The Pennsylvania State University
Graduate School
College of Engineering

A DISCUSSION OF METHODS OF REAL-TIME AIRPLANE FLIGHT SIMULATION

A paper in
Aerospace Engineering
by
Carl Banks

Abstract

This paper discusses the methods of flight simulation; specifically real-time simulation of airplanes. The paper describes methods to form a minimal, but complete, flight simulator. The simulator basically operates by solving the airplane's equations of motion of the airplane in real time. The major part of solving the equations of motion is determining the total force and moment on the airplane. There are many ways to model aerodynamic forces. This paper describes three general models: polynomial modeling, where each resultant force and moment is expressed as a polynomial function; multipoint modeling, where different parts of the aircraft are modeled separately; and tabular modeling, where the forces and moments are looked up in tables. The simulator must also calculate forces and moments produced by the engine and landing gear. The simulator solves the equations of motion in a real-time loop. In each loop, outputs the scenery based on the state, inputs the control positions and settings, calculates the state at the next time step, and finally pauses to synchronize the loop to real time.

Contents

List of Tables

List of Figures

Symbols

A_{th}	Throttle plate open area
b	Wingspan
c	Length of mean aerodynamic chord
$[C]$	Direction cosine matrix representing aircraft orientation
C_{ij}	Components of the direction cosine matrix
C_D	Discharge coefficient of the engine throttle
C_T	Propeller coefficient of thrust
C_P	Propeller coefficient of power required
C_X, C_Y, etc.	Nondimensional force and moment coefficients
$C_{X_\alpha}, C_{Y_\beta}$, etc.	Stability derivatives and/or polynomial coefficients
D	Propeller diameter
e_0, e_1, e_2, e_3	Quarternions representing aircraft orientation
F_f	Forward friction force exerted on a landing gear tire
F_n	Ground normal force exerted on a landing gear tire
F_s	Sideward friction force exerted on a landing gear tire
(F/A)	Fuel to air mixture ratio
g	Acceleration of gravity ≈ 32.2 ft/s
h	Altitude
h	Distance from the ground
I_p	Propeller polar moment of inertia
I_{xx}, I_{yy}, I_{zz}	Aircraft moments of inertia
I_{xy}, I_{xz}, I_{yz}	Aircraft products of inertia
J	Propeller advance ratio
L, M, N	Components of the resultant moment on the airplane
m	Mass of the airplane
m_f	Fuel mass
$\dot{m}_{th}$	Mass flow rate past the throttle
$\mathbf{M}$	Mach number
N	Propeller/engine angular speed, revolutions per second
p	Outside air pressure
p_m	Manifold pressure
P	Power required by propeller
P_e	Excess power delivered to the propeller
p, q, r	Components of aircraft angular velocity
$\overline{p}, \overline{q}, \overline{r}$	Nondimensional components of aircraft angular velocity
q	Dynamic pressure

Q_p	Resultant torque on the propeller
r	Distance from the Earth's center
R	Gas constant of the air ≈ 1716 (ft lb)/(slug °R)
S	Wing planform area
T	Thrust force
T	Outside air temperature
T_m	Manifold temperature
u, v, w	Components of aircraft velocity
u_a, v_a, w_a	Wind-relative components of aircraft velocity
u_g, v_g, w_g	Components of the velocity of a landing gear tire
V	Airspeed
V_k	Threshold velocity for static friction
$\mathbf{V}_d$	Total displacement of the engine
$\mathbf{V}_m$	Volume of the intake manifold
X, Y, Z	Components of the resultant force on the aircraft
x, y, z	Coordinates in a Cartesian coordinate system
x_c, y_c, z_c	Coordinates of the aircraft's CG
x_g, y_g, z_g	Coordinates of the bottom of an extended landing gear tire
x_s	Distance of the screen from user's eyes
z_p	Body z-coordinate of the propeller shaft
α	Angle of attack
β	Angle of sideslip
β	Compressibility correction factor
γ	Specific heat ratio of the air ≈ 1.4
δ_a	Differential aileron deflection
δ_b	Percent of full brake pressure applied
δ_e	Elevator deflection
δ_f	Flap deflection
δ_n	Leading-edge flap deflection
δ_r	Rudder deflection
η_p	Propeller efficiency
η_v	Volumetric efficiency of the engine
Θ	Pitch Euler angle
θ	Longitude
μ_s, μ_k	Static and kinematic coefficients of friction of tires
ρ	Outside air density
Φ	Roll Euler angle
ϕ	Geodetic latitude

ϕ'	Geocentric latitude
Ψ	Yaw Euler angle

Superscripts

B	Body Coordinate System
E	Eye Coordinate System
G	Ground-Path Coordinate System
L	Local Coordinate System
S	Screen Coordinate System
T	Transpose
W	Wind Axes

Subscripts

A	Aerodynamic Forces and Moments
B	Body Force
G	Landing Gear Forces and Moments
P	Propulsive Forces and Moments

1 Introduction

Airplane flight is expensive. For activities such as pilot training, flight research, and recreation, flight simulation can provide a suitable, and less expensive, alternative to real airplane flight.

For the flight simulator to be effective, it must convey some degree of realism to the user. The simulator conveys realism in two ways: first, by accurately modeling the behavior of the real airplane, and second, by resembling, to some degree, the cockpit and surroundings of the airplane. In today's flight simulators, digital computers handle the task of determining the aircraft's motion, while various hardware imitates the environment of the cockpit.

This paper describes methods in implementing a basic flight simulator. The paper tries not to present a detailed analysis of a specific example, nor to be a tutorial on implementing a flight simulator. Rather, its purpose is to describe the mentality of flight simulation, which is often different from the mentality of related fields such as stability and control. This paper takes a broad look at simulation, often exploring different ways for a simulator to accomplish a given task.

The methods presented in this paper are mostly applicable to propeller-driven, subsonic airplanes. Most of the equations and methods described in this paper can be used, with straightforward modifications, for any rigid airplane. However, the equations require extensive modification to model flight vehicles other than airplanes, or airplanes with elastic modes, because many of the assumptions made about the aircraft are no longer valid.

The rest of the introduction describes the virtual world of the flight simulator, and how it represents the airplane and its environment internally.

1.1 The Environment

One cannot simulate an airplane without modeling the environment through which it flies, that is, the Earth and the atmosphere.

1.1.1 Earth-Fixed Coordinate Systems

From our human perspective, we define location relative to the Earth. Therefore, in a flight simulator, the location of the airplane and every other object is specified relative (directly or indirectly) to a coordinate system

fixed to the Earth. The following paragraphs describe various Earth-fixed coordinate systems.

Geocentric Cartesian Coordinates. Geocentric Cartesian coordinates are fixed to the rotating Earth, originating from the Earth's center. The z-axis points through the geographic North Pole (and coincides with the Earth's axis of rotation). The x-axis points through the intersection of Equator and Prime Meridian (0° longitude by 0° latitude). The y-axis completes the right-handed system.

Geocentric Spherical Coordinates. Because the Earth is spherical, it makes sense to define spherical coordinates for the Earth. The spherical coordinates (r, θ, ϕ') are radius from Earth's center, longitude, and geocentric latitude, respectively. The range of θ is -180° to 180°, while the range of ϕ is -90° to 90°.

The relation between geocentric spherical and geocentric Cartesian coordinates is:

$$x = r\cos\theta\sin\phi' \qquad y = r\sin\theta\sin\phi' \qquad z = r\cos\phi' \tag{1}$$

Geodetic Coordinates. Spherical coordinates can be inconvenient for two reasons. First, the geocentric latitude, ϕ', is not exactly the same as the geographic latitude used in navigation. This is because the Earth is actually an oblate spheroid, slightly flattened at the poles. And second, radius from the Earth's center is an unwieldy coordinate.

Geodetic coordinates are often more convenient than spherical coordinates. In the geodetic coordinate system, the coordinates (h, θ, ϕ) are altitude, longitude, and latitude. The geodetic latitude and longitude are the same latitude and longitude used in navigation and on maps. The geodetic and geocentric longitudes are the same.

The transformations to and from geodetic coordinates are complex. Reference 1 (pp. 809-810) gives formulas for the transformation from geodetic to geocentric coordinates. The opposite transformation, from geocentric to geodetic coordinates, is very tricky, as it requires the solution of a quartic equation and selection of the proper root. References 2 and 3 give closed-form formulas for this transformation.

Because of the complexity in transforming to and from geodetic coordinates, a flight simulator usually does not use geodetic coordinates internally. Rather, it uses geodetic coordinates only for input and output, and uses a geocentric system internally.

Flat-Earth Coordinates. In many flight simulators, global navigation is not important. For example, the range of flight could be limited to a small area, or the simulator might not care about the airplane's location.

In such cases, it is appropriate to model the Earth as a plane half-space rather than an oblate spheroid. Then, the simulator need not worry about how the local horizontal plane changes as the airplane flies around the Earth. This simplifies the bookkeeping in the simulator considerably.

The flat-Earth coordinate system is a Cartesian system, which originates at the surface. The z-axis points vertically down, the x-axis points north, and the y-axis points east.

1.1.2 Atmospheric Modeling

Obviously, the local atmospheric conditions affect the airplane. Air density and the wind speed are the most important conditions. Other conditions, such as pressure and temperature, affect some of the instruments.

The Standard Atmosphere model[4] gives the average temperature, pressure, and density (over time) at any point in the atmosphere, as a function of altitude. For altitudes less than 36000 ft, the average temperature as a function of altitude (which is an empirical formula) is:

$$\overline{T} = 518.69^\circ\text{R} - (0.003566\,^\circ\text{R/ft})h \tag{2}$$

$$\overline{p} = (2116.8\ \text{psf}) \left(\frac{\overline{T}}{518.69^\circ\text{R}} \right)^{5.262} \tag{3}$$

$$\overline{\rho} = \frac{\overline{p}}{R\overline{T}} \tag{4}$$

Equations 2-4 only give the average conditions at a point in the atmosphere, hence the overbars. Depending on the weather, the temperature and pressure could be much higher or lower than the average. Many flight simulators have meteorological models to simulate these fluctuations.

Unlike density, there is no standard model for wind; it is (almost) entirely dependent on the weather. Again, a meteorological model can be

used. In the absence of a meteorological model, the user could simply input a velocity vector for the wind he or she wants to fly in (this could be useful to practice landing in a crosswind, for instance).

Some types of wind are easily predictable and not much dependent on the weather. For example, thermal updrafts and downdrafts occur over specific locations at specific times of day.

1.2 The Airplane

This paper treats the airplane as a symmetric, rigid body. The simulator calculates its motion by solving the rigid-body equations of motion.

1.2.1 Airplane-Centered Coordinate Systems

Airplane-centered coordinate systems originate at a fixed point in the vehicle, called the reference point. The choice of the reference point is arbitrary, but it must be fixed point. In particular, the center of gravity (CG) cannot be the reference point because, over time, the CG does change in an airplane. Instead, a point such as the leading edge of the root chord, the tip of the nose, or the aerodynamic center of the wings serves as a reference point.

Body Axes. Body axes are fixed with respect to the airplane. If the airplane moves, the body axes move with it.

The orientation of the body axes with respect to the vehicle frame is arbitrary. However, by convention, the x-axis points towards the front of the airplane, the y-axis points to the right, and the z-axis points to the bottom.

A special case of body axes is when they coincide with the vehicle's principal axes. This simplifies the equations of motion, because the airplane's products of inertia become zero. In most aircraft, the x-z-plane is a plane of symmetry, and so the y-axis is automatically a principal axis. While many simulators align the x- and z-axes with principal axes, some do not due to practical considerations. In many cases, it is simpler to make the x-axis parallel the horizontal reference axis from the blueprints, or some other line.

The x-z-plane in body axes is called the reference plane. This plane is useful for defining other vehicle coordinate systems.

Wind Axes. In wind axes, the x-axis directly points into the relative wind. The z-axis remains in the reference plane, but rotates so that it remains perpendicular to the x-axis. The y-axis completes the right-handed system.

The transformation from body to wind axes consists of two rotations. First the body axes are rotated about the y-axis through the angle of attack α; the axes are then rotated about the z-axis through the angle of sideslip β, yielding the wind axes. The angle of attack and the angle of sideslip are defined respectively by

$$\alpha \equiv \tan^{-1}(w_a/u_a) \tag{5}$$

$$\beta \equiv \sin^{-1}(v_a/V) \tag{6}$$

The main reason for the wind axis system is that it is more convenient for calculating aerodynamic forces. For instance, lift is, by definition, perpendicular to the relative wind, while drag is parallel. With wind axes, both lift and drag resolve into a force parallel to one of the axes.

1.2.2 The Local Coordinate System.

Humans tend to think of an airplane's orientation relative to the horizontal, and instruments such as the artificial horizon bear this out. In addition, many calculations (for example, gravity force) require the airplane's orientation relative to the local horizontal plane. There is not, however, much direct physical meaning to the aircraft's orientation relative to geocentric axes.

Therefore, the simulator specifies the aircraft's orientation relative to an intermediate axis system, rather than directly to geocentric axes. This intermediate system is referred to as the local axis system, sometimes called the local horizontal axis system. In local axes, the z-axis always points vertically down, while the x- and y-axes are always aligned with the local horizontal plane. Usually, the x-axis points north and the y-axis points east, but in some cases (such as near the poles) it may be convenient to have the x- and y-axes point in other directions.

Generally, it doesn't matter where the local axes originate, so long as the x-y-plane is parallel to the local horizontal plane. The local axes could be

centered at the airplane's CG. They could originate directly below the airplane, at sea level. They could originate at a fixed point on the Earth's surface, close to where the airplane is flying. If the simulator uses a flat-Earth model, then the flat-Earth coordinate system serves as the local system.

Once the local reference frame is defined, the orientation of the airplane is given by the orientation of body axes relative to the local axes. The relative orientation is specified by a direction cosine matrix, whose components are the cosines of the angles between the respective body and local axes. The airplane's direction cosine matrix can transform a vector from the body reference frame to the local reference frame by premultiplying it. (Some references define the cosine matrix to do the opposite: to transform vectors from the local frame into the body frame. The orthogonality property of the cosine matrix, $[C]^{-1} = [C]^T$, obviates the need to invert the cosine matrix, and so it does not matter which definition is used.)

Due to its orthogonality condition, the direction cosine matrix can be represented in general by three independent quantities. Euler angles and quarternions are two common ways to represent orientation. Flight simulators prefer Euler angles and quarternions over the direction cosine matrix for storing aircraft orientation, because there are fewer quantities to keep track of, and fewer constraints to satisfy.

1.2.3 State Variables

To represent the airplane at a point in time, the flight simulator uses a set of state variables. The state variables completely determine the state of the airplane. The flight simulator simulates the airplane's motion by calculating the time derivatives of the state variables, using the equations of motion. Using the time derivatives, it numerically integrates the state variables in real time.

The airplane exists in three dimensions; thus three state variables (x_c, y_c, z_c in a Cartesian system) determine the position of the CG, relative to a local or Earth-fixed coordinate system.

Three Euler angles (Φ, Θ, Ψ) or four quarternions (e_0, e_1, e_2, e_3) determine the airplane's orientation relative to local axes; these are state variables.

In dynamics, the velocity (linear and angular) of a body is part of the state as well. Thus, the three components of linear velocity in body axes (u, v, w), and the three components of angular velocity in body axes

(p, q, r), are state variables.

There are other state variables, as well. The mass of the airplane affects its dynamics, and the mass does change as the fuel burns. Thus, the mass of the fuel in the tanks in a state variable. The engine speed and manifold pressure are state variables. Additionally, other state variables can emerge as the complexity of the airplane model increases.

1.2.4 Control Variables

Control variables serve as the inputs to the simulator. The user has more or less complete control over the values; the simulator sets these variables based on input from the user. For example, if a control stick is attached to the simulator, the simulator sets the elevator deflection variable to correspond to the stick's position.

The basic airplane control variables include the aerodynamic surface deflections (elevator, ailerons, rudder, and flaps), and the engine controls (throttle, blade pitch, mixture). Other control variables include the brake pressure, trim tab deflections, the position of the landing gears (retracted, extended, or in transition), whether carburetor heat is on, etc. There are far too many to list.

Even some values which are not normally considered to be controls can be control variables. For example, payload mass is a variable directly controlled by the user, and so the flight simulator considers it a control variable.

Note that there is no requirement that a particular variable be of a certain type. A quantity could be a control variable in one flight simulator, and a state variable in another. For example, a flight simulator could treat force on the control stick as the control variable, with the control surface deflections being state variables. (This is a convenient way to model stick-free flight.) Some engines have governors on them the fix the RPM; in this case, the RPM is a control variable while the blade pitch is a state variable.

Other variables. The state and control variables are sometimes inconvenient to use directly in calculations. For example, aerodynamic forces are more directly expressed as a function of the angle of attack α than of the state variables u and v. Variables such as α are neither state nor control variables, but are calculated as closed functions of the state and control

variables each time step.

1.3 Coordinate Transformations

Because of the many coordinate systems used in the flight simulator, the simulator makes extensive use of coordinate transformations. Most coordinate systems in the simulator are Cartesian, so that the transformation is linear.

(A word on notation: this paper indicates which coordinate system a component belongs to by a superscript. For example, the velocity components of the airplane in body axes are u^B, v^B, w^B. In local axes, the components are u^L, v^L, w^L¿. The superscript is not used where there is no confusion about what coordinate system a quantity belongs to.)

To transform a velocity, or some other vector quantity, from one coordinate system to another, only a rotation is required. For example, the transformation of the wind velocity vector from local axes to body axes is simply:

$$\vec{u}^B = [C]^T \vec{u}^L \tag{7}$$

where $\vec{u} = [\ u\ v\ w\ ]^T$.

Transforming points requires a rotation and translation. Transforming points between body and local axes has an additional complication as well. The body axes originate from the reference point of the airplane; however, because of dynamic considerations, the airplane's location is specified by the coordinates of its CG. This requires an additional translation from the CG to the reference point. The transformation of a point from body to local coordinates is:

$$\vec{x}^L = [C](\vec{x}^B - \vec{x}_c^B) + \vec{x}_c^L \tag{8}$$

where $\vec{x} = [\ x\ y\ z\ ]^T$ is the point in question, and $\vec{x}_c$ is the coordinates of the CG in the indicated coordinate system. The opposite transformation is:

$$\vec{x}^B = [C]^T(\vec{x}^L - \vec{x}_c^L) + \vec{x}_c^B \tag{9}$$

2 Airplane Dynamics

"Flight dynamics courses are exiting to the student and to the teacher, except for the initial portion where the equations of motion are derived."

–S. Pradeep[5]

Many references derive the equations of motion for a rigid airplane [6,7,8]. This paper will not duplicate the derivation.

The flight simulator ultimately uses the equations of motion to calculate the time derivatives of the state variables. As such, this paper presents the equations in state variable form, where the equations are solved for the time derivatives.

2.1 Dynamic Equations

The dynamic equations derive from Newton's Second Law, $\vec{F} = m\vec{a}$ and the analogous equation for angular motion. Resolving these equations along body axes, and including terms to account for centrifugal forces due to the rotating body reference frame, yields the dynamic equations of motion. For a symmetric airplane ($I_{xy} = I_{yz} = 0$), the dynamic equations are given by Equations 10-15.

$$\dot{u} = rv - qw + X/m \tag{10}$$

$$\dot{v} = pw - ru + Y/m \tag{11}$$

$$\dot{w} = qu - pv + Z/m \tag{12}$$

$$\dot{p} = \frac{I_{xz}L' + I_{xx}N'}{I_{xx}I_{zz} - I_{xz}^2} \tag{13}$$

$$\dot{q} = \frac{M - (I_{xx} - I_{zz})rp + I_{xz}(p^2 - r^2)}{I_{yy}} \tag{14}$$

$$\dot{r} = \frac{I_{zz}L' + I_{xz}N'}{I_{xx}I_{zz} - I_{xz}^2} \tag{15}$$

The L' and N' from Equations 13 and 15 are given by:

$$L' = L - (I_{zz} - I_{yy})qr - I_{xz}pq$$

$$N' = N - (I_{yy} - I_{xx})pq - I_{xz}qr$$

Equations 10-15 introduce forces and moments (X, Y, Z, L, M, N). Because of these terms, the simulator must calculate the resultant force and moment on the airplane before it can calculate the time derivatives.

2.2 Kinematic Equations

The kinematic equations relate the velocities (angular and linear) of the airplane to the time derivatives of the position and orientation.

2.2.1 Euler Angles

The orientation of an airplane, relative to local axes, can be specified by the three sequential rotations about the body axes. Starting with the body axes aligned with the local axes, the first rotation is about the z-axis through an angle Ψ, followed by a rotation about the y-axis through an angle Θ, followed by a rotation about the x-axis through an angle Φ. These angles of rotation are the Euler angles, and can represent any possible orientation of the airplane.

Equation 16 lists the airplane's direction cosine matrix constructed from the Euler angles.

$$[C] = \begin{bmatrix} \cos\Theta\cos\Psi & \sin\Phi\sin\Theta\cos\Psi - \cos\Phi\sin\Psi & \cos\Phi\sin\Theta\cos\Psi + \sin\Phi\sin\Psi \\ \cos\Theta\sin\Psi & \sin\Phi\sin\Theta\sin\Psi + \cos\Phi\cos\Psi & \cos\Phi\sin\Theta\sin\Psi - \sin\Phi\cos\Psi \\ -\sin\Theta & \sin\Phi\cos\Theta & \cos\Phi\cos\Theta \end{bmatrix} \tag{16}$$

Equations 17-19 represent the kinematic equations for the Euler angles, relating the Euler angle time derivatives to the angular velocity.

$$\dot{\Phi} = p + q\sin\Phi\tan\Theta + r\cos\Phi\tan\Theta \tag{17}$$

$$\dot{\Theta} = q\cos\Phi - r\sin\Phi \tag{18}$$

$$\dot{\Psi} = q\sin\Phi\sec\Theta + r\cos\Phi\sec\Theta \tag{19}$$

2.2.2 Quarternions

Euler angles have two disadvantages. First, the Euler angle equations contain many trigonometric functions. Trigonometric functions are very slow compared to basic arithmetic operations such as addition and multiplication. For computational efficiency, it is almost always better to choose a method using only simple arithmetic operations.

The more important disadvantage is the numerical singularity appearing in Equations 17 and 19 when$\Theta = \pm 90^\circ$, that is, when the airplane's nose points straight up or down. While not a problem in normal, level

flight, the singularity can present numerical problems when the airplane performs maneuvers such as loops.

Replacing the three Euler angles with four quarternions alleviates both difficulties. Because only three parameters define any possible rotation, the quarternions need a constraint so that there are only three independent variables. This constraint is[9]:

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1 \tag{20}$$

One nice feature of quarternions is that numerical integration of their time derivatives tends not to destroy the constraint. Thus, a flight simulator only needs to normalize the quarternions occasionally. With high enough numerical precision, it may not ever have to normalize them.

The direction cosine matrix for the airplane, constructed from the four quarternions, is given by Equation 21[9].

$$[C] = \begin{bmatrix} e_0^2 + e_1^2 - e_2^2 - e_3^2 & 2(e_1e_2 + e_0e_3) & 2(e_1e_3 - e_0e_2) \\ 2(e_1e_2 - e_0e_3) & e_0^2 - e_1^2 + e_2^2 - e_3^2 & 2(e_2e_3 + e_0e_1) \\ 2(e_1e_3 + e_0e_2) & 2(e_2e_3 - e_0e_1) & e_0^2 - e_1^2 - e_2^2 + e_3^2 \end{bmatrix} \tag{21}$$

The kinematic equations for the quarternions are given by Equations 22-25[9].

$$\dot{e}_0 = -\tfrac{1}{2}(e_1p + e_2q + e_3r) \tag{22}$$
$$\dot{e}_1 = \tfrac{1}{2}(e_0p - e_3q + e_2r) \tag{23}$$
$$\dot{e}_2 = \tfrac{1}{2}(e_3p + e_0q - e_1r) \tag{24}$$
$$\dot{e}_3 = \tfrac{1}{2}(-e_2p + e_1q + e_0r \tag{25}$$

Quarternions are more computationally efficient than Euler angles, and do not exhibit singularities. Furthermore, the kinematic equations using quarternions are hardly more complex then the corresponding equations using Euler angles; some might say that the quarternion equations are simpler. In fact, the only drawback to using quarternions is that they have no useful physical meaning.

In practice, this is not a serious drawback. Only a human can benefit from the direct physical meaningfulness of the Euler angles. Humans can use Euler angles to visualize an aircraft's orientation, but computers determine the orientation by mathematical computation. Thus, it is only important to use Euler angles when transferring information to and from

a human, i.e., for input and output. It is not important at all to use Euler angles internally.

When a simulator using quarternions inputs the initial orientation of an airplane as Euler angles, it must convert them to quarternions. Equations 26-29 give the formulas for this[9].

$$
\begin{aligned}
e_0 &= \cos\tfrac{\Psi}{2}\cos\tfrac{\Theta}{2}\cos\tfrac{\Phi}{2} + \sin\tfrac{\Psi}{2}\sin\tfrac{\Theta}{2}\sin\tfrac{\Phi}{2} && (26)\\
e_1 &= \cos\tfrac{\Psi}{2}\cos\tfrac{\Theta}{2}\sin\tfrac{\Phi}{2} - \sin\tfrac{\Psi}{2}\sin\tfrac{\Theta}{2}\cos\tfrac{\Phi}{2} && (27)\\
e_2 &= \sin\tfrac{\Psi}{2}\cos\tfrac{\Theta}{2}\sin\tfrac{\Phi}{2} + \cos\tfrac{\Psi}{2}\sin\tfrac{\Theta}{2}\cos\tfrac{\Phi}{2} && (28)\\
e_3 &= \sin\tfrac{\Psi}{2}\cos\tfrac{\Theta}{2}\cos\tfrac{\Phi}{2} - \cos\tfrac{\Psi}{2}\sin\tfrac{\Theta}{2}\sin\tfrac{\Phi}{2} && (29)
\end{aligned}
$$

The reverse process, obtaining the Euler angles from quarternions, is done indirectly, after calculating the cosine matrix. By inspection of Equation 16, the following relations yield the Euler angles:

$$
\begin{aligned}
\Phi &= \tan^{-1}(C_{32}/C_{33}) && (30)\\
\Theta &= \sin^{-1}(-C_{31}) && (31)\\
\Psi &= \tan^{-1}(C_{21}/C_{11}) && (32)
\end{aligned}
$$

One reason for the simulator to calculate the Euler angles is to display the instruments properly. For example, the heading indicator requires the value of Ψ.

2.2.3 Position

The time derivative of the position is simply the velocity, rotated into the coordinate system of the position variables. For simulators that store the aircraft's CG position in local coordinates, the kinematic equations for position are:

$$
\begin{aligned}
\dot{x}_c &= C_{11}u + C_{12}v + C_{13}w && (33)\\
\dot{y}_c &= C_{21}u + C_{22}v + C_{23}w && (34)\\
\dot{z}_c &= C_{31}u + C_{32}v + C_{33}w && (35)
\end{aligned}
$$

Simulators that store position in a Cartesian coordinate system other than local coordinates substitute components of the appropriate rotation matrix into Equations 33-35.

Simulators that store aircraft position in geocentric spherical coordinates can calculate the spherical coordinate time derivatives using the using the local coordinate time derivatives as intermediate variables[10]:

$$\dot{r} = -\dot{z}_c^L \tag{36}$$
$$\dot{\theta} = \dot{y}_c^L/(r\cos\phi') \tag{37}$$
$$\dot{\phi}' = \dot{x}_c^L/r \tag{38}$$

2.3 Forces and Moments

The equations of motion have reduced the solution of aircraft motion from determining the time derivatives of the state variables to determining the resultant forces and moments on the aircraft. The external forces and moments on the aircraft come from four general sources: aerodynamic force, propulsive force, force on the landing gear, and body forces (mostly gravity). The simulator calculates these separately and then adds them together. For example, the force in the x-direction is given by:

$$X = X_A + X_P + X_G + X_B \tag{39}$$

There are other minor sources of forces and moments, such as internal damping, but this paper does not consider them.

Of the different types of force, calculation of body forces is the most straightforward. The most important body force in flight simulation is gravity. Gravity acts parallel to the local z-axis. Thus, the components of gravity force along body axes is the component in the local z-direction are:

$$X_B = mgC_{31} = -mg\sin\Theta \tag{40}$$
$$Y_B = mgC_{32} = mg\cos\Theta\sin\Phi \tag{41}$$
$$Z_B = mgC_{33} = mg\cos\Theta\cos\Phi \tag{42}$$

There are no body moments.

Another body force on the airplane is the Coriolis force due to the Earth's rotation. This force is important for high-speed, high-altitude flight, but is not that important for low-altitude, low-speed flight.

Calculation of the other types of force is difficult. The following two sections describe in detail the calculation of these forces.

3 Airplane Aerodynamic Modeling

Precise modeling the aerodynamic forces is very difficult indeed. Aerodynamic forces and moments are complex functions of the airplane's state and control variables, and the local environment. The important state and control variables are the airspeed along body axes (u_a, v_a, w_a), the angular velocity components (p, q, r), and the control deflections ($\delta_a, \delta_e, \delta_r, \delta_f$, etc.). Aerodynamic interactions between different components of the airplane (wing and stabilizer, for example) can complicate the relationships. External interactions such as ground effect complicate these relationships even more.

There are many models for the aerodynamic forces and moments. Most aerodynamic models are accurate only in a restricted domain. The very simplest models, for example, are accurate only near a certain trim condition. More sophisticated models are accurate for normal flight conditions, but not for post-stall situations or other unusual situations. Very detailed simulations can be accurate for almost the entire flight envelope.

The rest of this section describes three general models: polynomial models, multipoint models, and tabular models.

3.1 Polynomial Models

Polynomial models represent the resultant aerodynamic forces and moments at the CG as polynomial functions of state variables and control variables, or values derived from these. The coefficients of the polynomials are constant parameters, defined to fit the aerodynamics as closely as possible for the flight conditions of significance. The polynomials can have terms in more than one variable.

One of the amazing things about airplane aerodynamics is that, for all of its inherent complexity, a mostly linear polynomial model can be fairly accurate for a wide range of flight conditions. Certain phenomena cause the relative linearity of the airplane to break down; accurate prediction of aerodynamic forces and moments in these situations require a higher-order model. The two most important phenomena that require the higher-order models are stall (high angles of attack) and drag divergence (high Mach numbers). But in flight regimes not involving stall or drag divergence, a mostly linear model can be quite accurate.

In order for a polynomial model to work well, it must do three things. First, the model must calculate aerodynamic moments about the reference point (or some other convenient fixed point). Calculating the moment about the non-fixed CG requires the polynomial model to incorporate CG location as a variable, an unnecessary complication. After the moments about the fixed point have been calculated, the moments are transfered to the CG for use in the equations of motion.

Second, the model must calculate forces and moments in wind axes. After the forces are determined in wind axes, they are rotated to body axes using Equation 43.

$$\begin{bmatrix} X_A^B \\ Y_A^B \\ Z_A^B \end{bmatrix} = \begin{bmatrix} \cos\beta\sin\alpha & -\sin\beta\sin\alpha & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \cos\beta\cos\alpha & -\cos\beta\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} X_A^W \\ Y_A^W \\ Z_A^W \end{bmatrix} \tag{43}$$

Moments are transformed similarly.

Third, the model must use nondimensionalized forces and moments.

3.1.1 Nondimensionalization

Nondimensionalization reduces the dependence of all aerodynamic forces and moments on density and airspeed to a simple factor. Without nondimensionalization, almost every term in the polynomial equations would include V and ρ as variables.

The nondimensional forces and moments are defined by Equations 44-49.

$$C_X \equiv X \,/\, (\tfrac{1}{2}\rho V^2 S) \tag{44}$$
$$C_Y \equiv Y \,/\, (\tfrac{1}{2}\rho V^2 S) \tag{45}$$
$$C_Z \equiv Z \,/\, (\tfrac{1}{2}\rho V^2 S) \tag{46}$$
$$C_L \equiv L \,/\, (\tfrac{1}{2}\rho V^2 S b) \tag{47}$$
$$C_M \equiv M \,/\, (\tfrac{1}{2}\rho V^2 S c) \tag{48}$$
$$C_N \equiv N \,/\, (\tfrac{1}{2}\rho V^2 S b) \tag{49}$$

The angle of attack and angle of sideslip, defined by Equations 5 and 6, are the nondimensional replacements for the dimensional wind-relative velocity components (u_a, v_a, w_a).

Equations 50-52 define nondimensional angular speeds.

$$\overline{p} \equiv pb/V \tag{50}$$
$$\overline{q} \equiv qc/V \tag{51}$$
$$\overline{r} \equiv rb/V \tag{52}$$

Unfortunately, there is not a standard definition of nondimensional angular speeds. Some references define the nondimensional speeds with a 2 in the denominator.

One other nondimensional parameter is defined analogously to the nondimensional angular speeds. $\dot{\alpha}$, the time derivative of α, is used to account for the effect of downwash lag. Its nondimensional form is:

$$\overline{\dot{\alpha}} \equiv \dot{\alpha}c/V \tag{53}$$

The control deflections, which are angles, are already nondimensional.

3.1.2 Basic Polynomial Model

Aerodynamic models rely on wind tunnel and flight test data for their accuracy. However, many references list formulas for estimating stability derivatives based on geometry and mass properties alone [6,7,11]. These stability derivatives can serve as the coefficients in a basic polynomial model.

Many stability derivatives are constant for any trim point or transient point of significance. The polynomial model does not require higher-order terms to represent such relationships. However, some stability derivatives (for example, C_{X_α}) vary with flight condition. The model requires higher-order terms to account for the variation.

Obviously, not all forces depend on all variables. Linear stability analysis decouples longitudinal motion from lateral-directional motion. Many of these coupled terms are not needed. However, in flight simulation, there are some coupled terms. Some of these terms, such as $C_{X_\beta}\beta$ and $C_{X_{\delta_r}}\delta_r$, are fairly significant terms that stability analysis neglects to decouple the motions. Other terms, such as $C_{N_{\delta_a}}\delta_a$, depend on the lift coefficient, which is considered constant in stability analysis. The equivalent term in flight simulation is a mixed coupled term: $C_{N_{\delta_a C_Z}}\delta_a C_Z$.

There are several nonlinear terms. Arguably, the most important relationship is C_Z versus α. While this relationship is linear for low angles

of attack, the onset of stall destroys this linearity. Modeling stall requires higher order terms; probably at least fifth-order for a close match. The relationship between C_M and α is also nonlinear beyond stall; if the flight simulator models C_Z beyond stall, it should model C_M beyond stall also. (Other relationships may lose their linearity beyond stall also; however, it is not considered worthwhile to model these in a simple polynomial model.)

Another nonlinear relationship is C_X versus α. Basic airplane performance calculations treat this relationship as quadratic, which is sufficient for a polynomial model, also.

Equations 54-59 present a basic polynomial model, with some nonlinear terms.

$$\begin{aligned} C_Z &= C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_{\alpha^2}}\alpha^2 + C_{Z_{\alpha^3}}\alpha^3 + C_{Z_{\alpha^4}}\alpha^4 + C_{Z_{\alpha^5}}\alpha^5 \\ &\quad + C_{Z_q}\overline{q} + C_{Z_{\delta_e}}\delta_e + C_{Z_{\delta_f}}\delta_f \end{aligned} \tag{54}$$

$$\begin{aligned} C_X &= C_{X_0} + C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_\beta}\beta + C_{X_{\beta^2}}\beta^2 \\ &\quad + C_{X_{\delta_f}}\delta_f + C_{X_{\delta_e}}\delta_e + C_{X_{\delta_r}}\delta_r \end{aligned} \tag{55}$$

$$\begin{aligned} C_M &= C_{M_0} + C_{M_\alpha}\alpha + C_{M_{\alpha^2}}\alpha^2 + C_{M_{\alpha^3}}\alpha^3 + C_{M_{\alpha^4}}\alpha^4 + C_{M_{\alpha^5}}\alpha^5 \\ &\quad + C_{M_q}\overline{q} + C_{M_{\dot{\alpha}}}\overline{\dot{\alpha}} + C_{M_{\delta_e}}\delta_e + C_{M_{\delta_f}}\delta_f \end{aligned} \tag{56}$$

$$C_Y = C_{Y_0} + C_{Y_\beta}\beta + C_{Y_r}\overline{r} + C_{Y_{\delta_r}}\delta_r \tag{57}$$

$$\begin{aligned} C_L &= C_{L_0} + C_{L_\beta}\beta + C_{L_r}\overline{r} + C_{L_{rC_Z}}\overline{r}C_Z + C_{L_p}\overline{p} \\ &\quad + C_{L_{\delta_r}}\delta_r + C_{L_{\delta_a}}\delta_a \end{aligned} \tag{58}$$

$$\begin{aligned} C_N &= C_{N_0} + C_{N_\beta}\beta + C_{N_r}\overline{r} + C_{N_{pC_Z}}\overline{p}C_Z \\ &\quad + C_{N_{\delta_r}}\delta_r + C_{N_{\delta_a C_Z}}\delta_a C_Z \end{aligned} \tag{59}$$

This model's domain of accuracy extends through angles of attack several degrees beyond stall. Beyond this, the accuracy diminishes rapidly. This model is accurate for moderate values of sideslip, angular velocity, and control deflection. This model doesn't consider the effects of Reynolds number on drag or maximum lift, nor does it consider compressibility effects. In short, the model's accuracy is quite limited. Even so, a low-speed, subsonic airplane will spend most of its flying time within this model's domain of accuracy.

The major benefit of the basic model is that its coefficients are simple to calculate. Most coefficients stem from basic stability and performance equations, while a simple polynomial curve fit determines the higher order terms in Equations 54 and 56.

3.1.3 Compressibility Correction

The polynomial model described above is valid only for Mach numbers below about 0.3. Compressibility effects become important above Mach 0.3.

For subsonic flight, before the onset of drag divergence at about Mach 0.7, a simple correction is to divide each force coefficient by a compressibility correction factor β. For example, the equation to correct C_Z is $C_Z = C_{Z,incomp.}/\beta$.

The simplest correction is the Prandtl-Glauert factor (Reference 7, p. 212):

$$\beta = \sqrt{1 - M^2} \tag{60}$$

This relation is derived for isentropic flow. Most of the time, this is a reasonable assumption, as the flow around the streamlined aircraft is mostly isentropic. However, in conditions where isotropy is dubious (such as advanced stall, where there is substantial separation, and drag divergence, when shock waves appear), the correction is not valid.

3.1.4 More Complex Polynomial Models

The basic model described above has two restrictions. First, its domain of accuracy is small; it cannot correctly model flight conditions such as advanced stall and spins. Second, the basic model cannot simulate the many subtle effects (for example, how C_{L_β} changes with angle of attack), because there are no terms to account for such effects. These subtle effects are small compared to lift versus angle of attack, and are not easy to calculate, and so they are not used in the basic model. However, these effects are real, and not accounting for them limits the maximum accuracy of the model.

The obvious solution to these problems is to use more terms in the polynomials. More terms can make the model more accurate, as well as increase its domain of accuracy. Unfortunately, as more terms are added, it becomes increasingly difficult to calculate the coefficients. Therefore, complex polynomial models use numerical methods of parameter fitting to determine the coefficients. The parameter fitting methods rely on test data (either from flight tests or wind tunnel tests).

However, before parameter fitting, one must choose which parameters are needed. One possibility is to use brute force; simply choosing any desired polynomial terms, and letting the optimizer decide whether there is

any relationship of significance. This method is problematic for two reasons. First, the memory and computer time required for this is very large. Second, as the degree of the polynomials increase, it becomes increasingly likely that two terms may be nearly linearly dependent with the given test data[12]. This creates an ill-conditioned problem.

Reference 12 presents a method to determine a set of terms that are not linearly dependent. It derives a method of selecting terms that are orthogonal to each other based on the given data set. (Suppose there are N data points, with each data point listing the force F and the state variables $a, b, c, \ldots$ at some point in time. Two terms $P_1(a, b, c, \ldots)$ and $P_2(a, b, c, \ldots)$ are orthogonal if

$$\sum_{n=1}^{N} (F_n - P_1(a_n, b_n, c_n, \ldots))(F_n - P_2(a_n, b_n, c_n, \ldots)) = 0,$$

where the subscript n indicates the nth data point.) The method presented in Reference 12 does not yield polynomial terms; however, a well-conditioned polynomial can be derived from the orthogonal terms.

3.1.5 Ground Effect

One major factor affecting the aerodynamics is ground effect. There are two approaches to capture this effect.

One approach is to incorporate distance from the ground as a variable in the polynomials. For example, let ξ be some nondimensional parameter representing the distance from the ground. This parameter might be defined as $\xi = b/h$, where b is the wingspan and h is the distance from the wing root chord to the ground. (The reason that h is in the denominator is so ξ can go to zero far from the ground.)

Then, to capture ground effects, there could be terms such as C_{Z_ξ} and $C_{L_{\xi\Phi}}\xi\Phi$. The latter term introduces something that is not seen outside of ground effect. Typically, the aerodynamic force and moment depends only on wind-relative speed; however, in ground effect the force depends on orientation as well, hence the Euler angle Φ. (To be more exact, it depends on orientation relative to the ground. Thus, the Euler angle Φ could not be used to simulate landing on surface that is not level.)

The other approach to ground effect is to use corrections. Reference 7 (pages 359-360) lists empirical corrections to lift and drag in ground effect.

These corrections are not as accurate as incorporating ground distance into the polynomials; however, they are useful for basic, hand-calculated models.

3.1.6 Domain of Accuracy Checking

Every polynomial model has a domain in which it is accurate. (The domain of accuracy could be the set of possible flight conditions, but probably is not.) Because polynomials generally approach infinity, the model can yield ridiculous values for the force and moment coefficients when outside its domain of accuracy. It is therefore important to detect when the aircraft exceeds the model's domain of accuracy, especially with variables involved in high-order terms.

Depending on its purpose, the simulator may choose to abort or reset the simulation, informing the user that the airplane encountered a flight condition that the simulator could not handle. Or, the simulator could choose to use reasonable values for the force and moment coefficients, perhaps by setting them to an upper or lower bound, or perhaps by setting them to zero.

Flight conditions outside of the domain of accuracy often happen unexpectedly. For example, suppose an airplane is sitting on a runway, heading north, while a breeze crosses the airplane in a direction one degree north of east. Relative to the wind, the airplane's angle of sideslip is $89°$. Because of terms such as $C_{L_\beta}\beta$ and $C_{Y_\beta}\beta$, large forces and moments can result from this small breeze, especially when β appears in higher powers.

3.2 Multipoint Models

The polynomial models described above, as well as many other models, are single point models. Such models treat the aerodynamic forces and moments as point forces and moments acting through the reference point of the airplane. However, there are situations where it is useful to model distributed aerodynamic forces, because of the radical difference in conditions that appear in different parts of the aircraft in certain flight conditions. This can have drastic effects on the resultant aerodynamic force and moment. A spin is a good example of a flight condition where this occurs; among other things, the outer wingtip has a significantly higher airspeed than the inner wingtip due to the high yaw rate.

Because of their complexity, capturing these effects using a single point model is tricky. It can be done using enough terms. However, it is often to better to model phenomena as they really are. That is, if local variations in force cause complex behavior, it makes sense to consider these local variations, rather than to force single-point polynomials to model their effects.

A simple multipoint approach is to use a separate polynomial model for each lifting surface. Thus, the individual aerodynamic force and moment on the wing, horizontal tail, vertical tail, and fuselage is summed to yield the total aerodynamic force and moment. A slightly more complex model considers the left and right halves of each lifting surface separately. A still more complex model separates each lifting surface, and the fuselage, into lengthwise panels, calculating the force and moment produced by each panel. For example, the wing may consist of eight panels, the horizontal tail four, the vertical tail two, and the fuselage six.

The logical extreme of the multipoint model is to consider infinitely thin panels. This can be done using strip theory [13,14], where the resulting force on a lifting surface is the integral of force per unit length along the span. The force can be evaluated analytically by expressing the components of the integral as Taylor-expanded polynomials. For example, the lift of the right half of the wing could be represented by Equation 61:

$$Z = \int_0^{b/2} c_z(\alpha(y))\, q(y)\, c\, dy \tag{61}$$

where

$$\begin{aligned} q(y) &= q(0) + \left.\frac{dq}{dy}\right|_{y=0} y + \left.\frac{d^2q}{dy^2}\right|_{y=0} y^2 \\ \alpha(y) &= \alpha + \left.\frac{d\alpha}{dy}\right|_{y=0} y + \frac{1}{2}\left.\frac{d^2\alpha}{dy^2}\right|_{y=0} y^2 \\ c_z(\alpha) &= C_0 + C_1\alpha + C_2\alpha^2 + \ldots \end{aligned}$$

Note that $q(y)$ and $\alpha(y)$, and therefore the Taylor expanded coefficients, can be determined from the airplane's velocity and angular velocity. The parameters c_1, c_2, etc., are unknown parameters. Integrating Equation 61, and rearranging, the equation for for Z-force becomes:

$$Z = C_0k_0 + C_1k_1 + C_2k_2 + \ldots \tag{62}$$

where $k_0, k_1, k_2, \ldots$ are functions of $\alpha(0)$, $q(0)$, and their derivatives at $y = 0$. Then, using flight test or wind tunnel data, a numerical optimizer can determine values for C_0, C_1, etc. Reference 14 reports that the aerodynamics forces in a spin are modeled very well by this model, even when simulating one type of spin with a model with parameters identified using data from a different type of spin.

One problem with multipoint models is that they usually have a large number of parameters. The large number of parameters can lead to an ill-conditioned problem, very similar to the ill-conditioned problem of single-point polynomial models, making numerical parameter identification difficult. A possible remedy for this is to collect data on the distributed force, rather than just the resultant force.

3.3 Tabular Data Models

Often, it is not worth it to try to find a closed function modeling the aerodynamics. In such cases, the the simulator can obtain the forces and moments from a tabular data obtained from flight or wind tunnel tests.

Table 1 lists a simple example of a table. The table lists a drag polar typical of a wing with a laminar flow airfoil. To determine the drag, the simulator determines which two points in the table the angle of attack is between, and then calculates the drag by linear interpolation. To speed up the interpolation, the slope of each segment is often precomputed and stored in the table as well [15].

Aerodynamic force is generally a function of more the one variable; therefore, flight simulators use multidimensional tables. Theoretically, the table could have a dimension for each variable the force coefficient is a function of. For example, C_M may be expressed as

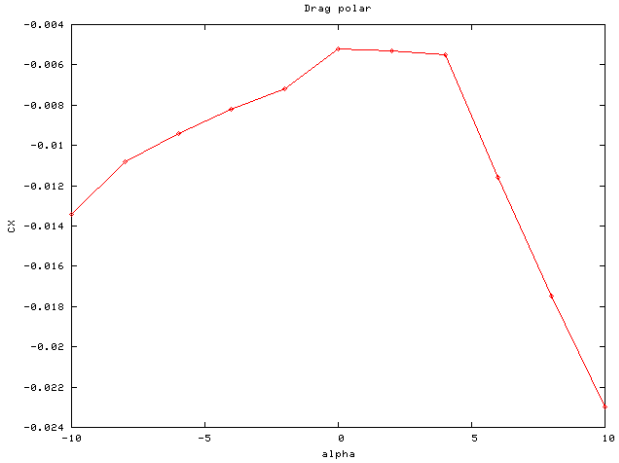
$$C_M = C_M[\alpha, \beta, \delta_e, \delta_f, \overline{q}, \mathbf{M}], \tag{63}$$

where the square bracket notation indicates a tabulated function. However, the memory and testing time required for a table with many dimensions is extremely large. (If each variable of the table in Equation 63 takes on 20 different values, which is a typical number, then the table will have 64 million data points.)

Fortunately, the effects of some variables are not influenced, or are influenced only weakly, by other variables. For example, the effect of δ_f on

Table 1: Sample table of C_X versus α, with plot.

α (deg)	C_X
-10	-0.0134
-8	-0.0108
-6	-0.0094
-4	-0.0082
-2	-0.0072
0	-0.0052
2	-0.0053
4	-0.0055
6	-0.0116
8	-0.0175
10	-0.0230



pitching moment is not influenced by $\overline{q}$, and vice versa. Thus, δ_f and $\overline{q}$ never need to appear in the same table. This allows the large table to be broken into several smaller tables, each limited to three or four dimensions. The tabular model sums the contributions from each table.

A tabular aerodynamic model has been used to simulate an F/A-18[16]. Equation 64 presents an example of an extremely simplified version of the model for the yawing moment:

$$\begin{aligned} C_N &= C_{N_0}[\beta,\alpha,\mathbf{M}] + C_{N_{\delta_f}}[\beta,\alpha,\mathbf{M}]\left(\frac{\delta_f}{20}\right) + C_{N_{\delta_n}}[\beta,\alpha,\mathbf{M}]\left(\frac{\delta_n}{25}\right) \\ &\quad + \Delta C_{N_{\delta_a}}[\delta_a,\alpha,\mathbf{M}] + \Delta C_{N_{\delta_r}}[\delta_r,\alpha,\mathbf{M}] + C_{N_p}[\alpha,\mathbf{M}]\overline{p} + C_{N_r}[\alpha,\mathbf{M}] \end{aligned} \quad (64)$$

The main problem with tabular models is the lack of smoothness. For example, the C_X curve from Table 1 is nondifferentiable at the tabulated points, and its derivative is discontinuous. Flight simulators can remedy this somewhat by using tables with smaller increments. Another technique is to use higher-order interpolation.

4 Airplane Subsystem Modeling

While aerodynamics is the major contributor to the forces and moments on an aircraft, some of the systems, the propulsion system and the landing gear in particular, can contribute as well.

4.1 Piston Engine Modeling

This section describes a model for a non-supercharged piston engine with a variable pitch propeller.

Piston engine manufacturers provide charts of engine power output for different flight conditions. To use the information in these charts in the flight simulator, data fitting techniques can yield closed form equations for the engine performance[17]. Unfortunately, the charts give engine performance as a function of instrument readings (tachometer, manifold absolute pressure), while a flight simulator must calculate the engine performance as a function of the engine controls (throttle, propeller pitch, and mixture).

Thus, the engine speed and the manifold pressure appear as intermediate variables in the calculations of engine power. In fact, the simulator

introduces manifold pressure and engine speed as state variables of the engine (which are numerically integrated in real time, like the airplane state variables).

Given the manifold pressure, engine speed, and outside air density, one can calculate the shaft power delivered by the engine.

4.1.1 Manifold Pressure Calculation

Equation 65 presents the quasi-static differential equation for manifold pressure[18].

$$\dot{p}_m = \frac{2\dot{m}_{th} R T_m - \eta_v \mathbf{V}_d N p_m}{2\mathbf{V}_m} \tag{65}$$

For high speed engines, this equation could be numerically unstable with the simulator's time step size; in such cases, it is appropriate to ignore the transients by setting $\dot{p}_m$ to zero and solving Equation 65 directly for p_m. (Reference 18 reports that the time constant of the Equation 65 is 2 to 4 times the length of an intake stroke. At 2400 RPM, twice the intake stroke is 0.025 seconds. A typical time step size for a flight simulator is 0.01 seconds.)

In Equation 65, the volumetric efficiency η_v is only weakly dependent on the manifold pressure; this paper assumes it to be independent of p_m. The manifold temperature T_m is assumed known; an adequate estimate is to use outside air temperature. $\mathbf{V}_d$ and $\mathbf{V}_m$ are constants for the engine, and N is the engine speed. Only $\dot{m}_{th}$ is left to be determined.

If $p_m > 0.528p$, then the flow is not choked at the throttle. The mass flow rate past the throttle is given by Equation 66[18]:

$$\dot{m}_{th} = \frac{C_D A_{th} \rho}{\sqrt{RT}} \left(\frac{p_{th}}{p}\right)^{1/\gamma} \sqrt{\frac{2\gamma}{\gamma - 1}\left(1 - \left(\frac{p_{th}}{p}\right)^{\frac{\gamma-1}{\gamma}}\right)} \tag{66}$$

The coefficient of discharge C_D in Equation 66 is an empirical, engine-dependent parameter, which accounts for the pressure loss due to all of the obstacles in the intake (air filter, throttle, venturi, etc.). It is a function of the engine speed and throttle plate open area. One can obtain a first-order estimate of C_D by setting it so that Equation 65 yields the observed manifold pressure drop at steady state with a wide-open throttle at a high engine speed. p_{th} is the pressure at the throttle. It is not clear from Reference 18

how to determine this; however, it seems that, although technically invalid, the manifold pressure is used for p_{th}.

If $p_m < 0.528p$, then the flow is choked at the throttle. In this case, mass flow rate is given by Equation 67[18]:

$$\dot{m}_{th} = \frac{C_D A_{th} p}{\sqrt{RT}} \sqrt{\gamma} \left(\frac{2\gamma}{\gamma + 1} \right)^{\frac{\gamma+1}{2(\gamma-1)}} \tag{67}$$

4.1.2 Propeller Modeling

Many thrust and power calculations use nondimensional parameters; these are defined by Equations 68-70.

$$C_T \equiv \frac{T}{\rho N^2 D^4} \tag{68}$$

$$C_P \equiv \frac{P}{\rho N^3 D^5} \tag{69}$$

$$J \equiv \frac{V}{ND} \tag{70}$$

Propeller efficiency charts, which are useful in performance analysis of steady, level flight, are not sufficient for flight simulation. The reason is that, in performance calculations, thrust is considered equal to drag. This is not true in general for a flight simulator.

Therefore, to model the propeller, one needs to determine the thrust produced. The thrust produced by the propeller is a function of the forward airspeed, the air density, the blade geometry (including the variable pitch setting), and the propeller's angular speed. The thrust can be calculated with a theory such as blade-element theory or vortex theory. One can also use empirical data (such as the chart on Page 306 of Reference 7) to determine the thrust.

Once the thrust is determined, the power consumed by the propeller can be found using the propeller efficiency charts in tabular form, by solving the following relation for C_P:

$$\eta_p = C_T J / C_P \tag{71}$$

Or, the power consumed can be calculated directly, much as the thrust was, as a function of airspeed, density, blade geometry, engine speed, and thrust.

Note that the power consumed by the propeller does not necessarily equal the power delivered to it by the engine. Any excess power delivered through the shaft increases the propeller's angular speed. Similarly, if the propeller does not receive enough power from the engine, its angular speed decreases.

Newton's Second Law for angular momentum, applied to the propeller, is $Q_p = I_p \dot{N}$. The excess power is related to the propeller torque by the relation $P_e = Q_p N$. Combining the two equations yields the differential equation for engine speed:

$$\dot{N} = \frac{P_e}{I_p N} \tag{72}$$

Using this equation, the simulator calculates N each time step by numerical integration.

Assuming the shaft axis is parallel to the body x-axis, and lies in the x-z-plane, the propulsive force and moment components are:

$$X_P = T \tag{73}$$

$$L_P = \pm Q_p \tag{74}$$

$$M_P = T(z_c^B - z_p) \tag{75}$$

The sign in front of Q_p in Equation 74 depends on which way the propeller spins.

4.1.3 Mixture

The engine performance charts do not factor in mixture setting, assuming that the pilot will keep the mixture near optimal. Any effects the mixture has on the power or behavior of the engine can be modeled as corrections.

An important effect of mixture setting is the rate of fuel consumption. The rate of fuel burn is given by:

$$\dot{m}_f = -(F/A)\dot{m}_{th} \tag{76}$$

where (F/A) is settable with the mixture control. Of course, a flight simulator uses a negative sign because it is interested in the actual fuel weight, which decreases as the fuel burns.

4.2 Landing Gear Modeling

Two phenomena result in force on the landing gear. The first is the normal force exerted by the ground; the second is the friction on the tires. Either way, the tire must contact the ground for the force to be generated. Thus, the simulator tests whether the extreme point of the fully extended landing gear is below the ground. If it is, the tire is touching the ground, with the strut and tire compressed.

The landing gear model in this section is mostly a simplified version of the model found in Reference 19. Each gear is considered separately in calculating landing gear force.

4.2.1 Ground Normal Force

For simplicity, assume that the strut is parallel to the airplane's body z-axis, and that the landing surface is level. If the tire has made contact, its z-coordinate in local axes would be greater than the z-coordinate of the ground (remember, the z-axis points down).

The tire's coordinates in body axes are (x_g^B, y_g^B, z_g^B)). Using these together with Equation 8, the tire's local z-coordinate is given by:

$$z_g^L = z_c^L + C_{31}(x_g^B - x_c^B) + C_{32}(y_g^B - y_c^B) + C_{33}(z_g^B - z_c^B) \qquad (77)$$

Let z_r^L be the ground's local z-coordinate (the r means runway). If $z_g^L > z_r^L$, then the tire touches the ground. Of course, the tire cannot actually be located below the ground; the ground compresses the tire and landing gear strut.

The compressive force in the strut depends on the total gear displacement Δz_g^B. This is found by replacing z_g^L in Equation 77 with z_r^L, and solving for z_g^B, to yield z_g^B', the actual position of the compressed gear where it touches the ground. The difference between the uncompressed and compressed tire locations is Δz_g^B. Equation 78 gives the formula for this.

$$\Delta z_g^B = z_g^B - z_g^{B\prime} = z_g^B - \frac{z_r^L - z_c^L + C_{31}(x_g^B - x_c^B) - C_{32}(y_g^B - y_c^B)}{C_{33}} \qquad (78)$$

Because the strut is damped, the force also depends on the rate of landing gear displacement. Taking the time derivative of Equation 78, the only

non-constant term on the right side is z_c/C_{33}. Therefore, the displacement rate is given by:

$$\Delta \dot{z}_g^B = \frac{\dot{z}_c^L}{C_{33}} \tag{79}$$

A simple model for landing gear displacement force is to assume a linear damped elastic strut, and ignore the tire compression. Then, the compressive force in the strut is:

$$F_{strut} = K(\Delta z_g^B) + C(\Delta \dot{z}_g^B), \tag{80}$$

where K and C are the spring and damping constants for the strut. The strut force acts along the axis of strut; however, the force exerted by the ground acts vertically upward. The value of the total ground normal force is the value where the component along the strut axis is the strut force. That is,

$$F_n = F_{strut}/C_{33} \tag{81}$$

More advanced models consider the tire deflection as well as the the strut deflection in calculating ground normal force.

4.2.2 Friction Force

A tire is designed to roll easily in one direction, while sliding with difficulty in the perpendicular direction. It makes sense to resolve the friction force into a forward rolling friction and a sideward sliding friction. This paper assumes that the plane of the tires is parallel to the body x-z-plane (i.e., no steering).

Because friction always acts in a direction opposite the velocity, the calculation of friction force requires the velocity of the tires resolved into those directions. The velocity of the tire in body coordinates is the velocity of the CG plus contributions due to angular rates:

$$u_g^B = u + q(z_g^{B\prime} - z_c^B) - r(y_g^B - y_c^B) \tag{82}$$

$$v_g^B = v + r(x_g^B - x_c^B) - p(z_g^{B\prime} - z_c^B) \tag{83}$$

$$w_g^B = w + p(y_g^B - y_c^B) - q(x_g^B - x_c^B) \tag{84}$$

These can be transformed to local coordinates with the airplane direction cosine matrix. Then, the tire velocity in local coordinates are transformed

into the ground-path coordinate system by Equations 85 and 86:

$$u_g^G = u_g^L \cos\Psi + u_g^L \sin\Psi \quad (85)$$

$$v_g^G = -v_g^L \sin\Psi + v_g^L \cos\Psi \quad (86)$$

The ground-path coordinate system is like local coordinates in that the z-axis points vertically down. However, in ground-path axes, the x-axis points in the direction of travel along the ground (while remaining in the horizontal plane). The x- and y-components of the tire's velocity in ground-path coordinates are, in fact, the forward and sideward components.

For the rest of the section, the superscript on u_g^G and v_g^G is dropped; all tire velocities are in ground-path coordinates.

Sideward Force. First, v_g, the sideward groundspeed component, is compared to a threshold velocity, V_k. If —v_g—¿ V_k, then a kinetic coefficient of friction applies. If $|v_g| \leq V_k$, the static coefficient of friction applies[10]. The friction force in the sideward direction is given by Equation 87.

$$F_s = \begin{cases} -\dfrac{v_g}{V_k}\mu_s F_n, & |v_g| > V_k \\ -\text{sign}(v_g)\mu_k F_n, & |v_g| \leq V_k \end{cases} \quad (87)$$

Both the static and kinetic coefficients are functions of both the tire and landing surface. Note that the force of static friction is proportional to the negative velocity.

There is one factor which has been neglected above. A rolling tire has a relieving effect on the sideward friction, meaning that if the tire is rolling, the sideward friction force of a tire will be somewhat lower than calculated. The method of calculating this effect is not very transparent. Reference 19 presents a more sophisticated method for calculating side force of a rolling tire.

Forward Force. The forward force calculation resembles to the sideward force calculation. One complication seen with forward force calculations is the possibility of braking.

Braking changes both the threshold velocity and the kinetic friction coefficient. We assume the effect varies linearly with brake pressure. Equations 88 and 89 give formulas for the threshold velocity and kinetic friction

coefficient under different braking conditions, where the brake pressure varies from no brake pressure ($\delta_b = 0$) to full pressure ($\delta_b = 1$).

$$V_k = V_{k,roll} + \delta_b(V_{k,slide} - V_{k,roll}) \quad (88)$$

$$\mu_k = \mu_{k,roll} + \delta_b(\mu_{k,slide} - \mu_{k,roll}) \quad (89)$$

Using these definitions of V_k and μ_k, the forward friction force is as listed in Equation 90

$$F_f = \begin{cases} -\dfrac{u_g}{V_k}\mu_s F_n, & |u_g| > V_k \\ -\text{sign}(u_g)\mu_k F_n, & |u_g| \leq V_k \end{cases} \quad (90)$$

The forward friction, sideward friction, and ground normal forces are the x-, y-, and z-components of force in the ground-path coordinate system. (F_n could have been written Z_G^G.) The forces are transformed first into local coordinates:

$$X_G^L = F_f \cos\Psi - F_s \sin\Psi \quad (91)$$

$$Y_G^L = F_f \sin\Psi + F_s \cos\Psi \quad (92)$$

$$Z_G^L = F_n \quad (93)$$

Then, the components of force are transformed from local to body coordinates. The moments are obtained from multiplying the force components in body coordinates by the appropriate moment arms:

$$L_G = Y_G(z_g^{B\prime} - z_c^B) - Z_G(y_g^B - y_c^B) \quad (94)$$

$$M_G = Z_G(x_g^B - x_c^B) - X_G(z_g^{B\prime} - z_c^B) \quad (95)$$

$$N_G = X_G(y_g^B - y_c^B) - Y_G(x_g^B - x_c^B) \quad (96)$$

5 The Simulator Program

After modeling the aircraft and its environment, the next task is implementing those models in a computer. This section describes the implementation of the flight simulator program.

The flight simulator program internally has five different tasks: input, calculations, output, time synchronization, and support tasks. Input refers to getting the control values from the user, such as positions of the yoke,

Figure 1: Flowchart of the simulator's main loop.

throttle, and pedals, or whether a button has been pressed or a switch has been toggled. Calculations refer to the process of numerically integrating the state variables. Output refers to displaying the aircraft and its surroundings in the current state, as in drawing the scenery and instrument panel. Time synchronization code makes sure virtual time is the same as real time. Support tasks refer to tasks that are not directly involved in flight simulation, such as loading the scenery from a file. This paper does not cover support tasks.

5.1 Main Loop

The main loop of the program repeatedly executes tasks in the proper sequence to simulate the airplane. Each iteration of the loop is called a frame, and a frame occupies a fixed length of time. Figure 1 presents the main loop.

Before the main loop begins, the simulator sets the state variables of the airplane based on the initial conditions. The flight simulator may have to convert the initial conditions to its internal format; for example, converting Euler angles to quarternions using Equations 26-29, or converting geodetic to geocentric coordinates using the equations from Reference 1.

The frame begins with the flight simulator performing output based on the current aircraft state. (The loop is timed so that the output step finishes at the exact time represented by the state variables.) Following output, the simulator inputs the control positions and settings, and sets the control variables accordingly.

Following the input, the simulator calculates the state of the airplane for the beginning of the next frame. This is done using a finite-difference numerical integration of the state variables. The time step in the numerical integration must be small enough to capture the shortest time scales in airplane motion. The short period mode and roll mode have very short half-damping times for many aircraft. For example, Reference 7 reports a roll mode half-damping time of 0.13 seconds for a Piper Cherokee 180. Thus, the time step should be somewhat smaller than 0.1 seconds.

However, output, which is usually by far the most time consuming task of the flight simulator, often requires more time than is available in a single

time step. To accommodate the output, one frame can have multiple time steps. For example, if the time step size is 0.01 seconds, and the simulator takes about 0.03 seconds to perform all output for a single frame, then the frame length would be 0.04 seconds, with 4 time steps per frame. (Output dominates processing time so much that the time required by other tasks is often considered negligible.)

The actual time to output a frame can vary. The plain landscape of Nebraska has much fewer polygons than the mountainous landscape of Colorado, and thus requires much less processing time. As an airplane flies from Colorado to Nebraska, the simulator might be able to decrease the number of time steps per frame, and thus get a better frame rate. Conversely, when flying from Nebraska to Colorado, the simulator might have to increase the time steps per frame. Therefore, the simulator periodically adjusts the number of steps per frame to optimize the frame rate.

After all tasks in the frame have been completed, the simulator pauses. The purpose of the pause is to maintain a fixed frame length. For example, if the processing time for a frame is 0.03 seconds, and the frame length is 0.04 seconds, then the simulator pauses for 0.01 seconds. The pause varies in duration to accommodate any fluctuations in the processing time.

Following the pause, the simulator loops back to the output step.

The rest of the section describes the output and calculation tasks in more detail.

5.2 Output

A real-time flight simulator is a virtual reality program. It presents the user with an environment that mimics, to some degree, the environment of a real airplane. A very expensive flight simulator might rest on a full-motion platform, and have a cockpit identical to the simulated aircraft's cockpit, identical controls and instruments, a realistic view of the scenery, and simulated engine noise. On the other hand, an inexpensive flight simulator running on a PC might display the scenery and instruments on the monitor, and employ the keyboard for airplane controls.

The output of the flight simulator is what gives the user the perception that he or she is flying a real aircraft. There are many ways to contribute to the overall experience of flying, including very subtle effects such as cockpit ventilation. This paper covers only the most basic form of output, namely,

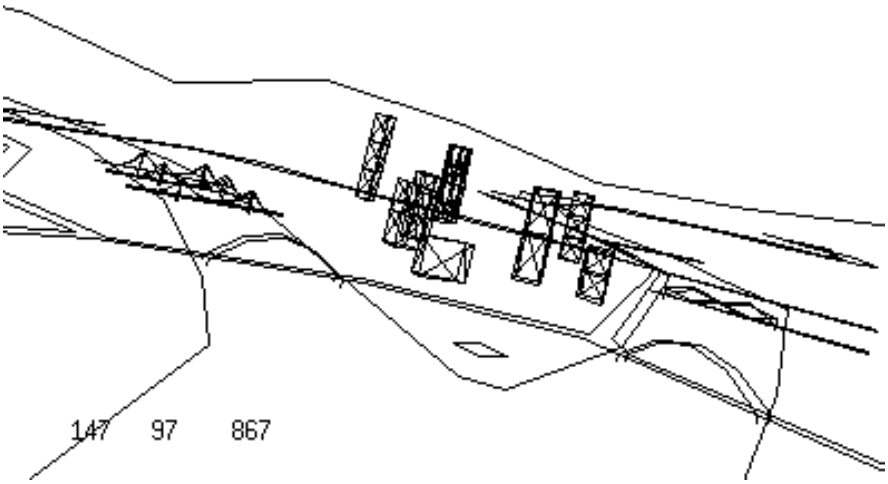


Figure 2: Wireframe scene of Downtown Pittsburgh, Pennsylvania.

drawing the scenery. Reference 15 details some more advanced methods of output.

Wireframe Scenery. The simplest type of scene is a wireframe scene. A wireframe scene is a scene drawn using only straight line segments. Figure 2 depicts a wireframe scene of Pittsburgh, Pennsylvania, as it might appear from an aircraft.

The description of a wireframe scene consists simply of a list of line segments in 3-D space. For example, in a Cartesian coordinate system, a cube could be represented by the following twelve line segments:

$(1,1,1) - (0,1,1)$	$(1,1,1) - (1,0,1)$
$(1,1,1) - (1,1,0)$	$(0,1,1) - (0,0,1)$
$(0,1,1) - (0,1,0)$	$(1,0,1) - (0,0,1)$
$(1,0,1) - (1,0,0)$	$(1,1,0) - (0,1,0)$
$(1,1,0) - (1,0,0)$	$(1,0,0) - (0,0,0)$
$(0,1,0) - (0,0,0)$	$(0,0,1) - (0,0,0)$

Because scenery is (usually) fixed relative to the Earth, the endpoints of the line segments are stored in an Earth-fixed coordinate system.

To draw the scene, the flight simulator transforms each endpoint in the scene into eye coordinates. The eye coordinate system is a Cartesian coordinate system based on the pilot's view. The eye axes originate approximately at the pilot's eyes. The x-axis points along the pilot's line of sight, the y-axis points to the right as seen by the pilot, and the z-axis points down as seen by the pilot.

After the endpoints have been transformed to eye coordinates, the flight simulator calculates their projections onto the 2-D plane of the screen. Figure 3 illustrates this. The location at which to draw the object on the screen is where the line of sight intersects the plane of the screen. Given the distance from the eyes to the screen (x_s), and eye coordinates of a point (x^E, y^E, z^E), the screen coordinates can be obtained using similar triangles. (The screen coordinate system is the 2-D coordinate system of the physical plane of the screen, originating at the screen's center, with the y-axis pointing to the right, and the z-axis pointing down.) Equations 97 and 98 list the formulas for the projection.

$$y^S = x_s y^E / x^E \quad (97)$$

$$z^S = x_s z^E / x^E \quad (98)$$

The projection yields the screen coordinates of the endpoint. The screen coordinates are actual lengths. Thus, if an endpoint's z_s-coordinate is -1 inch, and its y_s-coordinate is zero, the flight simulator draws the endpoint one inch above the screen's center.

So, to draw the wireframe scene, the simulator transforms the endpoints of every line segment into screen coordinates, and then draws a straight, 2-D line segment on the screen connecting the endpoints of each segment.

Clipping. There is one problem, however. A point in eye coordinates is not always within the field of view. Projecting such a point would lead to incorrectly rendered images. Therefore, the simulator must draw only the part of the line segment within the field of view. Before it transforms an endpoint into screen axes, the simulator must detect when an endpoint is not visible, and use the endpoint of the visible part of the segment instead, if there is one. This process is called clipping. To draw the scene correctly, the flight simulator must clip every line segment to the visible region[20].

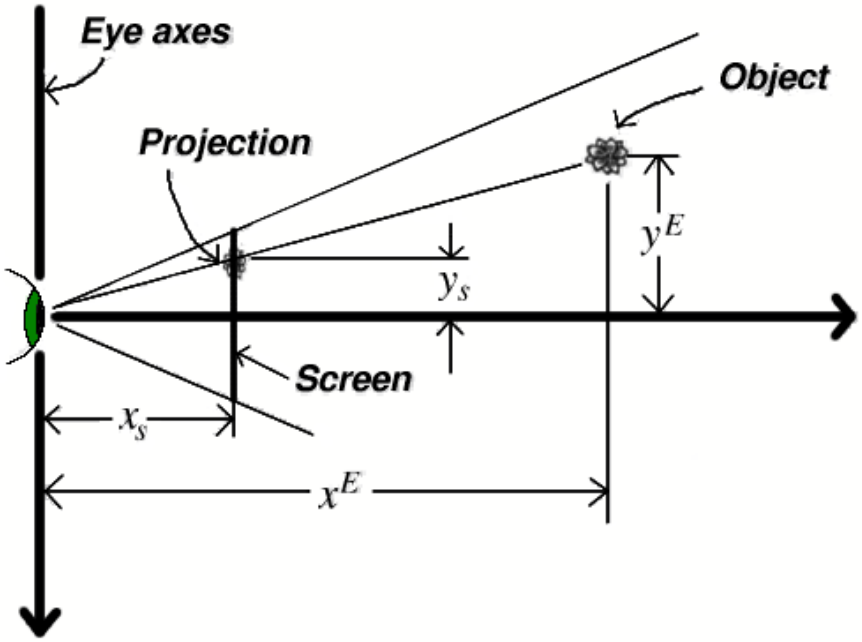


Figure 3: Schematic of a projection. The screen coordinate, y^S, is determined by similar triangles.

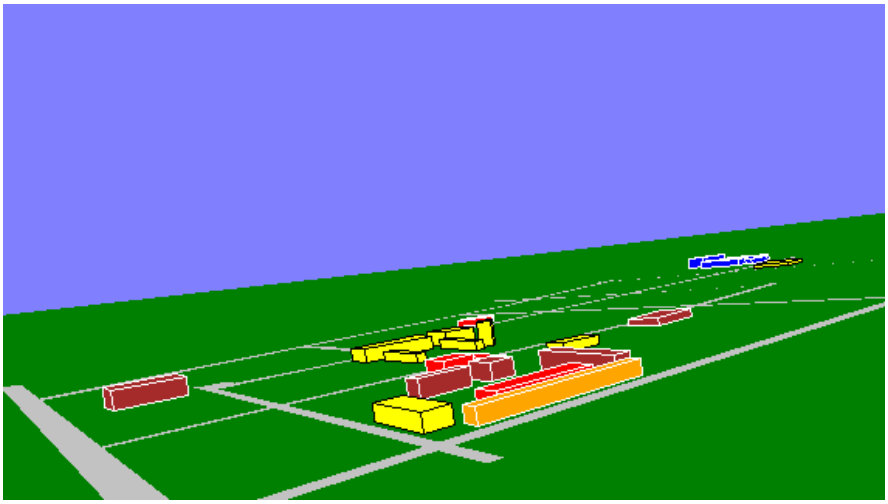


Figure 4: Polygon scene of Penn State's University Park Campus, State College, Pennsylvania.

Polygon Scenery. Wireframe scenery is not realistic. It is bland and colorless, and surfaces do not hide objects behind them. Using polygons instead of line segments increases the realism of the scenery. Figure 4 depicts a polygon scene of Penn State.

Drawing polygons is not much different than drawing line segments. (In fact, one can think of a line segment as a degenerate polygon with two vertices.) A polygon is described by a sequence of 3-D points, which represent the vertices. For example, in Cartesian coordinates, a square might be described this way:

$$(0,0,0) - (1,0,0) - (1,1,0) - (0,1,0)$$

To draw the polygon, the simulator transforms these points into screen coordinates, and then draws the polygon described by the screen coordinates, filling its interior.

The most important difficulty arising from polygon scenery is how to draw the scene so that distant polygons do not obscure close ones. One algorithm that accomplishes this is Painter's Algorithm, which draws the most distant polygons first, and the closer polygons afterward. Thus, the closer polygons obscure the more distant ones.

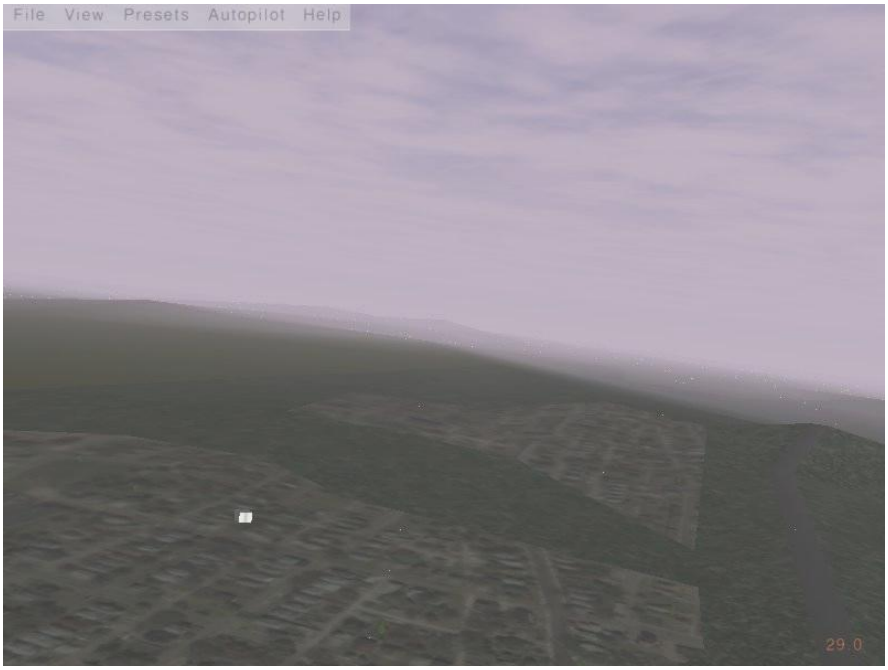


Figure 5: Textured polygon scene of an area near San Francisco.

Texturing. Monocolor polygons, although much better than wireframes, are still not very realistic. Textures can give the scene structure at scales much too small to efficiently model as polygons. Figure 5 depicts a textured polygon scene of the outskirts of San Francisco, captured from Flight Gear Flight Simulator. In the scene, the the variation in color is not represented by many different polygons, but rather is part of the ground's surface texture. (Figure 5 exhibits some other advanced rendering techniques, such as fogging.)

5.3 Time Step Calculation

Using the current state and control variables of the airplane, the flight simulator calculates, by finite differences, the state of the airplane a small time increment in the future.

The first task for the time step calculation is to calculate variables dependent on the environment, including the air density and wind velocity; perhaps using Equations 2-4 to determine density.

The second task is to calculate the variables that depend on the state and control variables. The simulator calculates the cosine matrix using Equation 16 or Equation 21, depending on how it stores orientation. If the simulator stores orientation with quarternions, it then calculates the Euler angles using Equations 30-32. It transforms the wind velocity to body axes using Equation 7, and then subtracts this from the aircraft velocity to yield the airspeed components (u_a, v_a, w_a). If it is important, the simulator may use the formulas from Reference 3 to calculate the geodetic coordinates (h, θ, ϕ).

The flight simulator calculates α and β using Equations 5 and 6. It calculates nondimensional parameters like $\overline{q}$ using Equations 50-53. To calculate $\dot{\alpha}$, it uses the chain rule with $\dot{u}$ and $\dot{v}$:

$$\dot{\alpha} = \frac{d}{dt} \tan^{-1}\left(\frac{w_a}{u_a}\right) = \frac{u_a \dot{w} - w_a \dot{u}}{u_a^2 + w_a^2} \tag{99}$$

Equation 99 assumes $\dot{u_a} \approx \dot{u}$ and $\dot{w_a} \approx \dot{w}$.

Once the necessary variables are calculated, the flight simulator uses the aerodynamic, landing gear, engine, and body force models from Sections 2, 3, and 4 to calculate the total resultant force and moment on the airplane.

Then it employs the the rigid body equations of motion (Equations 10-15, 17-19 or 22-25, and 33-35 or 36-38) to calculate the time derivatives of all the rigid-body state variables. Additionally, it uses Equations 65, 72, and 76, to calculate the time derivative of other state variables within the airplane.

Having determined the time derivatives of all state variables, the simulator is ready to use a finite-difference scheme to calculate the future state of the airplane. Generally, schemes that require only one calculation of the time derivatives per time step are preferred.

Reference 21 recommends using the predictive Adams-Brashford method. For a given state variable s, the method is given by Equation 100, in which Δt is the step size.

$$s(t + \Delta t) = s(t) + \frac{\Delta t}{2}(3\dot{s}(t) - \dot{s}(t - \Delta t)) \tag{100}$$

The method is second-order accurate. Another second order accurate method from Reference 21 is the trapezoidal method, given by Equation 101.

$$s(t + \Delta t) = s(t) + \frac{\Delta t}{2}(\dot{s}(t) + \dot{s}(t - \Delta t)) \tag{101}$$

Reference 21 reports that both methods have acceptably low phase errors. Note that both methods require the time derivatives from the previous time step.

References

1. Larson, J. W., and Wertz, J. R., Eds., *Space Mission Analysis and Design,* 2nd Ed., Microcosm Inc., Torrance, California, 1992.

2. Sofair, I., "Improved Method for Calculating Exact Geodetic Latitude and Altitude," *Journal of Guidance, Control, and Dynamics,* Vol. 20, No. 4, 1997, pp. 824-826.

3. Sofair, I., "Improved Method for Calculating Exact Geodetic Latitude and Altitude Revisited," *Journal of Guidance, Control, and Dynamics,* Vol. 23, No. 2, 2000, p. 369.

4. Anon., *U.S. Standard Atmosphere: 1976,* October 1976.

5. Pradeep, S., "Formulation of Equations of Motion of Aircraft," *AIAA Flight Mechanics Conference, Collection of Technical Papers,* Portland, Oregon, August 1999, p. 797.

6. Etkin, B., and Duff, L., *Dynamics of Flight: Stability and Control,* 3rd Ed., Wiley, New York, 1996.

7. McCormick, B. W., *Aerodynamics, Aeronautics, and Flight Mechanics,* 2nd Ed., Wiley, New York, 1995.

8. Ashley, H., *Engineering Analysis of Flight Vehicles,* Dover, New York, 1974.

9. Anon., "Recommended Practice: Atmospheric and Space Flight Vehicle Coordinate Systems," ANSI/AIAA R-004-1992, February 1992.

10. Jackson, E. B., "Manual for a Workstation-based Generic Flight Simulation Program (LaRCSim) Version 1.4," NASA TM-110164, April 1995.

11. Anon., *USAF Stability and Control Datcom,* January 1975.

12. Morelli, E., "Global Nonlinear Aerodynamic Modeling Using Multivariate Orthogonal Functions," *Journal of Aircraft,* Vol. 32, No. 2, 1995, pp. 270-275.

13. Jaramillio, P. T., and Nagati, M. G., "Multipoint Approach for Aerodynamic Modeling in Complex Flowfields," *Journal of Aircraft,* Vol. 32, No. 6, pp. 1335-1341.

14. Cho, Y., and Nagati, M. G., "Coupled Force and Moment Parameter Estimation for Aircraft," *Journal of Aircraft,* Vol. 35, No. 2, pp. 247-253.

15. Rolfe, J. M., and Staples, K. J., *Flight Simulation* , Cambridge University Press, Cambridge, 1986.

16. Buttrill, C. S., Arbuckle, P. D., and Hoffler, K. D., "Simulation Model of a Twin-Tail, High Performance Airplane," NASA TM-107601, April 1992.

17. Smith, H. C., and Dreier, M., "A Computer Technique for the Determination of Brake Horsepower Output of Normally-Aspirated Reciprocating Aircraft Engines," SAE Paper No. 770465, March 1977.

18. Heywood, J. B., *Internal Combustion Engine Fundamentals* , McGraw-Hill, New York, 1988, pp. 279-325.

19. Baarspul, M., "A Review of Flight Simulation Techniques," *Progress in the Aerospace Sciences,* Vol. 27, No. 1, 1990, pp. 1-120.

20. Woo, M., Neider, J., and Davis, T., *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.1,* Addison Wesley, Reading, Mass., 1998.

21. McFarland, R. E., "A Standard Kinematic Model for Flight Simulation at NASA Ames," NASA CR-2497, January 1975.