to find x, which is the initial condition used to start the nonlinear trajectory in the d-dimensional state-space. The recommended generalized cell-to-cell map construction technique is summarized by the four-step procedure listed in Table 3.2.

Selection of appropriate cutting planes is not a trivial task and requires insight into the fundamental system dynamics. The process is equivalent to establishing a specific operating condition before a reduced-order model is created by hand. Simulating single trajectories over a range of operating conditions can identify general regions of interest within the state-space. Further insight can be gained by creating an initial velocity map of the subspace. The velocity of any point in the cell subspace can be found by differentiating (3.20),

$$\dot{y} = F^T \dot{x} \tag{3.25}$$

Let the nonlinear system differential equations be written in standard form,

$$\dot{x} = \sigma(x) \tag{3.26}$$

where $\sigma(x)$ is a vector function. The velocity in the subspace becomes,

$$\dot{y} = F^T \sigma(x) \tag{3.27}$$

By combining (3.24) and (3.27), the following expression results,

$$\dot{y} = F^T \sigma(Eg + Fy) \tag{3.28}$$

Selecting a point inside each domain cell and using (3.28) to find the velocity at that point creates an initial velocity map. The velocity vector will have two components since y represents a two-dimensional subspace. The velocity vector components can be plotted using arrows to yield a picture of the initial flow of trajectories as they leave the subspace.

Table 3.2 Two-Dimensional Cell-to-Cell Map Construction

Step 1:	E^T and F^T are chosen to define an orthogonal transformation that satisfies (3.22) and (3.23).
Step 2:	The vector g is selected to specify a two-dimensional subspace for analysis.
Step 3:	The domain of the two-dimensional subspace (defined by y) is divided into cells.
Step 4:	For each domain cell i, Equation (3.24) is used to generate a large number of initial conditions from within the cell. These trajectories are followed for a specified time period. The domain cell j in which trajectory ends is found using (3.20). The generalized cell map entry M(j,i) is the number of trajectories that end in cell j divided by the total number of trajectories that were started from cell i.

3.4 Cell Mapping Example

The cell mapping and cutting plane analysis method will be illustrated using an example based upon the well-known Lorenz model.[16] This model represents a fluid convection process and is described by the following nonlinear differential equations:

$$\dot{x}_1 = -\upsilon_1(x_1 - x_2) \tag{3.29}$$
$$\dot{x}_2 = \upsilon_2 x_1 - x_2 - x_1 x_3 \tag{3.30}$$
$$\dot{x}_3 = x_1 x_2 - \upsilon_3 x_3 \tag{3.31}$$

The Lorenz model incorporates many complicated behaviors dependent upon the values of the parameters υ_1, υ_2, and υ_3. The parameter set used in this example will be the original set studied by Lorenz: $\upsilon_1 = 10$, $\upsilon_2 = 8/3$, and $\upsilon_3 = 28$. With this set of parameters, the Lorenz model has three equilibrium points. The first equilibrium point is located at the origin. A linearized model formed about this point will have one positive, real eigenvalue and two negative, real eigenvalues. The second and third equilibrium points are located at $x^T = (8.5, 8.5, 27)$ and $x^T = (-8.5, -8.5, 27)$. A linearized model formed about either of these points will have one negative, real eigenvalue and a pair of complex conjugate eigenvalues with positive real parts.

Analysis of the Lorenz model will begin by defining a two-dimensional subspace of the original three-dimensional state-space using the following cutting plane:

$$e_1^T = [\ 0\ \ 0\ \ 1\ ] \qquad\qquad g_1 = 27 \tag{3.32}$$

This cutting plane defines a two-dimensional subspace such that $x_3 = 27$. This subspace passes through the second and third equilibrium points. The dimensions of the subspace are defined such that $y_1 = x_1$ and $y_2 = x_2$; therefore,

$$f_1^T = [\ 1\ \ 0\ \ 0\ ] \qquad\qquad f_2^T = [\ 0\ \ 1\ \ 0\ ] \tag{3.33}$$

The two-dimensional plane is divided into four hundred cells. The domain of interest is selected such that $-20 < y_1 < 20$ and $-20 < y_2 < 20$. The generalized cell-to-cell map is found using one hundred trajectories from each domain cell. Each trajectory was followed for 0.7 seconds using an integration step size of 0.02 seconds. The resulting Markov matrix has 2,702 nonzero elements for a density of about 1.7%.

The generalized cell-to-cell map for the $x_3 = 27$ cutting plane has two absorbing cells. As expected, the absorbing cells are located near the second and third equilibrium points. These cells are marked in Figure 3.3 as *v* and *w*, respectively. Shading the transient cells in this figure depicts the probability of absorption into cell *v*. A dark-shaded transient cell indicates a higher probability of being absorbed. The darkest region in Figure 3.3 therefore describes the basin of attraction for *v*. Conversely, the lightest shaded region defines the basin of attraction for equilibrium point *w*. The maximum expected absorption times for the transient cells are shown in Figure 3.4. A darker shade indicates a longer expected absorption time.

The results shown in Figures 3.3 and 3.4 indicate that the domains of attraction for the absorbing cells *v* and *w* are not defined with absolute certainty. For example, there are a large number of transient cells in Figure 3.3 that are neither a dark nor a light shade. These transient cells have probabilities of absorption closer to one-half. There are also two pronounced bands of darker shaded transient cells in Figure 3.4 that indicate longer absorption times. One might expect these bands to be located closer together and to define a more definite boundary between the two domains of attraction. The reason that this uncertainty exists along the basin of attraction boundary is that solutions of the Lorenz system can cross over the boundary.

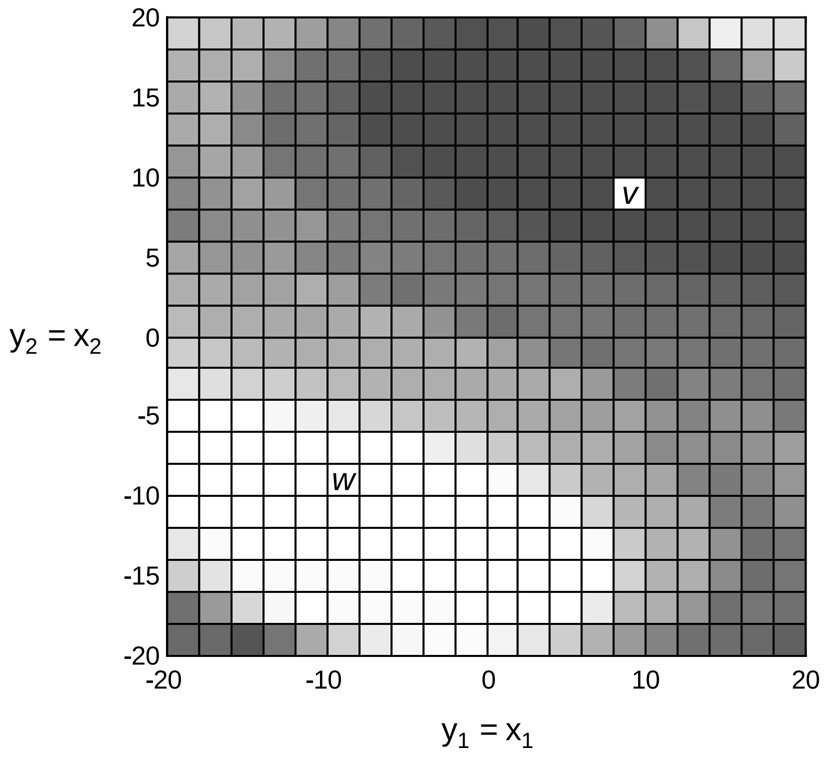


Figure 3.3 Absorption Probability for Absorbing Cell *v*

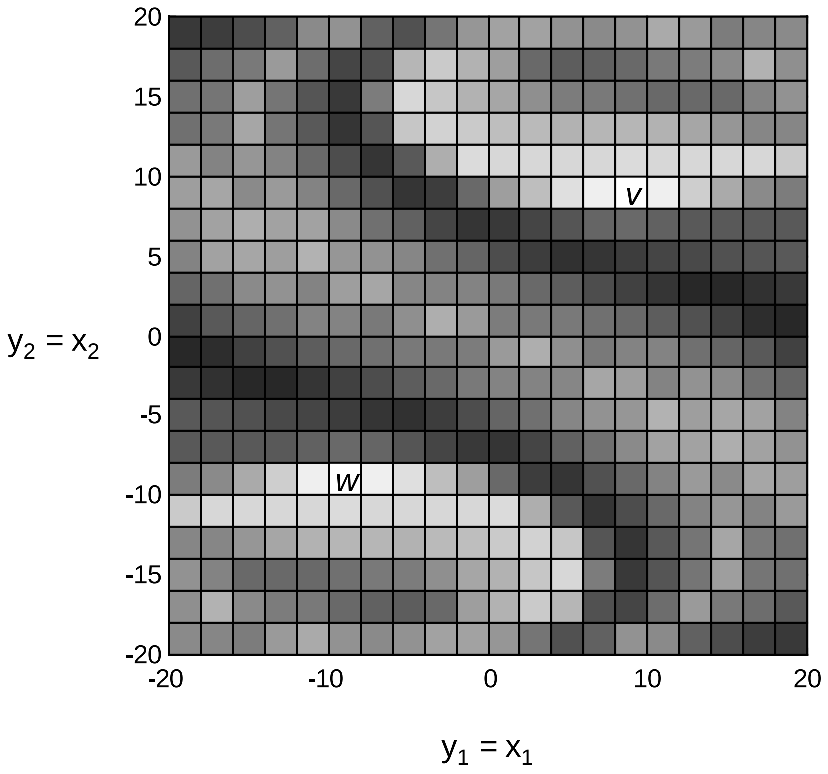


Figure 3.4 Expected Absorption Times of the Lorenz System

Figure 3.5 shows three sample trajectories of the Lorenz system. The initial point of each trajectory is marked with a large circle. One can see that two of the solutions are drawn to absorbing cell *v* and remain near the cell. The third trajectory, marked by an open circle, is initially attracted to cell *v* and encircles it three times. The trajectory then breaks from *v* and becomes attracted to *w*. It encircles point *w* three times before returning to the vicinity of cell *v*. The path of this trajectory helps to explain why the basin of attraction boundary is fairly wide. Solutions that start in the transient cells located in this boundary region can be attracted to both absorbing cells. The generalized cell-to-cell map predicts the "strange attractor" by assigning absorption probabilities close to one-half in a wide boundary region.

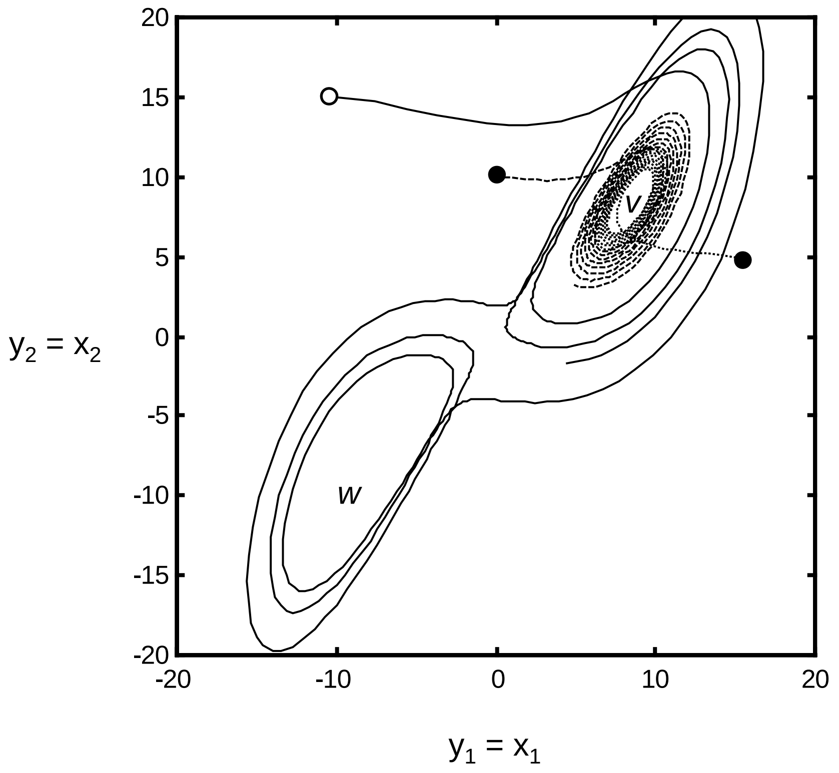


Figure 3.5 Trajectories of the Lorenz System

4. General Aviation Aircraft Application

The general aviation configuration is modeled after the Grumman Yankee aircraft. A slightly modified version of this aircraft has been extensively tested in the NASA Langley wind tunnels as well as in-flight. Wind tunnel testing included static and rotary balance wind tunnel tests and flight testing included a large number of spin entries.[17-19] The availability of this extensive test data set makes this configuration an ideal candidate for demonstrating the nonlinear lifting line and cell mapping programs.

4.1 GA Aerodynamic Model

The GA aircraft geometry is described in Table 4.1. It should be noted that the information shown in Table 4.1 is nearly all that is needed to represent the aircraft in the nonlinear lifting line program. With the exception of the airfoil sections, most of the geometric information can be obtained from simple three-view sketch of the aircraft. The remaining information that is needed (and not shown in Table 4.1) is the relative locations of the aerodynamic surfaces. The coordinate system is aligned in the plane of symmetry and the origin is defined at the nose of the aircraft. The reference point for resolving the aerodynamic forces and moments is the center-of-gravity, located 6.5 ft behind the origin.

The wing is divided into two distinct surfaces and the vortex structure for each wing surface is solved separately. The separate solutions are intended to represent the effect of the fuselage in this low-wing configuration. Each wing surface is divided into eight sections and five rows of wake vortices are used for each surface. The horizontal tail is also split into two separate surfaces, with four sections for each surface. The vertical tail has four sections as well. The wing airfoil section is the NACA 64_2-415. The tail surfaces are modeled using the NACA 0012 section (the actual Yankee utilizes a NACA 65_1-012 section). Figure 4.1 illustrates the vortex system and physical layout of the general aviation aircraft configuration. The angle-of-attack is equal to two degrees in this figure and the horizontal tail surfaces are immersed in the wing wake.

The static lift and drag coefficients for the GA configuration are compared in Figure 4.2. The solid lines in this figure are the coefficients predicted by the nonlinear lifting line program and fitted to analytical functions. The open circles and crosses represent wind tunnel data given in Reference [18]. The wind tunnel data marked by open circles was collected at a Reynolds number of Re = 288,000 while the crosses were collected at Re = 3,450,000. This figure indicates that the lifting line modeling procedure offers a reasonable approximation to the wind tunnel data over a large angle-of-attack range. The largest differences occur in the lift coefficient near stall. The resolution of the lifting line program in this angle-of-attack range is highly dependent upon the number of sections used to model the wing. On the other hand, the wind tunnel results show significant variation in this area due to Reynolds number.

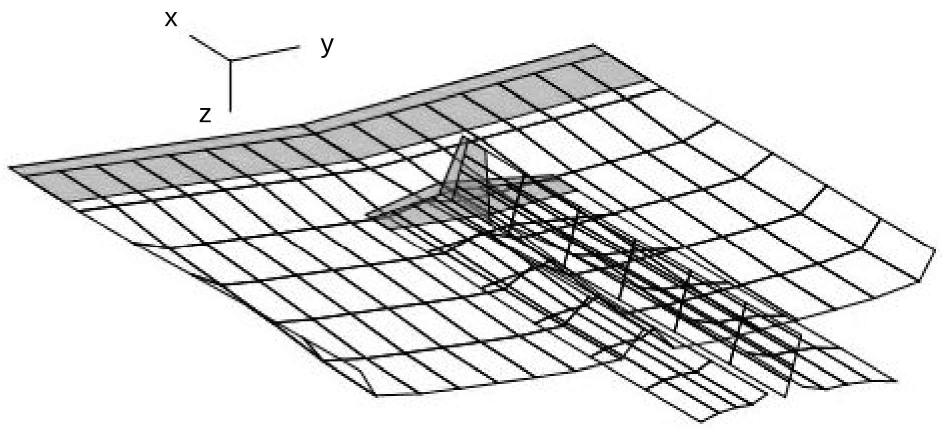
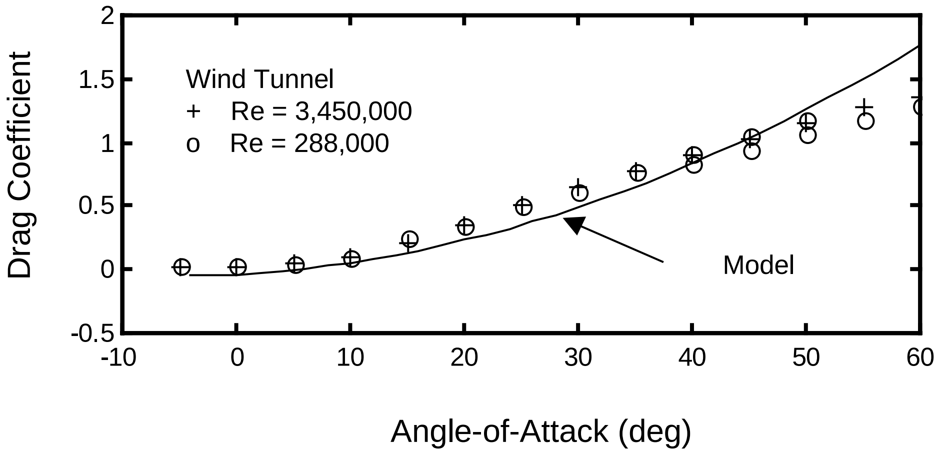


Figure 4.1 Vortex System for the GA Configuration



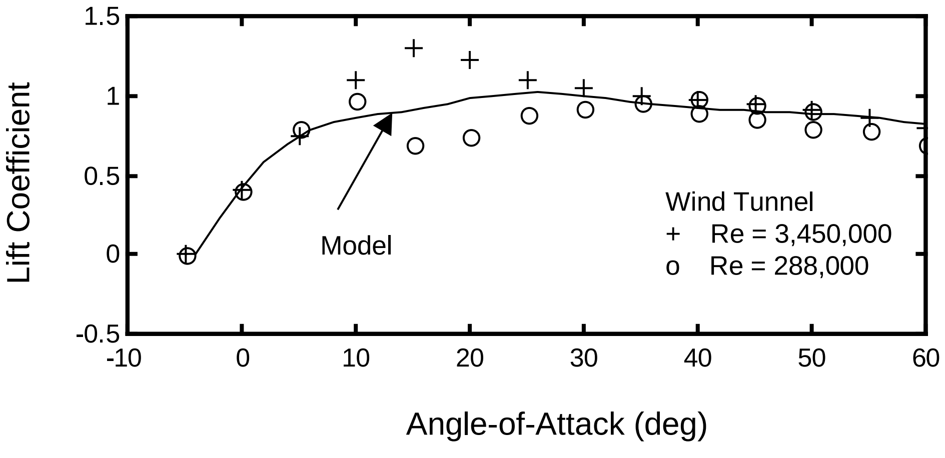


Figure 4.2 Lift and Drag Coefficients

Table 4.1 GA Configuration Geometry

Overall Dimensions	
Wing Span	24.5 ft
Length	19.0 ft
Wing Area	98.1 ft^2
Mean Aerodynamic Chord	4.0 ft
Aspect Ratio	6.1
Mass Properties	
Weight	1577 lb
Center-of-Gravity Location (from wing leading edge)	1.0 ft
Ixx	596 slug-ft^2
Iyy	738 slug-ft^2
Izz	1268 slug-ft^2
Right and Left Wing Surfaces	
Span	12.2 ft
Root Chord	4.0 ft
Tip Chord	4.0 ft
Dihedral	5.0 deg
Sweep	0.0 deg
Number of Sections	8
Airfoil Section Type	NACA 64_2-415
Right and Left Horizontal Tail Surfaces	
Span	3.9 ft
Root Chord	3.6 ft
Tip Chord	1.7 ft
Dihedral	0.0 deg
Sweep	20.0 deg
Number of Sections	4
Airfoil Section Type	NACA 0012
Vertical Tail Surface	
Span	4.1 ft
Root Chord	3.6 ft
Tip Chord	1.6 ft
Dihedral	90.0 deg
Sweep	22.0 deg
Number of Sections	4
Airfoil Section Type	NACA 0012

Elevator control effectiveness is illustrated in Figure 4.3 for the GA configuration. The solid line in this figure is the pitching moment coefficient predicted by the model for an elevator control deflection of -25 deg. The dotted line represents the model result for an elevator deflection of +15 deg. Wind tunnel results are also shown for deflections of -25 deg (crosses) and +15 deg (open circles). The wind tunnel results were collected at Re = 3,450,000. The model results are comparable to the wind tunnel results but predict lower elevator effectiveness throughout the angle-of-attack range. This discrepancy is most likely a result of the airfoil section characteristics used to model the horizontal tail. Typical airfoil section data includes the effect of a 20% chord flap. The elevator width of the GA configuration is about 30% chord, so a factor of 1.3 was applied to the elevator control deflection to model this difference (as described in Reference [20]). However, this effectiveness factor can vary with control deflection and mechanization (plain flap, split flap, etc.) so a larger factor could be justified to better match the wind tunnel data. It should also be noted that the model results show a significant reduction in elevator effectiveness at higher angles-of-attack. The wind tunnel results, on the other hand, show virtually no change in elevator effectiveness as angle-of-attack increases.

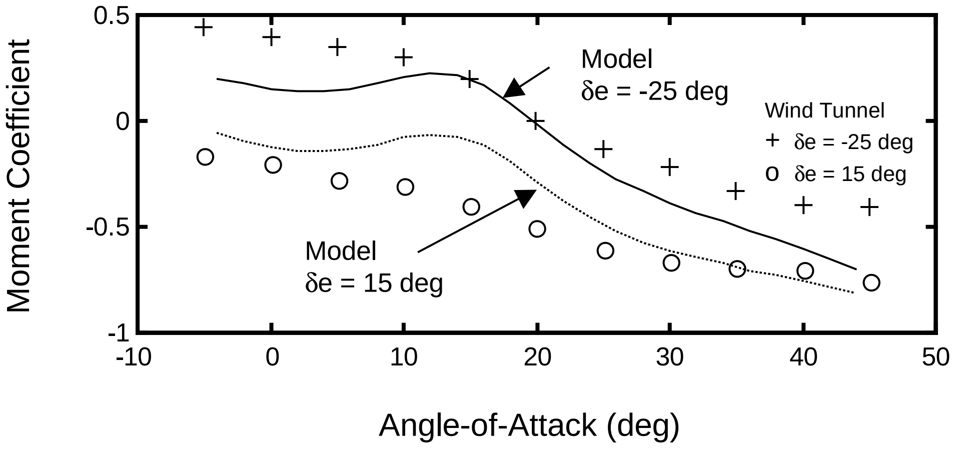


Figure 4.3 Elevator Control Effectiveness

The variation in yaw moment coefficient with angle-of-attack is shown in Figure 4.4 for sideslip angles of 5 and 20 deg. The solid line in this figure is the model prediction for 5 deg sideslip while the dotted line is the model result for 20 deg sideslip. Wind tunnel test results are shown using crosses (β = 5 deg) and open circles (β = 20 deg) for a Reynolds number of Re = 3,450,000. These results demonstrate further that the model predictions are reasonably close to the wind tunnel results.

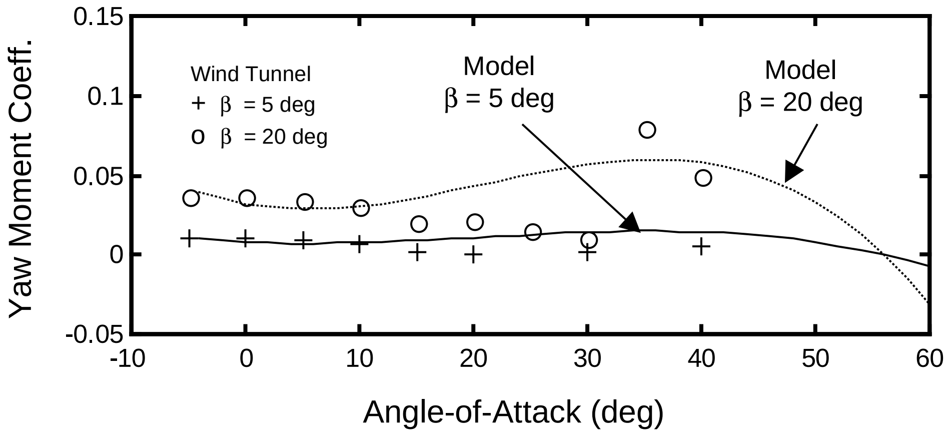


Figure 4.4 Yaw Moment Coefficient

The lifting line program was used to determine each of the coefficient functions as described in Section 2.4 (see Table 2.2). A complete list of the aerodynamic coefficient functions is given in Appendix A. The coefficient functions were fitted to an angle-of-attack range from -4 to 60 deg. The sideslip coefficients were developed assuming a maximum sideslip of 10 deg. Roll and yaw rate coefficients assumed a maximum nondimensional value of 0.4. The maximum expected nondimensional pitch and angle-of-attack rates are set to 0.02. The angle-of-attack rate coefficients are computed assuming a nondimensional delay constant equal to 5.0. The control input variables were obtained using a deflection of 20 deg. The only adjustment that has been made from the lifting line program results is to increase the drag coefficient at zero angle-of-attack by a value of 0.18. This change was made so that the engine power required for level flight matched the values reported by NASA during flight testing.

4.2 Stall, Spin, and Recovery Analysis

NASA flight testing resulted in a great deal of information about the stall and spin characteristics of this GA configuration.[17] The aircraft was found to have two steady spin modes. The dominant spin mode was a moderately steep spin occurring at an angle-of-attack between 40 and 50 deg. Spin entry was typically achieved by simultaneously applying full elevator and rudder deflections with the power set to idle. Small variations in the steady spin rate and attitude could be achieved if the ailerons were deflected with or against the spin. The nominal spin rate was found to be about 147 to 155 deg/s at a velocity in the range of 120 to 130 ft/s. The nominal spin recovery procedure was to deflect the rudder against the spin and then follow with down elevator. Recovery could also be achieved using neutral controls, but complete recovery was slower than the nominal procedure.

An example spin trajectory, obtained using the analytical model of the GA configuration, is shown in Figures 4.5 and 4.6. Figure 4.5 shows the model angle-of-attack trajectory. Spin entry controls are applied at 12.5 seconds into the simulation. The power is set to zero and the elevator is deflected 25 deg trailing-edge-up. The rudder is set to -25 deg and the ailerons are deflected +20 deg so that the aircraft is forced to roll into the spin. It takes about 10 seconds for the aircraft to stabilize in a right spin at a 45 deg angle-of-attack. The elevator, rudder, and ailerons are neutralized at about 32 seconds into the simulation and the aircraft quickly recovers.

Figure 4.6 shows the roll, pitch, and yaw angular rates of the GA model during the spin simulation. The steady spin rate of 164 deg/s yields a normalized spin rate of about 0.3. The steady roll and the yaw rates are about 114 deg/s. These rates are slightly higher than those found during flight testing. The aircraft model is pitched 44 deg nose-down during the steady spin and the wings are almost level. The spin radius is about 5 ft and is consistent with the characteristics observed in flight.

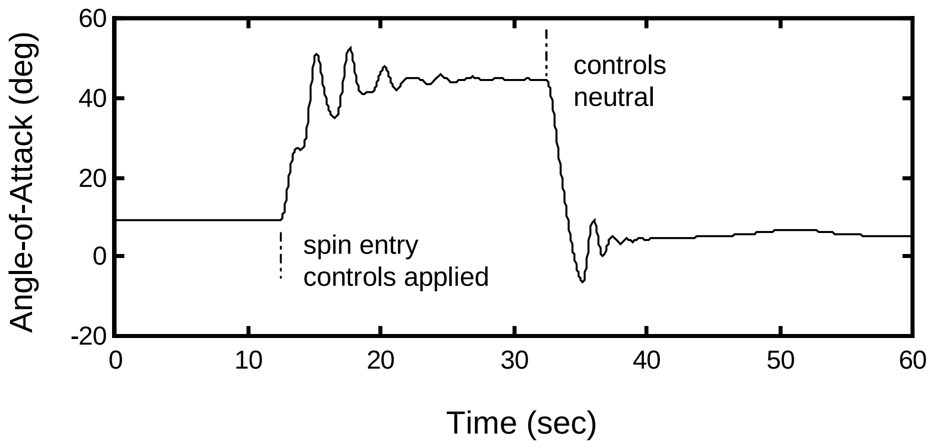


Figure 4.5 Angle-of-Attack During Spin Maneuver

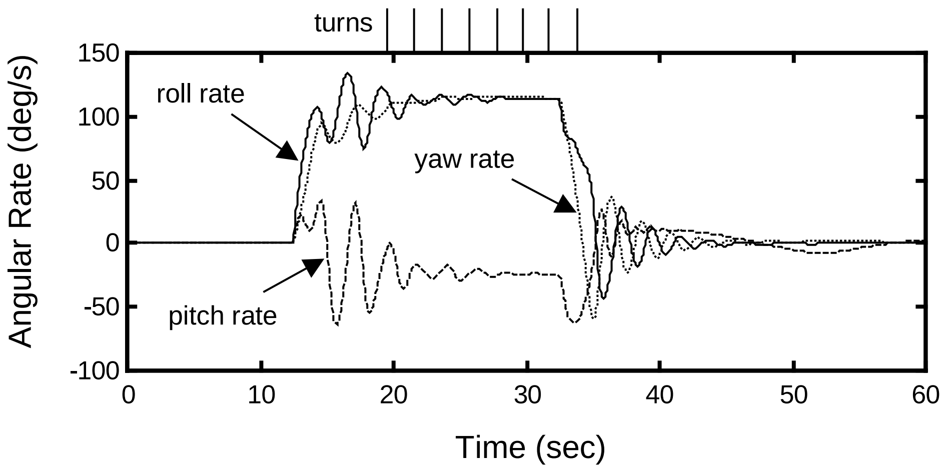


Figure 4.6 Angular Rates During Spin Maneuver

The purpose of this example analysis is to illustrate how the cell-to-cell mapping approach is used to investigate the motion and stability of the GA configuration model during stall, spin, and recovery. A general discussion of aircraft spinning dynamics will be given to begin the example. Figure 4.7 illustrates the stall, spin, and recovery phases that are evident in the simulated GA aircraft model trajectory. The abscissa and ordinate of Figure 4.7 are defined, respectively, as follows:

$$y_1 = \alpha \tag{36}$$

$$y_2 = \frac{p}{\sqrt{2}} + \frac{r}{\sqrt{2}} \tag{37}$$

The coordinates y_1 and y_2 are chosen to be consistent with the cell-to-cell maps that will be presented later. The y_1 coordinate is angle-of-attack and the y_2 coordinate approximates spin rate.

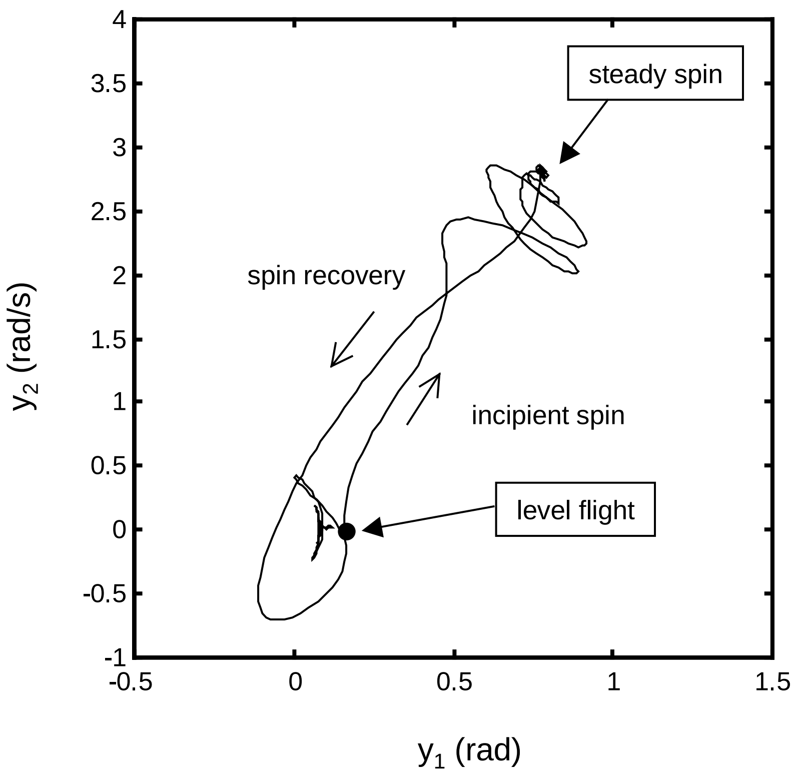


Figure 4.7 Spin Maneuver Phases

With the aircraft lateral controls undeflected, the aircraft will seek a stable equilibrium point that depends upon altitude, power setting, and elevator deflection. An equilibrium or fixed point is defined as a constant solution of the system differential equations. In this case, a 9 deg angle-of-attack, airspeed of 145 ft/s, and an altitude of 10,000 ft specify the level-flight equilibrium point. A linear stability analysis about this equilibrium point will yield the well-known dynamic modes of flight such as the short-period and Dutch-roll.

When the pilot applies abrupt spin entry controls, the level-flight equilibrium point disappears and a new critical point appears at the conditions associated with the steady spin. It can be argued that the steady spin is not a true equilibrium solution because the heading and altitude rates are obviously not equal to zero. For this reason, the steady spin will be called a critical point instead of an equilibrium point to highlight this distinction.

If the airplane is inside the basin of attraction for steady spin critical point, it will be drawn towards it. This phase of the trajectory is called the incipient spin. The aircraft is not under pilot control during the incipient spin because the control surface deflections are constant. The motion of the aircraft is governed by the characteristics of the spin basin of attraction. The motion in this phase appears to be very erratic as the aircraft literally bounces through the state-space towards the spin critical point. The aircraft eventually converges to the spin critical point.

From a dynamics perspective, the spin recovery is the opposite of the spin entry. When the pilot neutralizes the elevator, rudder, and aileron controls, the spin critical point is annihilated and a new critical point emerges. This new critical point is defined by angle-of-attack, sideslip, and angular rates nearly equal to zero. It is not an equilibrium point in the strict sense because the altitude may not be constant. The aircraft will recover from the spin as long as spin flight conditions are within the basin of attraction for the new critical point near the origin. Once again, the pilot does not control the actual spin recovery trajectory because the control deflections are constant.

The preceding description of aircraft spin dynamics was drawn from the existing subject literature.[21] A similar analysis using the cell-to-cell mapping technique should yield an even more revealing picture of the underlying dynamics. The GA aircraft model will be represented using fifteen state variables. The state variables are defined in Table 4.2 for reference. Three state variables are used to represent aircraft translational velocity and three are used for angular velocity. The attitude of the aircraft is represented using four quaternions. The only position variable is altitude. The final four state variables represent the control surface deflections.

Several generalized cell-to-cell maps will be constructed and presented in the following discussion. Each cell map contains four hundred cells. One hundred trajectories were started from each domain cell and each trajectory was followed for 10 integration steps. The integration step size is 0.05 seconds in all cases. Consequently, the Markov model that is constructed from each cell-to-cell map represents a time increment of about 0.5 seconds.

Table 4.2 GA Model State Variables

Variable	Description
V	airspeed (ft/s)
β	sideslip (rad)
α	angle-of-attack (rad)
p	roll rate (rad/s)
q	pitch rate (rad/s)
r	yaw rate (rad/s)
s_0	quaternion
s_1	quaternion
s_2	quaternion
s_3	quaternion
h	altitude (ft)
δ_E	elevator deflection (rad)
δ_A	aileron deflection (rad)
δ_R	rudder deflection (rad)
T	thrust (lb)

The dimension of the GA model state-space is fifteen, so thirteen cutting planes are required to define a two-dimensional subspace for cell mapping. The first subspace will be chosen to pass through the level-flight equilibrium point. The equations for the thirteen cutting planes are listed in Table 4.3. The first twelve planes represent cuts along a single dimension. The last cutting plane is placed perpendicular to the y_2 coordinate. This cutting plane has the effect of forcing roll rate and yaw rate to be equal in the subspace.

Table 4.3 Level Flight Subspace Cutting Planes

Plane	Equation
1	$V = 145$
2	$\beta = 0$
3	$q = 0$
4	$s_0 = 0.997$
5	$s_1 = 0$
6	$s_2 = 0.077$
7	$s_3 = 0$
8	$h = 10000$
9	$\delta_E = -0.1$
10	$\delta_A = 0$
11	$\delta_R = 0$
12	$T = 400$
13	$\frac{p}{\sqrt{2}} - \frac{r}{\sqrt{2}} = 0$

Two absorbing cells were found in the generalized cell-to-cell map of the GA aircraft model using the subspace defined in Table 4.3. The absorbing cells are marked *u* in Figure 4.8. The basin of attraction for these absorbing cells includes the entire domain of interest, $-0.5 < y_1 < 1.5$ and $-1.0 < y_2 < 1.0$. In other words, the absorption probability is equal to unity for all transient cells in this domain. Figure 4.8 illustrates the maximum expected absorption time for the transient cells to move into the two absorbing cells. A darker shade indicates a longer absorption time. The darkest shade is associated with seventeen steps of the Markov chain or about eight seconds.

A velocity map of the level flight subspace is shown in Figure 4.9. The velocity map depicts the initial velocity vectors at the center of each cell. The location of the absorbing cells u are also marked on the map. One can see from Figure 4.9 that the velocity vectors generally point towards the absorbing cells.

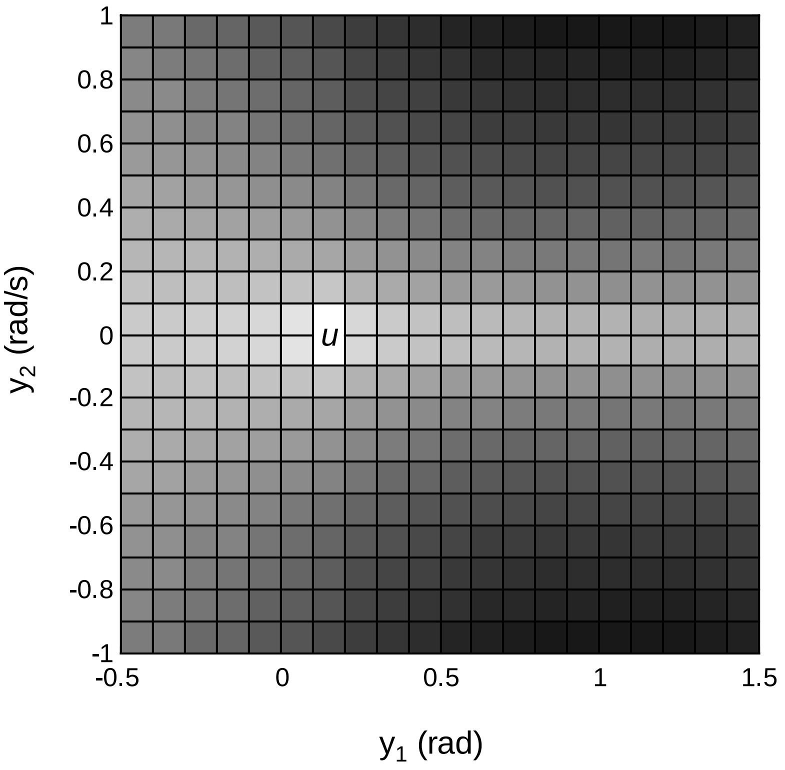


Figure 4.8 Expected Absorption Times in Level Flight Subspace

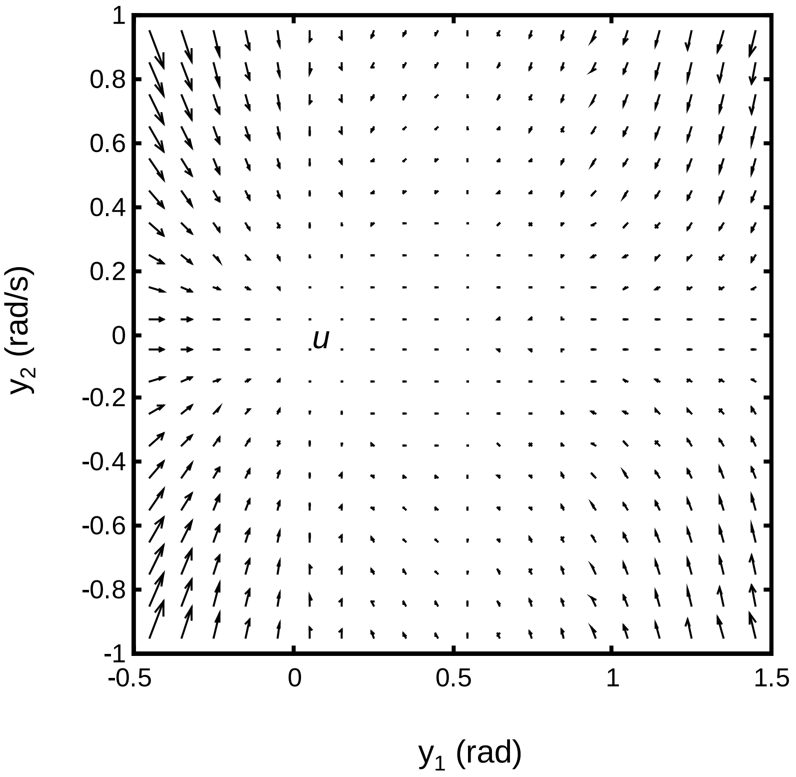


Figure 4.9 Velocity Map in Level Flight Subspace

A second set of cutting planes is chosen to define a subspace passing through the spin critical point. These cutting planes are defined in Table 4.4. Note that the control surfaces are now deflected with spin entry controls for a right-turning spin. The cutting planes associated with quaternions s_0 and s_2 are selected to define a pitch attitude of -44 deg. The airspeed has been reduced to the value representing the spin condition. Three absorbing cells were found in the cell-to-cell map defined by these cutting planes. As expected, these cells are located near the spin critical point. The basin of attraction for these cells includes the entire domain of interest, from $-0.5 < y_1 < 1.5$ to $-1.0 < y_2 < 4.0$.

Table 4.4 Spin Subspace Cutting Planes

Plane	Equation
1	$V = 116$
2	$\beta = 0$
3	$q = 0$
4	$s_0 = 0.927$
5	$s_1 = 0$
6	$s_2 = -0.375$
7	$s_3 = 0$
8	$h = 10000$
9	$\delta_E = -0.44$
10	$\delta_A = 0.35$
11	$\delta_R = -0.44$
12	$T = 0$
14	$\frac{p}{\sqrt{2}} - \frac{r}{\sqrt{2}} = 0$

Figure 4.10 depicts the expected absorption times of the transient cells in the domain of interest. The three absorbing cells associated with the spin critical point are labeled *s*. A darker shade indicates a longer expected absorption time. The darkest shade transient cell has an expected absorption time of about 8 steps of the Markov model. Since each step of the Markov model represents 0.5 sec, we can expect the aircraft to initially reach the absorbing cells in about 4 seconds.

A velocity map for the spin subspace is shown in Figure 4.11. This map gives a sense of the complicated nature of the dynamics associated with incipient spin phase. Once the spin entry controls are applied, the aircraft state will be attracted from a position near the origin to the *s* absorbing cells. The actual trajectory depends upon the direction and length of the velocity vectors along this path. For example, the velocity vectors in Figure 4.11 indicate a clockwise flow near *s*. This result is consistent with the sample spin trajectory in Figure 4.7, which shows a clockwise spiral trajectory as the aircraft approaches the spin critical point.

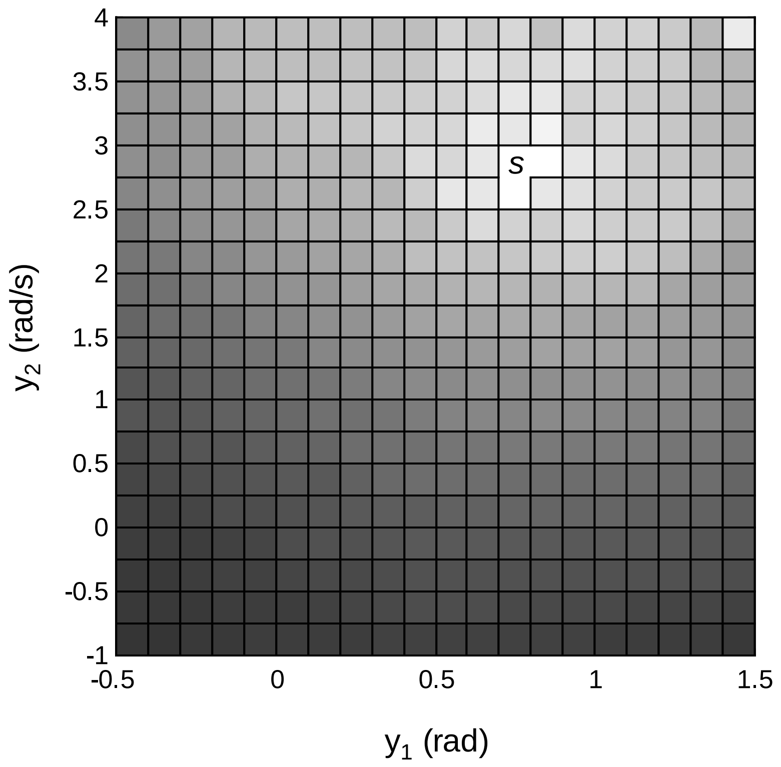


Figure 4.10 Expected Absorption Times in Spin Subspace

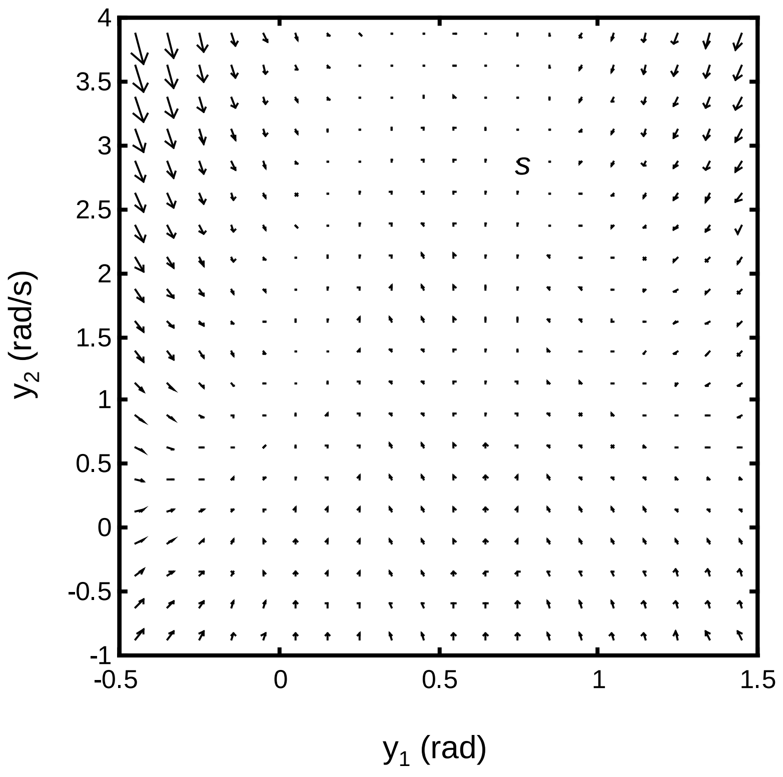


Figure 4.11 Velocity Map in Spin Subspace

Spin recovery is achieved by returning the control surfaces to a neutral position. A new cell-to-cell map results if the control state variables are neutralized in the cutting plane definitions from Table 4.4. This map has two absorbing cells located near the origin. All of the transient cells in the domain of interest are attracted to these two cells. This result confirms that the aircraft will recover from the spin using neutral controls. Figure 4.12 shows the expected absorption times for the transient cells. The absorbing cells are marked *r*. The transient cells are shaded such that a completely black cell denotes an absorption time of 12 steps of the Markov model. Consequently, the aircraft is expected to recover from the spin in about six seconds.

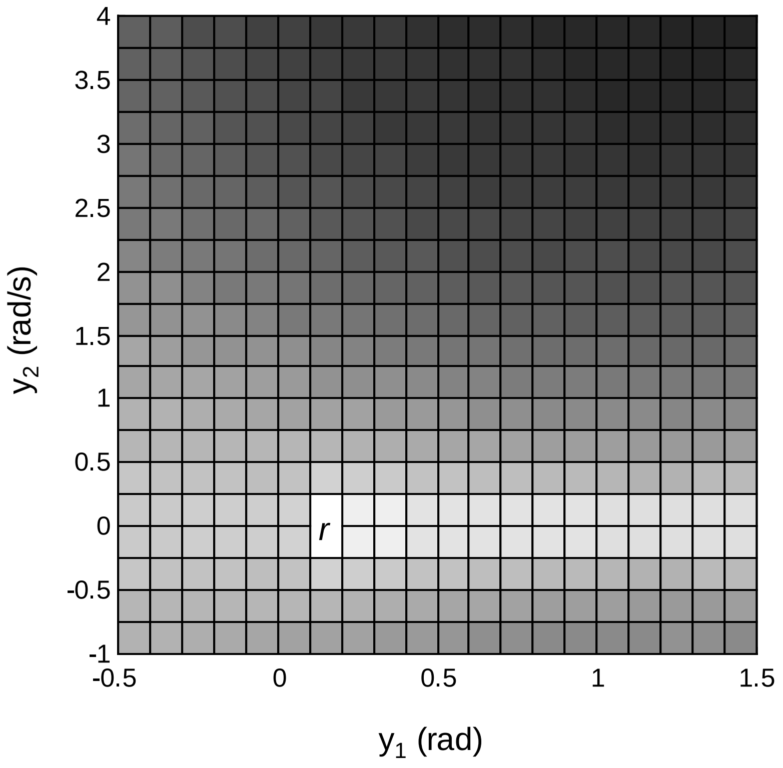


Figure 4.12 Expected Absorption Time with Controls Neutral Recovery

5. Conclusion

The nonlinear lifting program offers a number of advantages over empirical methods that have been traditionally used to construct aerodynamic models for stability and control analysis. New configurations can be constructed using only a three-view sketch of the aircraft layout. Airfoil section data can be modified to represent a wide variety of control effectors. The modeling program can be used to generate models for post-stall and spinning flight conditions. The resulting compact, state-space aerodynamic models are readily incorporated into flight dynamic simulations.

The nonlinear lifting line model is sensitive to the characteristics of the airfoil sections. There is a plethora of airfoil section data for angles-of-attack below stall, but very little information available for post-stall flight conditions. Without supporting wind tunnel data, it is difficult for the user to define these characteristics. The airfoil section template derived from NACA 0012 airfoil data provides a useful baseline, but may not be appropriate for more advanced airfoil shapes. It is theoretically possible to incorporate variations in Reynolds number into the definition of the airfoil section characteristics. It was found that the resolution of the method is inadequate in the stall region to accurately reproduce these variations. For fundamental flight dynamic studies, a Reynolds number variation could be added to the aerodynamic model as a random or deterministic uncertainty.

It is no surprise that ease-of-use comes at the expense of model fidelity. The example GA configuration model proved to be an excellent basis for quantifying the limitations of the modeling method and program. The model proved to be only moderately accurate when compared to wind tunnel data. However, the spin and recovery characteristics that are predicted for the general aviation configuration accurately reflect previous flight test experience. This result is rather remarkable when one considers the time and expense that would be required to generate an equivalent aerodynamic model from experimental data or more sophisticated computational methods.

Traditional linearization techniques, that assume small perturbations in each of the motion variables, yield models that are appropriate for a small region of the state-space centered at a given point. The cutting plane method introduced in this report assumes small perturbations in all but two dimensions of the state-space. The variables that define the two-dimensional subspace are allowed to vary over a wide range. One can therefore conclude that the cutting plane method creates a model by linearization with respect to a plane in the state-space rather than a point. It yields the exact same information as one would obtain after manually building a reduced-order model by setting a subset of state variables to constant values. The advantage of the cutting plane method is that the analysis procedures are applied directly to the full-order model in a systematic and intuitive way.

6. References

[1] Piszkin, S.T. and Levinsky, E.S., "Nonlinear Lifting Line Theory for Predicting Stalling Instabilities on Wings of Moderate Aspect Ratio," N62269-75-C-0356, Naval Air Warfare Center, June 1975.

[2] Levinsky, E.S., "Theory of Wing Span Loading Instabilities Near Stall," AGARD CP-204, *Prediction of Aerodynamic Loading*, 1976, pp. 25.1-25.16.

[3] Hsu, C.S., *Cell-to-Cell Mapping: A Method of Global Analysis for Nonlinear Systems*, Springer-Verlag, Inc., New York, 1987.

[4] Katz, J. and Plotkin, A., *Low-Speed Aerodynamics*, Second Edition, Cambridge University Press, Cambridge UK, 2001.

[5] Etkin, B., *Dynamics of Atmospheric Flight*, John Wiley & Sons, New York, 1972.

[6] McCormick, B.W., "The Prediction of Normal Force and Rolling Moment Coefficients for a Spinning Wing," NASA CR-165680, Feb. 1981.

[7] Abbott, I.H. and Von Doenhoff, A.E., *Theory of Wing Sections*, Dover Publications, Inc., 1959.

[8] Critzos, C.C., Heyson, H.H., and Boswinkle, W., "Aerodynamic Characteristics of NACA 0012 Airfoil Section at Angles of Attack from 0 to 180 Degrees," NACA TN 3361, Jan. 1955.

[9] Fan, Y. and Lutze, F.H., "Identification of an Unsteady Aerodynamic Model at High Angles of Attack," Paper No. 96-3407, AIAA Atmospheric Flight Mechanics Conference San Diego, CA, July 1996.

[10] Brandon, J.M., "Dynamic Stall Effects and Applications to High Performance Aircraft," AGARD Report No. 776, April 1991, pp. 2.1-2.15.

[11] Scharf, L.L., *Statistical Signal Processing*, Addison-Wesley Publishing Company, 1991.

[12] Morelli, E.A., "Gobal Nonlinear Parametric Modeling with Application to F-16 Aerodynamics," American Control Conference, Philadelphia, PA, June 1998.

[13] Hanselman, D. and Littlefield, B., *Mastering MATLAB 5*, Prentice-Hall, Inc., New Jersey, 1998.

[14] Levitas, J., Weller, T., and Singer, J., "Poincare'-Like Simple Cell Mapping for Non-Linear Dynamical Systems," *Journal of Sound and Vibration*, Vol. 176, No. 5, 1994, pp. 641-662.

[15] Hinneburg, A., Keim, D.A., "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering," Proceedings of the 25th Very Large Database Conference, Edinburgh, Scotland, 1999.

[16] White, M.T. and Tongue, B.H., "Application of Interpolated Cell Mapping to an Analysis of the Lorenz Equations," *Journal of Sound and Vibration*, Vol. 188, No. 2, 1995, pp. 209-226.

[17] Stough, H.P. and Patton, J.M., "The Effects of Configuration Changes on Spin and Recovery Characteristics of a Low-Wing General Aviation Research Airplane," Paper No. 79-1786, AIAA Aircraft Systems and Technology Meeting, New York, 1979.

[18] Bihrle, W., Barnhart, B., and Pantason, P., "Static Aerodynamic Characteristics of a Typical Single-Engine Low-Wing General Aviation Design for an Angle-of-Attack Range of -8 to 90 Degrees," NASA CR-2971, 1978.

[19] Hultberg R.S. and Mulcay, W., "Rotary Balance Data for a Typical Single-Engine General Aviation Design for an Angle-of-Attack Range of -8 to 90 Degrees," NASA CR-3100, 1980.

[20] McCormick, B.W., *Aerodynamics, Aeronautics, and Flight Mechanics,* John Wiley & Sons, New York, 1979.

[21] Mehra, R.K. and Carroll, J.V., "Bifurcation Analysis of Aircraft High Angle-of-Attack Flight Dynamics," Paper No. 80-1599, AIAA Atmospheric Flight Mechanics Conference, Danvers, MA, 1980.

Appendix A: GA Aircraft Aerodynamic Model

Coefficient = $k_0 + k_1 \alpha + k_2 \alpha^2 + k_3 \alpha^3 + k_4 \alpha^4$ (note: angular units in radians)

Coefficient	k_0	k_1	k_2	k_3	k_4
C_{D0}	0.14	0.30	1.36	0.0	0.0
C_{Dq}	-0.53	1.79	3.38	0.0	0.0
$C_{D\delta e}$	0.0	0.02	0.28	-1.68	1.20
$C_{Y\beta}$	-0.28	0.13	-1.72	0.0	0.0
C_{Yp}	-0.17	0.52	-0.31	0.0	0.0
C_{Yr}	0.25	-0.15	0.43	0.0	0.0
$C_{Y\delta a}$	0.03	-0.01	0.0	0.0	0.0
$C_{Y\delta r}$	0.08	-0.06	0.0	0.0	0.0
C_{L0}	0.40	4.36	-10.93	11.17	-4.15
$C_{L\dot{\alpha}}$	4.17	7.35	-29.07	17.67	0.0
$C_{L\|p\|}$	-0.29	-5.43	29.25	-42.50	19.08
C_{Lq}	5.39	-8.56	2.71	0.0	0.0
$C_{L\|r\|}$	0.05	1.54	-2.60	1.22	0.0
$C_{L\delta e}$	0.14	0.06	0.04	0.0	0.0
$C_{l\beta}$	-0.09	0.46	-0.70	0.36	0.0
C_{lp}	-0.44	1.13	-0.95	0.0	0.0
C_{lr}	0.08	0.07	-0.81	0.57	0.0
$C_{l\delta A}$	0.10	0.0	0.0	0.0	0.0
$C_{l\delta R}$	0.0	0.0	0.0	0.0	0.0
C_{m0}	0.06	-0.57	-0.53	0.0	0.0
$C_{m\dot{\alpha}}$	-2.63	-4.32	25.73	22.80	3.77
$C_{m\|p\|}$	-0.02	-0.86	2.74	-1.76	0.0
C_{mq}	-6.29	0.0	0.0	0.0	0.0
$C_{m\|r\|}$	-0.16	-2.23	6.59	-4.45	0.0
$C_{m\delta E}$	-0.40	-0.24	0.66	0.0	0.0
$C_{n\beta}$	0.10	-0.24	1.00	-0.77	0.0
C_{np}	-0.03	0.39	-0.24	0.0	0.0
C_{nr}	-0.11	0.09	-0.22	0.0	0.0
$C_{n\delta A}$	0.0	0.15	-0.09	0.0	0.0
$C_{n\delta R}$	-0.04	0.02	0.0	0.0	0.0

Appendix B: NLIFT Program Description

This appendix serves as a programmer's guide to the nonlinear lifting line software program. The NLIFT program is written in ANSI C++. It is a stand-alone program that does not require any special libraries or compiler features. The results shown in the report were generated using an engineering workstation equipped with an 866 MHz Pentium III microprocessor.

The program consists of six source code files. It is organized such that the user will work primarily with two files: main.cpp and section.cpp. The main.cpp file is used to specify the aircraft configuration for study and the section.cpp file is used to specify airfoil section information. Examples of these files are included as part of the software distribution. The other files include the source code necessary to manipulate and create class objects.

Aircraft Layout

The aircraft is represented as a combination of aerodynamic surfaces such as the wing, horizontal and vertical tails. The position of these individual surfaces is indicated using a conventional right-handed coordinate system. The x-axis points forward through the nose of the aircraft, the y-axis points in the direction of the right wing, and the z-axis points downward. This coordinate system is body-fixed.

There are two reference points that must be defined. The first point defines the origin of the coordinate system. This point is not explicitly specified in the program as all other positions are to given relative to this point. For example, if the origin is positioned at the nose of the aircraft, then the wing root position is specified as a relative distance from the nose. The second reference point is the point about which the motion variables are defined and the forces and moments are resolved. This point is usually at the aircraft center-of-gravity and must also be specified relative to the origin of the coordinate system.

Vector Class

The Vector class implements a general three-dimensional vector. This class includes member functions for most arithmetic operations such as addition, subtraction, dot and cross product. The Vector class is most often used to define the locations of aerodynamic surfaces. Force and moment calculation results are also specified using the Vector class. Consequently, the most important member function of the Vector class is the class constructor. An object of the Vector class is created using either of the following forms:

```
Vector v1;                          // creates a vector with
                                    // x = 0.0, y = 0.0, z = 0.0

Vector v2(1.0, 2.0, 3.0);           // creates a vector with
                                    // x = 1.0, y = 2.0, z = 3.0
```

Note that the components of the Vector class are private. Therefore, it is not possible to access the components of the Vector class directly. The Vector class supports standard file output so that the components of a Vector can be written to a file or the console.

```
cout << v2;                         // prints Vector v2 to console
```

Vortex Class

The Vortex class implements a ring vortex. This class is used by the Surface class to represent the bound and wake vortices.

Section Class

The Section class is used to define two-dimensional airfoil section characteristics. It is an abstract class that requires the user to implement functions for lift, drag, and moment coefficient. The member function prototypes of the Section class are:

```
double lift(const double alpha, const double flap);
double drag(const double alpha, const double flap);
double moment(const double alpha, const double flap);
```

The coefficient functions have two input variables, angle-of-attack `alpha`, and a `flap` control parameter. The control parameter must be specified in the function definition (as shown above), but it does not need to be used in the function. It is intended to represent control devices such as flaps, spoilers, and ailerons. The value of the control parameter is set for each section using a member function of the Surface class.

Surface Class

The Surface class is used to create, locate, and solve for the vortex system of an aerodynamic surface. Calling the Surface class constructor creates an aerodynamic surface. Input arguments of the Surface class constructor include the number of spanwise sections and the number of wake vortex rows. In the example below, the surface will have 4 bound vortices along the quarter-chord line and the wake will have 5 rows and 4 sections (20 ring vortices).

```
Surface rwing(4,5);                 // creates an aerodynamic surface
                                    // with 4 spanwise sections and
                                    // 5 wake vortex rows
```

After the surface has been created, its shape and position must be defined. The position of the surface, in the body-axis, is defined using the RootPt function:

```
rwing.RootPt(Vector(-5.0, 0.0, 1.0));
                                    // surface root is located at
                                    // x = -5.0, y = 0.0, z = 1.0
```

The span of the surface is defined by the Span function while the RootChord and TipChord functions define the chord length at the root and tip, respectively.

```
rwing.Span(20.0);                   // span is set to 20.0
rwing.RootChord(8.0);               // chord length at root is 8.0
rwing.TipChord(6.0);                // chord length at tip is 6.0
```

The dihedral angle specifies as a rotation of the entire surface about the -x axis. The dihedral angle is expressed in radians. A conventional vertical tail surface will have a dihedral angle of 1.57 rad.

```
rwing.Dihedral(10.0*pi/180.0);      // 10 deg dihedral angle
```

(Note that `pi` is specified as a global variable in main.hpp). The sweep angle specifies a rotation of the surface leading edge about the +z axis.

```
rwing.Sweep(45.0*pi/180.0);         // 45 deg sweep angle
```

The reference point is defined using a Vector class object:

```
rwing.RefPt(Vector(-8.0, 0.0, 0.0));
                                    // reference is located at
                                    // x = -8.0, y = 0.0, z = 0.0
```

Note that a reference point must be established for each individual surface in the airplane. However, this point is usually the same for each surface, such as:

```
Vector rcg(-8.0, 0.0, 0.0);         // rcg is the center-of-gravity
                                    // location for the airplane

rwing.RefPt(rcg);                   // reference points are set at the
lwing.RefPt(rcg);                   // same point for each surface
```

The final parameter needed for surface layout is the length of each wake vortex. This distance defines the streamwise length of the ring vortices in the surface wake. Generally, this length is set close to the value of the wing mean aerodynamic chord.

```
rwing.WakeLength(7.0);              // set wake length to 7.0
```

Once the shape and position of the surface have been defined, the surface needs to be built so that several internal geometric variables can be computed and the vortex system can be created. The Surface class member function call is:

```
rwing.Build(true);                  // build rwing, tip along +y axis
lwing.Build(false);                 // build lwing, tip along -y axis
```

The Build function requires one Boolean input argument. Setting the argument to `true` specifies that the surface will be aligned so that its tip is along the +y axis (right side of the airplane). Setting the argument to `false` aligns the tip along the -y axis (left side of the airplane). This cumbersome manipulation is required so that the airfoil section data is corrected aligned with the aerodynamic surface.

Compound surfaces with varied dihedral or sweep are constructed by joining two previously defined surfaces. A second constructor of the Surface class creates a new surface from two others. For example, wing halves can be joined as follows:

```
Surface wing(rwing, lwing);         // creates a "wing" surface by
                                    // combining the "rwing" and "lwing"
```

Note that the individual surfaces must be built before they can be joined together.

A Section class object must be defined for each Surface object. For example, suppose a class called WingSec (derived from the abstract base Section class) is used to represent the section characteristics of a wing. The WingSec class object is attached to the Surface object using a pointer:

```
WingSec *ws = new WingSec;
wing.SectionType(ws);
```

or simply,

```
wing.SectionType(new WingSec);
```

The Control function of the Surface class is used to specify the control parameter of each spanwise section of the aerodynamic surface. For example, a rudder deflection on the vertical tail surface could be modeled using:

```
Surface vtail(4,5);                 // vtail surface has 4 sections

double rudder[] = {0.2, 0.2, 0.2, 0.2};
vtail.Control(rudder);              // rudder deflects control variable
                                    // at all 4 sections of vtail
```

The spanwise sections are ordered from root to tip. Therefore, a partial span flap can be modeled by setting a nonzero control variable at only those spanwise section locations where the flap is physically located. Also note that the sections of a joined surface are numbered in the order that the individual surfaces were first joined.

There are four membership functions associated with solving and manipulating the vortex structure of the aerodynamic surface. Each of these functions requires definition of a unit length velocity vector at the reference point and a normalized rotational rate vector. These two vectors are typically defined by:

```
Vector vrel(cos(alpha)*cos(beta), sin(beta), sin(alpha)*cos(beta));
Vector omega(2.0*phat/bspan, 2.0*qhat/cbar, 2.0*rhat/bspan);
```

where `alpha` and `beta` are the angle-of-attack and sideslip, respectively. The variables `phat`, `qhat`, and `rhat` are the nondimensional roll, pitch, and yaw angular rates. The `bspan` and `cbar` variables are the lateral and longitudinal characteristic lengths.

The Solve function determines the bound vortex strength at each section:

```
if (wing.Solve(vrel, omega)) cout << "convergence warning: wing\n";
```

The Solve function returns a Boolean variable indicating success of the iterative solution. A return value of `true` indicates that an error occurred or convergence was not achieved.

It is possible to model the wake of one surface interacting with the wake of another surface. This interaction is specified using the NearSurface function. In the example shown below, the surface called `htail` is affected by the wake of the `wing` surface. As a result, when the Solve function is called for the `htail` surface, the strength of its bound vortices will be computed using the wake of the `wing` in addition to its own wake.

```
htail.NearSurface(&wing);
```

The wake of an individual surface is manipulated using two functions: WakeVelocity and ShiftWake. The WakeVelocity function computes the velocity at each corner of each wake vortex. The ShiftWake function moves each corner of each wake vortex in the direction of the velocity at that corner. The distance of the shift is defined by the WakeLength function discussed earlier.

```
wing.WakeVelocity(vrel, omega);     // compute wake velocity
wing.ShiftWake();                   // shift wake vortices
```

The force and moment acting on the surface are computed using the Force and Moment functions. For example,

```
Vector wfor = wing.Force(vrel, omega);
Vector wmom = wing.Moment(vrel, omega);
```

The Force and Moment functions return objects of the Vector class. The components of these vectors represent the force and moment resolved in the body-fixed axis, at the reference point, and normalized by dynamic pressure. These vectors can be converted to conventional coefficient form using arithmetic operations from the Vector class:

```
wfor /= Sref;                       // divide force vector by reference area
wfor.rotate(2, pi-alpha);           // rotate so that components are now
                                    // [cD, cy, cL]

wmom /= Sref;                       // divide moment vector by ref area
wmom.scale(1.0/bspan, 1.0/cbar, 1.0/bspan);
                                    // scale by lateral and longitudinal
                                    // characteristic lengths
                                    // [cl, cm, cn]
```

A rotating flow correction has been incorporated into the Surface class using the functions RotForce and RotMoment. These functions are called using the following syntax:

```
double alpha_sep = 20.0*pi/180.0;
Vector rwfor = rwing.RotForce(vrel, omega, alpha_sep);
Vector rwmom = rwing.RotMoment(vrel, omega, alpha_sep);
```

The third argument of the function calls above is the angle-of-attack at which separation is assumed to occur. This value is used to determine which sections of the surface are experiencing separated, rotating flow.

Appendix C: CMAP Program Description

The CMAP program is used to create two-dimensional generalized cell-to-cell and velocity maps from a set of nonlinear differential equations. It consists of four C++ source code files. One header file, main.hpp, provides the class definitions and prototypes. It is organized such that the user will work primarily with two files: main.cpp and eom.cpp. The main.cpp file is used to specify the parameters needed to construct the cell-to-cell map and the eom.cpp file is used to specify nonlinear differential equations. The other files include the source code necessary to manipulate and create class objects.

CPlane Class

The CPlane class specifies the cutting planes needed to define a two-dimensional subspace within the state space. The constructor of the CPlane class requires the user to specify d-2 cutting planes (d = state space dimension) and the coordinates of the two-dimensional cell map. The cutting planes and cell map coordinates are specified by two arrays. The first array represents a d-by-d matrix. The matrix elements are stored by rows in the array. For example,

```
double e_f[] = {1.0, 0.0, 0.0, 0.0,
0.0, 0.707, 0.0, -0.707,
0.0, 0.0, 1.0, 0.0,
0.0, 0.707, 0.0, 0.707};
```

specifies a 4-by-4 matrix with the first row equal to [1 0 0 0]. An error message is displayed if this matrix is not orthogonal. The second array represents the constants required to define the d-2 cutting planes. For example,

```
double g[] = {2.0, -5.0};
```

sets the first cutting plane to a value of 2.0 and the second to a value of -5.0.

The CPlane object constructor is called as follows:

```
CPlane pc(4, e_f, g);
```

The first parameter of the constructor specifies the problem dimension. The matrix `e_f` and array `g` follow in the calling sequence. This example will create a two-dimensional subspace within a four-dimensional space. If the space is specified by a vector $x^T = [x_1, x_2, x_3, x_4]$, then the cutting planes defined by `e_f` and `g` will be:

$$x_1 = 2.0$$

$$0.707\, x_2 - 0.707\, x_4 = -5.0$$

The coordinates of the subspace are defined by the last two rows of the input matrix `e_f`. In this example, the coordinates of the two-dimensional subspace $[y_1, y_2]$ are specified as:

$$y_1 = x_3$$

$y_2 = 0.707\ x_2 + 0.707\ x_4$

Once the CPlane object has been constructed, the two-dimensional subspace is divided into cells using two grid functions, Grid1 and Grid2. The Grid1 function defines the grid for the y_1 coordinate while Grid2 defines the grid in the y_2 direction. These functions are overloaded and can be called in two ways. A uniform grid is defined using three input arguments. In this case, the first argument is the number of cells, the second argument is the maximum value, and the third argument is the minimum value. The following example call creates a grid for the y_1 coordinate that has 25 cells in the range from -1.5 to 1.5.

```
pc.Grid1(25, -1.5, 1.5);
```

Calling Grid1 or Grid2 with two arguments generates a non-uniform grid. The first argument is the number of cells while the second argument is an array that specifies the grid cuts. If the coordinate is to be divided into n cells, the number of array elements will be n+1. The following example specifies that the y_2 coordinate is divided into 4 cells using cuts at: -1.5, -1.2, 0.0, 0.15, and 0.30.

```
double y2g[] = {-1.5, -1.2, 0.0, 0.15, 0.30};
pc.Grid2(4, y2g);
```

SSpace Class

The SSpace class is a base class that represents the dynamic system under study. The SSpace class provides all of the required functions for creating and manipulating cell and velocity maps. To implement a particular set of differential equations, the user must generate a new class that is derived from the SSpace base class. The derived class requires only two functions. A constructor must be defined for the derived class. The derived class constructor passes two parameters to the SSpace constructor. The first parameter is the state-space dimension while the second parameter is the integration time step. For example, the following source code creates a new class called Lorenz with a state-space dimension of 3 and an integration time step of 0.02 seconds.

```
Lorenz::Lorenz() : SSpace(3, 0.02) { }
```

The second function needed for the derived class is called `fofx`. This function implements the differential equations of the nonlinear system and must be provided by the user. This functions requires two arguments. The arguments are pointers to an array of double precision numbers. The elements of the first array will consist of the state variable rates while the elements of the second array refer to the state variables. The differential equations for the Lorenz system are implemented as follows:

```
void Lorenz::fofx(double *dx, const double *x)
{
double n1 = 10.0, n2 = 8.0/3.0, n3 = 28.0;

dx[0] = -n1*(x[0] - x[1]);
dx[1] = n2*x[0] - x[1] - x[0]*x[2];
dx[2] = x[0]*x[1] - n3*x[2];
}
```

Simulation

Single trajectories can be generated from the system equations using three functions from the SSpace class. The SetState function is used to specify the system state variables. This function has one input argument that points to double precision array of initial conditions. The Step function integrates the system equations one step forward in time using the well-known, fourth-order, Runge-Kutta algorithm. The GetState function can be used to copy the current state vector into an output array. This function is primarily used to change control variables.

The following example source code will generate a single trajectory from the Lorenz system. Note that the SSpace class supports output streams so that the current state vector can be written to a file or standard output using the << operator. The following example will write 250 state vectors to a file called `sim1.txt`. If the integration step size of the Lorenz constructor was specified as 0.02 seconds, then the 250 state vectors will represent a 5-second total duration. The file format will be tab-delimited ASCII text. Each line of the output stream will represent one time point.

```
ofstream cfile1("sim1.txt");        // open a file

Lorenz t3;                          // define system state-space

double x0[] = {15.0, 5.0, 27.0};    // set initial condition
t3.SetState(x0);

for (int i=0; i<250; i++)           // start recording trajectory
{
cfile1 << t3;                       // write current state to file
t3.Step();                          // step forward
}

cfile1.close();                     // close file
```

Cell Maps

A cell-to-cell map is constructed by first attaching a CPlane object to the SSpace object representing the system equations. This manipulation is completed using the Cut function of the SSpace class. The following source code illustrates the Cut function. (Note that the declarations for the arrays `mm` and `bb` have been omitted).

```
Lorenz t3;                          // define state-space

CPlane pc(3, e_f, g);               // create cutting plane

pc.Grid1(20, -20.0, 20.0);          // define grid for cell map
pc.Grid2(20, -20.0, 20.0);

t3.Cut(&pc);                        // attach cutting plane to state-space
```

The CellMap function is called after the cutting planes have been attached to the state-space. The CellMap function writes the resulting map information directly to an output stream or file. An example of the CellMap function call is:

```
ofstream cfile("cmap1.txt");        // open file

t3.CellMap(cfile, 10, 35);          // create cell map and write to file

cfile.close();                      // close output file
```

The second argument required by the CellMap function is the number of trajectories that will generated along each coordinate of each cell within the two-dimensional subspace defined by the CPlane object. If the second argument is 10, the range of y_1 and y_2 within each cell is divided into 10 points. The possible combinations of ten y_1 points and ten y_2 points will yield 100 total starting points from within each cell. As a result, the generalized cell-to-cell map is constructed by starting 100 trajectories from within each cell. Furthermore, these 100 trajectories are evenly spread within the domain of each cell. The third argument of the CellMap function specifies how many integration steps each trajectory is tracked before its range cell is determined. The example above specifies 35 steps, so that the Step function will called 35 times.

The cell-to-cell map information is sent to an output stream or file. This information is written in ASCII text format. The first three lines of the output format specify the grid information for the y_1 coordinate. The first line is the number of cells in y_1. The second line is a tab-delimited list of the center positions of each cell in y_1. The third line is the width of each cell in y_1. The next three lines of output repeat the same information for the y_2 coordinate.

The remaining data in the CellMap output are the cell-to-cell map entries. Each entry consists of a row index, a column index, and a solution count in tab-delimited ASCII format. The solution count is the number of trajectories that ended inside the given cell. The solution count is converted to a probability by dividing by the total number of solutions that were started from within each cell. Note that the output of the CellMap function represents a sparse matrix and includes only non-zero map entries. The sink cell is represented by a zero row index. The row and column indices are referenced to unity.

A MATLAB function called 'plotcmap' has been included for analysis of the cell-to-cell maps created by the CMAP program. This function loads a cell map and then locates the persistent and transient cells. It then creates a plot of the probability of absorption and the expected absorption time of the transient cells in the map. The Markov model, represent by the cell-to-cell map, is represented in the MATLAB workspace using the variable called 'cmp'. It can be used to create stochastic simulations from a given probability vector. This variable is a sparse matrix structure and should only be manipulated using the special sparse matrix functions within the MATLAB environment.

Velocity Maps

A velocity map contains the initial velocity vector from the center of each cell in the two-dimensional subspace. Information in the velocity map is very helpful in locating dynamic features within the state space. A velocity map is created using the VelocityMap function in the SSpace class. An example calling sequence is:

```
ofstream vfile("vmap1.txt");        // open file
```

```
Lorenz t3;                          // define state-space

// create cutting plane and grid (omitted)

t3.Cut(&pc);                        // attach cutting plane to model

t3.VelocityMap(vfile);              // find velocity map, write to vfile

vfile.close();                      // close file
```

The velocity map information is written to the output stream specified by the input argument of the VelocityMap function. The output format is tab-delimited ASCII text. The first six lines of output represent the cell grid information in the same manner as the output of the CellMap function. The remaining data are the velocity vector entries. Each line consists of three numbers. The first number is an index, the second number is dy_1/dt, and the third number is dy_2/dt.

A MATLAB function called 'plotvmap' is included for analysis of the velocity maps created by the VelocityMap function. This function reads the velocity map file and creates a quiver plot. The quiver plot helps the user to visualize the initial flow of trajectories from within the subspace. It is particularly helpful in cases where no absorbing cells are found in the cell-to-cell map.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 01- 08 - 2004	**2. REPORT TYPE** Contractor Report	**3. DATES COVERED** *(From - To)* 9 Jan 2003 to 9 Jan 2004

4. TITLE AND SUBTITLE Modeling and Analysis of Large Amplitude Flight Maneuvers	**5a. CONTRACT NUMBER** NAS1-02048
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Anderson, Mark R.	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER 23-762-65-AG
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center, Hampton, VA 23681-2199; Paper Pilot Research, Inc., P.O. Box 650776, Sterling, VA 20165-0776	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration, Washington, DC 20546-0001	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/CR-2004-212994

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 08
Availability: NASA CASI (301) 621-0390 Distribution: Nonstandard

13. SUPPLEMENTARY NOTES
Langley Technical Monitor: Martin R. Waszak
An electronic version can be found at http://techreports.larc.nasa.gov/ltrs/ or http://ntrs.nasa.gov

14. ABSTRACT
Analytical methods for stability analysis of large amplitude aircraft motion have been slow to develop because many nonlinear system stability assessment methods are restricted to a state-space dimension of less than three. The proffered approach is to create regional cell-to-cell maps for strategically located two-dimensional subspaces within the higher-dimensional model statespace. These regional solutions capture nonlinear behavior better than linearized point solutions. They also avoid the computational difficulties that emerge when attempting to create a cell map for the entire state-space. Example stability results are presented for a general aviation aircraft and a micro-aerial vehicle configuration. The analytical results are consistent with characteristics that were discovered during previous flight-testing.

15. SUBJECT TERMS
Global stability; Nonlinear systems; Cell mapping

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	62	(301) 621-0390