AIAA Modeling and Simulation Technologies Conference and Exhibit
11-14 August 2003, Austin, Texas

AIAA 2003-5818

AIRCRAFT SIMULATION TECHNIQUES USED IN LOW-COST, COMMERCIAL SOFTWARE

Michael K. Zyskowski*
Microsoft Corporation
Redmond, WA

ABSTRACT

The simulation of aircraft is probably one of the most complicated yet exciting fields in the engineering world today. As personal computers have become more powerful, the ability to simulate the complex behavior of aircraft motion in real time is now possible on these lower-cost platforms. Microsoft Flight Simulator is one of the most mature, low-cost and feature-packed simulation programs on the market today; Microsoft Combat Flight Simulator includes many of the same simulation capabilities available in Microsoft Flight Simulator. Through implementation of the classical aircraft Equations of Motion, Quaternion Transformation methods, Direct Cosine Euler Transformations and non-dimensional aerodynamic coefficient implementation, these programs provide a robust and accurate aircraft simulation software package that have been built, added to and improved upon through years of development. Together with the detailed propulsion simulation techniques, ground reaction fidelity and abundance of systems modeling, these mainstays of the marketplace offer many unique aircraft simulation features. This paper will provide a brief product history and describe the techniques used to simulate aircraft in both Microsoft Flight Simulator and Microsoft Combat Flight Simulator.

Figure 1. Microsoft Flight Simulator 2004: A Century of Flight – Wright Flyer

INTRODUCTION

One of the most popular entertainment titles published for the PC, Microsoft Flight Simulator (MSFS) is celebrating its 20th anniversary during the same year we are celebrating the 100th anniversary of the Wright Brothers' first powered flight. During these 20 years, continuous improvements have been made to the way the program simulates aircraft in flight – from simplified linear equations of flight to the inclusion of non-linear flight dynamic components, detailed ground reaction and damage modeling, extensive systems integration and robust propulsion simulation techniques. It is this continuous improvement in the core simulation engine that makes MSFS not only a solid simulation platform, but also very flexible and extensible in nature.

BRIEF HISTORY OF MICROSOFT FLIGHT SIMULATOR

Where It All Began

In 1975, Bruce Artwick was working on his Master's Thesis in Electrical Engineering at the University of Illinois entitled, "*A Versatile Computer-Generated Dynamic Flight Display.*"[1] This paper demonstrated that the computers of that day could not only handle the mathematical computations required for aircraft simulation routines, but that they could also generate graphical images as a way to provide visual feedback to the user.

Shortly thereafter, Artwick teamed up with his flight instructor, Stu Moment, to form a company called SubLOGIC to develop and market a commercial aircraft simulator based on his thesis work. The resulting *FS1 Flight Simulator* was an instant success, and by the end of 1980 had become one of the best-selling software titles for the Apple computer.[2]

Along Comes Microsoft

By 1981, Microsoft approached Artwick with a proposition – allow Microsoft to exclusively license his program for the new yet-to-be-introduced computer, the IBM-PC. Artwick accepted, and Microsoft Flight Simulator 1.01 was released for the IBM-PC in 1982.

*AIAA Member, Aerospace and Software Design Engineer

Not only was it one of the most popular software titles for the new IBM home-computer, but it was often used as the standard for determining whether or not a computer of that era was "PC-compatible." Many versions of MSFS were released throughout the decade, each improving all aspects of the product.[2]

The Saga Continues…

In 1988, Artwick formed his own company called BAO Ltd, for the Bruce Artwick Organization. Between 1989 an 1993, BAO's development efforts focused more on expansion packs and development tools rather than the core Flight Simulator product. Some of these included the *Aircraft and Scenery Designer*, intended to help create graphical improvements and add-ons, and *FlightShop*, which allowed users to create their own aircraft flight models.

These development tools created a marketplace for additional content that would work with MSFS, or "add-ons." Coupled with the advancement of the Internet, hundreds of aircraft, buildings, airports and other add-on components were available for download by any user, often for free.

After the release of Flight Simulator 95, Artwick sold all rights of the product to Microsoft. Many of the people employed at BAO Ltd at that time came with the program to Redmond, WA, and some are still working on it today.

Modern History

Through each successive version of the product, MSFS has been improved in many respects: aircraft, graphics, terrain, airports, sounds, special effects, and many others. By the release of FS98, MSFS seemed to be a popular franchise with a bright future – but there was something missing…guns.

Microsoft Combat Flight Simulator (MSCFS) was released the following year. Dubbed CFS1.0, this version of the popular MSFS series was entirely focused on the air combat aspects of WWII European Theater operations. Though the focus had changed from a civilian to military theme, the core Simulation Engine of the product has remained the same to this day.

Both MSFS and MSCFS have continued to be improved, with a new version of one or the other being released each year. Through each subsequent release, new features are added, old ones are improved, and the standard for quality is raised to meet the expectations of our customers.

Figure 2. Microsoft Combat Flight Simulator 3: Battle for Europe – P-47D

Figure 2 provides a glimpse of the latest MSCFS release, Microsoft Combat Flight Simulator 3.0: Battle for Europe. The latest release of MSFS, Microsoft Flight Simulator 2004: A Century of Flight, celebrates the centennial of flight and all the advancements made in the aviation industry over the first 100 years of powered flight (see Figure 1). Both products should be available upon release of this paper.

CORE SIMULATION ENGINE

The core simulation engine (a.k.a SimEngine) refers to the theory, principles, functions, and processes which directly relate to the way in which we simulate aircraft behavior. This includes the motion of aircraft, but also includes the way the aircraft reacts with the ground, the methodologies used to simulate thrust and propulsion, and the simulation of a host of systems typically found in a real aircraft. It's important to realize that the SimEngine is an ever-evolving core component of both MSFS and MSCFS, and many developers from both product teams have contributed to it through many years of development. I've taken the liberty to share this knowledge, but make no claim to have designed or otherwise invented all the information provided here.

First to be discussed is a general description of the SimEngine, and some of the key challenges and components thereof. Next will be a detailed breakdown of the Flight Dynamics, then Propulsion, Ground Reaction, Systems, Damage and finally the Realism factors used in the SimEngine.

Center of the Universe

Though it may be a skewed perception, the concept given in Figure 3 is intended to illustrate the fact that

the SimEngine interacts with virtually every part of the product. This makes sense when you consider the product we are developing is a flight simulator – of course the simulation of the aircraft is the centerpiece of the product! However, I imagine that the graphics developer or game designer might have a different view of the "universe."

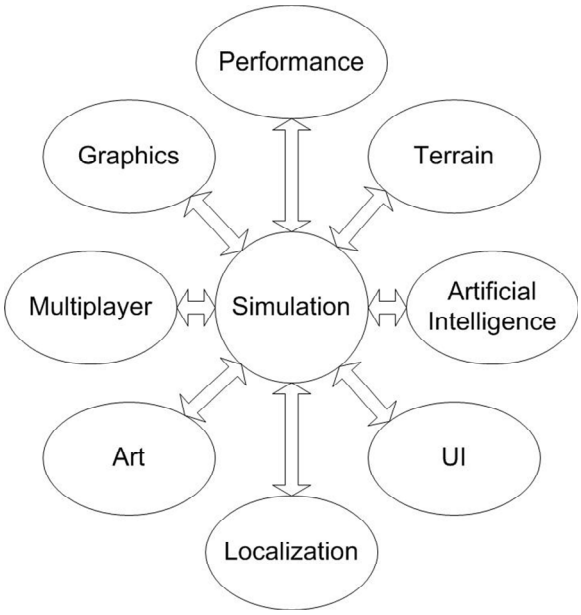

Figure 3. The "Center of the Universe"

Interface Support

To handle the interaction with all the different components of the game, the SimEngine must have a robust way to interface with these components. The majority of this interaction is handled through a C++ class structure interface, complete with unit conversions, error handling and multiple parameter indices.

Chicken and the Egg (Terrain/Sim Interaction)

One of the biggest problems with any simulation is the interaction between the simulated object and the world in which it is being simulated. Because of the limitations of hardware resources, it is not reasonable to expect to have the entire world "loaded" at all times. Therefore, there exists a classic "Catch 22" with respect to the SimEngine and the Terrain System: The terrain system must know where the aircraft is to know what terrain needs to be loaded, but the aircraft must know where the ground is to determine if the aircraft has crashed, landed or otherwise should be interacting with the ground.

The terrain system is designed to perform *asynchronous* loading whenever possible to minimize the stutters (instantaneous drop in frame rate) due to terrain loading. However, the terrain may need to be *synchronously* loaded, or force loaded, at the same time the SimEngine skips certain functions until a valid return from the terrain system is achieved. This way, during the especially difficult period of initialization, we can minimize the visual artifacts associated with this Catch-22 phenomenon.

Backwards Compatibility

Developing new features for a product that has 20 years of previous versions can require significant effort to ensure compatibility between aircraft of older versions to the newly designed feature. For instance, suppose we wish to develop a more robust turbine engine simulation routine. Not only would we need to consider the consequences of this change to other aspects of the product (artificial intelligence, special effects, gauge/panel interaction, etc.), but we would also need to consider how a previous-version MSFS or MSCFS aircraft might behave with regard to this new feature.

We normally handle this issue by establishing an upgrade path for aircraft developed for previous versions of the product that would not fit with the new routine. This allows users to utilize their older aircraft with the new version of the product, but at the same time we quietly upgrade their old aircraft behind the scenes.

This upgrade capability is sometimes as difficult a problem to solve as developing the new feature itself. Referring back to our example above, if our new turbine engine routine requires additional parameters to be defined for the engine to work correctly in the SimEngine, we must figure out how to estimate those missing parameters for older aircraft. However, we must also make sure the estimated parameters will provide similar aircraft performance in the new routine as it had in the old.

Delta T

One significant difference between Flight Simulator and Combat Flight Simulator is the time slice being simulated. As a general rule, the smaller the change in time between simulation cycles (ΔT), the more accurate the resulting aircraft position and attitude is. However, we must also concern ourselves with the limited computing power available. This means we must find the ΔT "sweet spot" which provides enough fidelity to prevent numerical instabilities, but doesn't gobble up too much processing resources. This is especially

critical when simulating large numbers of aircraft (or other objects, for that matter).

Flight Simulator incorporates a subroutine which computes what ΔT should be based on the current frame rate the program is achieving. As the performance of the program degrades, the simulation dynamically alters the time slice to take less processing time between cycles.

Combat Flight Simulator utilizes a fixed ΔT of 16Hz throughout the entire application. We discovered, however, that during certain high-dynamic maneuvers, numerical instabilities exist when attempting to simulate aircraft at this low a frequency. We therefore incorporate a doubling of certain SimEngine runtime functions to halve the ΔT – ultimately achieving a fixed ΔT of 32 Hz - and solving the numerical instability problem.

There are advantages and disadvantages to both methodologies, and ultimately either one is a valid and reasonable way to simulate an aircraft in real-time.

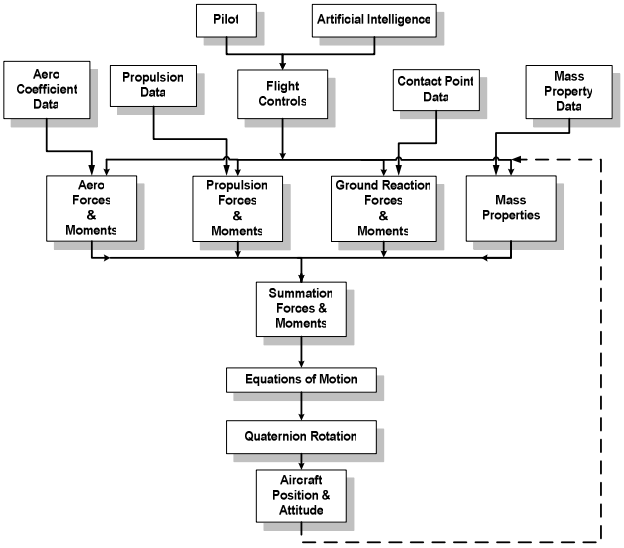

Figure 4. Simulation Loop

FLIGHT DYNAMICS

There are numerous textbooks outlining the fundamentals of aircraft aerodynamics, stability and control, and handling qualities[3,4,5]. However, most of these references focus on the design of aircraft rather than the simulation of them. There are a few references available which describe the ways in which to simulate aircraft, but most of these focus only on the aerodynamic principles involved[6,7,8]. The purpose of this section is to provide the reader with a basic view of how MSFS and MSCFS incorporate the classic aerodynamic principles of simulation – it is NOT intended to explain the theory behind these principles nor derive the equations of motion. More detailed discussion of the various supporting simulation roles will be discussed in later sections as well.

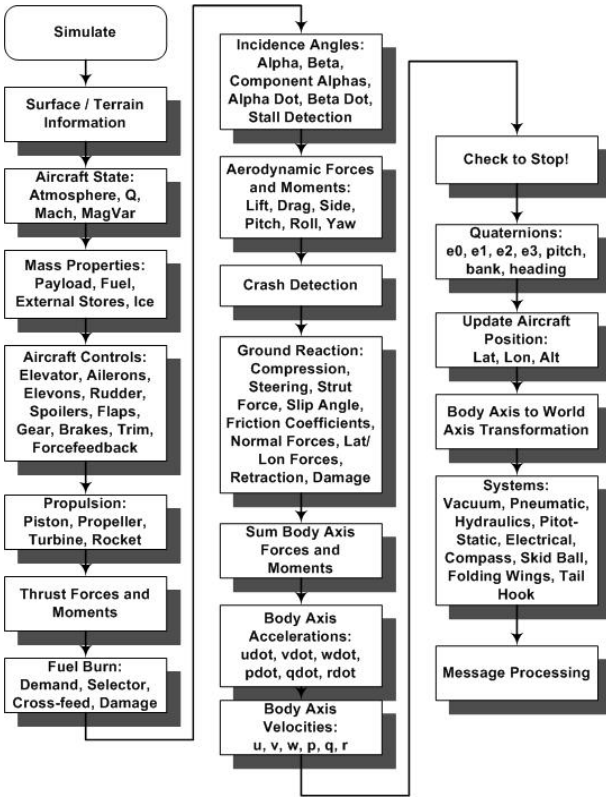

Figure 5. Detailed Simulation Components

Six Degrees of Freedom (6-DOF)

The SimEngine is based on the core concept of simulating in the 6-DOF space (rotational – pitch, roll and yaw; and translational – longitudinal, lateral and vertical). This provides the fidelity required to simulate any possible aircraft orientation and position in a three-dimensional (3-D) environment.

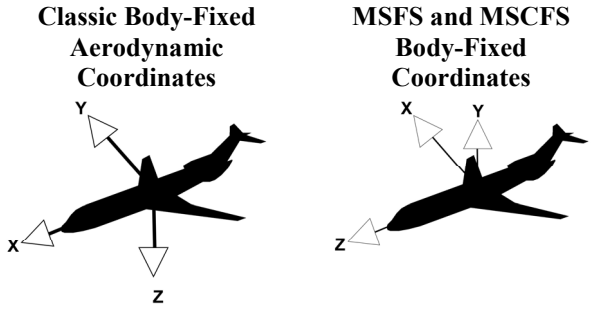

Figure 6. Comparison of Classic and FlightSim Body-fixed Aircraft Coordinate Systems

In aircraft simulation, there are two main axis systems used to describe aircraft motion: "Body-fixed" and "Earth-fixed" coordinates.[3] Most calculations are done in the body-fixed axis to simplify the equations, the results then transformed to the Earth-fixed coordinate system for proper updating of aircraft position and orientation in the visual environment. The Earth-fixed system is considered an inertial reference frame, neglecting the rotational velocity of the Earth and conforming to Newton's laws of motion.

In addition, the SimEngine makes use of the "flat earth assumption," which essentially aligns the acceleration of gravity along the vertical earth-fixed system. This is another common assumption used to simplify the equations of motion, and is generally negligible when not dealing with orbital mechanics.

In contrast to most aerodynamic applications, the coordinate system used in MSFS and MSCFS is based on the Left Hand rule – derived from the same coordinate system used to describe computer graphics, (See Figure 6). This coordinate system uses the Z-axis as the longitudinal axis, positive out the nose of the aircraft, the X-axis as the lateral axis, positive out the starboard wing, and the Y-axis as the vertical axis, positive up. However, for simplicity, the equations, diagrams and references made to axis in this paper maintain the classic aerodynamic representation of the Right-Hand rule: X-axis the longitudinal axis, positive forward; Y-axis the lateral axis, positive starboard; and the Z-axis the vertical axis, positive down.

Equations of Motion

As stated earlier, the derivation for the equations of motion will not be attempted here; rather, the resulting equations, and those used in the SimEngine, are as follows[3]:

The force equations in the airplane body-fixed axis system XYZ:

$$m(\dot{U} - VR + WQ) = -mg\sin\Theta + F_{A_X} + F_{T_X} \quad (1)$$

$$m(\dot{V} + UR - WP) = mg\sin\Phi\cos\Theta + F_{A_Y} + F_{T_Y} \quad (2)$$

$$m(\dot{W} - UQ + VP) = mg\cos\Phi\cos\Theta + F_{A_Z} + F_{T_Z} \quad (3)$$

Where:

- m = aircraft mass (slugs)
- g = gravitational constant (ft/s^2)
- Θ = Airplane pitch angle (rad)
- Φ = Airplane bank angle (rad)
- U, V, W: Components of airplane velocity along XYZ (ft/s)
- P, Q, R: Airplane angular velocities about XYZ (rad/s)
- F_{Ax}, F_{Ay}, F_{Az}: Aerodynamic force components along XYZ (lbs)
- F_{Tx}, F_{Ty}, F_{Tz}: Thrust force components along XYZ (lbs)

And the moment equations in the airplane body-fixed axis system XYZ:

$$I_{XX}\dot{P} - I_{XZ}\dot{R} - I_{XZ}PQ + (I_{ZZ} - I_{YY})RQ = L_A + L_T \quad (4)$$

$$I_{YY}\dot{Q} + (I_{XX} - I_{ZZ})PR + I_{XZ}(P^2 - R^2) = M_A + M_T \quad (5)$$

$$I_{ZZ}\dot{R} - I_{XZ}\dot{P} + (I_{YY} - I_{XX})PQ + I_{XZ}QR = N_A + N_T \quad (6)$$

Where:

- I_{XX}, I_{YY}, I_{ZZ}: Airplane Moments of Inertia about XYZ (slug-ft^2)
- L_A, M_A, N_A: Aerodynamic moments about XYZ (ft-lbs)
- L_T, M_T, N_T: Thrust moments about XYZ (ft-lbs)

Note that the cross-inertia terms have been eliminated for simplicity, assuming a symmetrical mass distribution with respect to the fore and aft plane of symmetry, $I_{yz} = I_{xy} = 0$.

After the aerodynamic and thrust forces and moments have been calculated, these equations are then solved for the translational and rotational accelerations in each simulation frame. The linear and angular velocities can then be found through numerical integration of these accelerations. A first-order, modified Euler simple integration method is used:

$$P_n = P_{n-1} + \left(\frac{\dot{P}_{n-1} + \dot{P}_n}{2}\right)\Delta T \quad (7)$$

Where:

- P_n = New Velocity
- P_{n-1} = Old Velcotiy
- $\dot{P}_n$ = New Acceleration
- $\dot{P}_{n-1}$ = Old Acceleration
- ΔT = Time Delta

The resulting position and orientation can then be found with the following methodology.

Quaternion Transformation Method

Because all the calculations done to this point are referenced to the body-fixed coordinate system, some form of transformation back to the Earth-fixed axis system is required. There are two main methodologies used when simulating aircraft:

- Euler Method
- Quaternion Method

There are advantages and disadvantages to both approaches, but we choose to use the Quaternion Transformation Method mainly to avoid singularities at $\Theta = \pm 90^{\circ}$. The details and derivations of these equations are beyond the scope of this paper, but a general description of how the two methods differ is given in Figure 7[6].

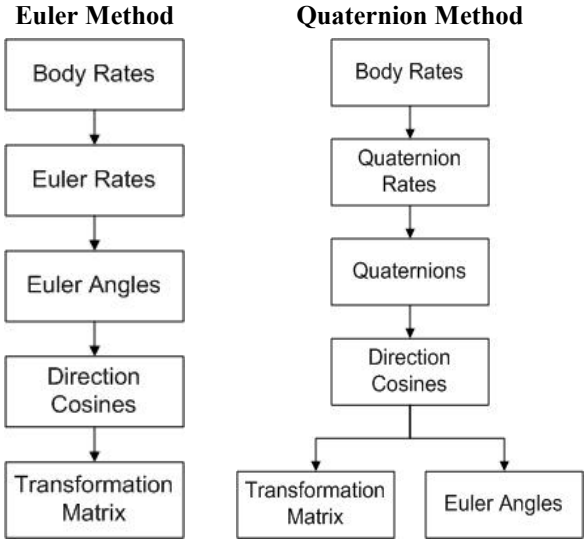

Figure 7. Euler and Quaternion Transformation Methods

Aerodynamic Coefficients

For the most part, the stability and control characteristics of the aircraft flight models used in MSFS are defined by non-dimensional aerodynamic coefficients. These coefficients are the classic linear-representation of aircraft aerodynamic forces and moments. Used in conjunction with the aircraft geometry, mass and dynamic pressure, these values can be solved for the over-all forces and moments for an aircraft during each time-step interval. The following equations are used to define the longitudinal aerodynamic force and moment coefficients[3]:

$$F_{A_X} = -D = -C_D \bar{q} S \tag{8}$$

$$F_{A_Z} = -L = -C_L \bar{q} S \tag{9}$$

$$M_A = C_M \bar{q} S \bar{c} \tag{10}$$

Where:

- D = Drag force (lbs)
- L = Lift force (lbs)
- q = dynamic pressure (lbs/ft^2)
- S = Wing surface area (ft^2)
- c = Wing mean geometric chord (ft)
- C_D = Airplane drag coefficient
- C_L = Airplane lift coefficient
- C_M = Airplane pitching moment coefficient

The lateral-directional aerodynamic force and moment coefficients are defined by:

$$L_A = C_l \bar{q} S b \tag{11}$$

$$F_{A_Y} = C_Y \bar{q} S \tag{12}$$

$$N_A = C_N \bar{q} S b \tag{13}$$

Where:

- b = Wing span (ft)
- C_l = Airplane rolling moment coefficient
- C_Y = Airplane side force coefficient
- C_N = Airplane yawing moment coefficient

The non-dimensional aerodynamic coefficients can then be broken down in to more finite components, which are defined on a per-flight model basis. These are the so-called steady-state derivatives[3]:

$$C_D = C_{D_0} + C_{D_\alpha}\alpha + C_{D_{i_h}} i_h + C_{D_{\delta_e}} \delta_e + \Delta C_{D_{flap}} + \Delta C_{D_{Spoiler}} \tag{14}$$

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{i_h}} i_h + C_{L_{\delta_e}} \delta_e + \Delta C_{L_{flap}} + \Delta C_{L_{Spoiler}} \tag{15}$$

$$C_M = C_{M_0} + C_{M_\alpha}\alpha + C_{M_{i_h}} i_h + C_{M_{\delta_e}} \delta_e + \Delta C_{M_{flap}} + \Delta C_{M_{Spoile}} \tag{16}$$

$$C_l = C_{l_0} + C_{l_\beta}\beta + C_{l_{a_e}} \delta_a + C_{l_{er}} \delta_r \tag{17}$$

$$C_Y = C_{Y_0} + C_{Y_\beta}\beta + C_{Y_{a_e}} \delta_a + C_{Y_{er}} \delta_r \tag{18}$$

$$C_N = C_{N_0} + C_{N_\beta}\beta + C_{N_{a_e}} \delta_a + C_{N_{er}} \delta_r \tag{19}$$

Where the subscripts denote:

- *0*: At zero-lift state
- α: Due to angle of attack
- i_h: Due to horizontal tail incidence angle
- δ_e: Due to elevator deflection
- *flaps*: Due to flaps
- *spoiler*: Due to spoiler

- β: Due to sideslip angle
- δ_a: Due to aileron deflection
- δ_r: Due to rudder deflection

The non-dimensional quasi-steady perturbed aerodynamic coefficients are also used, but for simplicity are not included here.

High Alpha/Beta Table Look-ups

Although most of the terms defined for an aircraft flight model's stability and control characteristics are of the linear non-dimensional form, two-dimensional tables are also used to capture the non-linear behavior of aircraft at higher angles of attack and sideslip. The $C_{L\alpha}$ and $C_{M\alpha}$ values used in the SimEngine are obtained through linear interpolation of a table for a given angle of attack, and are the only exceptions to this rule. All the other non-dimensional aerodynamic coefficients are non-linearized through the use of tables generated specifically for each coefficient. An interpolation routine is used to find the non-linear effect of a given coefficient, which is simply a scalar value multiplied by the base linear-form of the coefficient, to obtain the correct behavior in the non-linear range.

In essence, we incorporate both a linear non-dimensional coefficient methodology for simplicity with a table look-up methodology to capture the non-linear effects of aircraft behavior at extreme incidence angles, of the form:

$$C_X = C_{X_{Linear}} * K_{C_{X_{\alpha-Mod}}} * K_{C_{X_{\beta-Mod}}} \tag{20}$$

Where:

- C_X = Aerodynamic Coefficient of Interest
- $C_{XLinear}$ = Linear Form of Coefficient
- $C_{Xa\text{-}Mod}$ = Angle of Attack Non-linear Modifier
- $C_{Xb\text{-}Mod}$ = Sideslip Non-linear Modifier

Mach and Ground Effects

The effect of Mach number on the aircraft coefficients, drag, power and stability and control are handled through a similar table look-up interpolation routine to the one used for high alpha/beta corrections:

$$C_X = C_{X_{Linear}} * K_{C_{X_{MachEffect}}} \tag{21}$$

Ground effect is handled the same way. However, we utilize the ratio of height AGL to aircraft wing span to simulate the additional lift experienced in ground effect.

Propwash

In addition to the previously stated non-dimensional aerodynamic coefficients, we also define coefficients which represent the control surface power or stability with respect to an induced dynamic pressure.

PROPULSION

A significant portion of simulation time is spent dealing with the propulsion characteristics for a given aircraft. There are three main types of propulsion simulated to a high fidelity level:

- Piston Engines
- Turbine Engines
- Turboprop Engines

In addition to the engine modules, a Propeller module is also required to convert shaft horsepower or torque to thrust.

There are two other, lower fidelity engine models currently employed:

- Rocket Engine
- Helicopter Engine

These two will not be discussed in this paper, as they are either not prominent or relevant to this discussion.

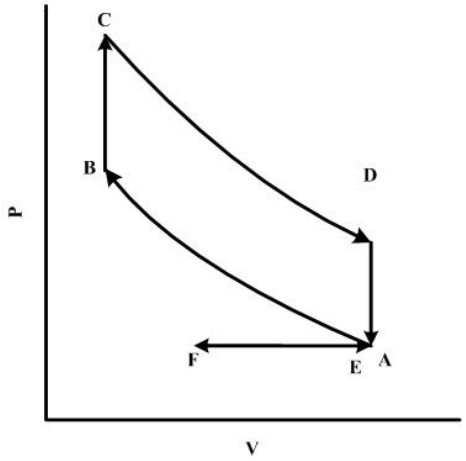

Figure 8. Piston Engine Otto Cycle

Piston Engine Simulation

One of the most remarkable aspects of the SimEngine is the high-fidelity piston simulation module employed. This module is designed to simulate the piston engine Otto cycle (see Figure 8).

The Otto cycles shown are:

- Intake (F-E)
- Compression (A-B)
- Combustion (B-C)
- Power (C-D)
- Exhaust (D-E)

Through the use of efficiency tables, flight model defined variables and propulsion theory, the engine brake power can be determined.

Figure 9 provides a component breakdown of the individual simulation routines used to simulate the piston engine. Each component of the Otto cycle is simulated on a per-cylinder basis, then multiplied by the number of cylinders to obtain the total shaft torque available.

Table look-ups are then performed for the engine friction and mechanical efficiency for a given RPM, which is then used to modify the shaft torque output. Finally, the engine brake power can be found by multiplying the engine shaft torque by the engine omega, or engine speed, in the form of radians per second.

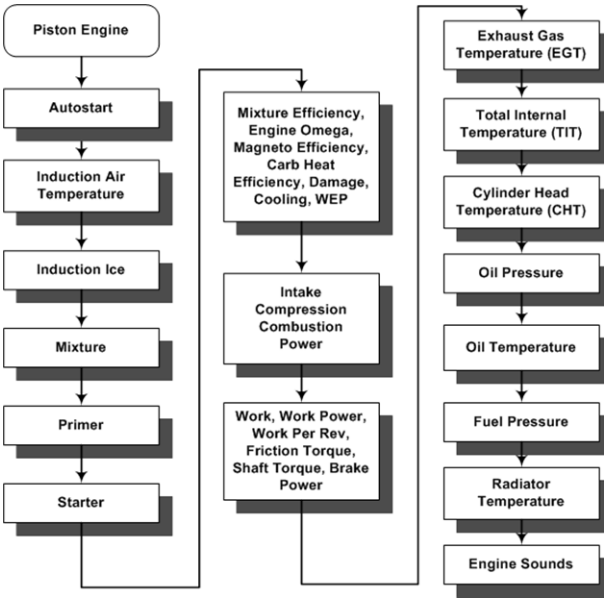

Figure 9. Piston Engine Simulation

Turbine Engine Simulation

The turbine engine is simulated quite differently from the piston engine. Instead of trying to simulate the thermodynamic properties of a turbine, which are quite complicated and involved, the SimEngine incorporates extensive use of normalized look-up tables that can be scaled by the required thrust output of the engine. Not only does this make the turbine code simpler and easier to maintain, but it also allows for simplified input to the flight model data file.

Figure 10 provides a detailed component breakdown of how the turbine simulation code works. Though we have taken a simplified approach to simulating turbine aircraft, we are able to provide the parameters necessary for output to the cockpit display through empirical and normalized tables.

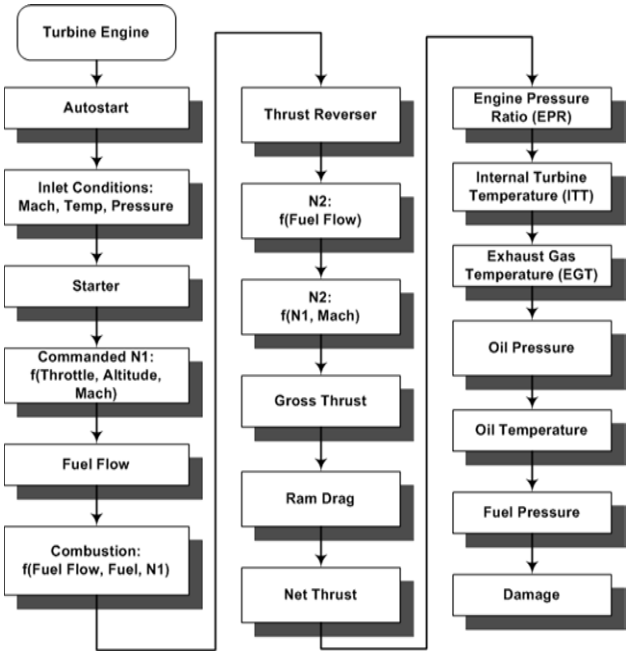

Figure 10. Turbine Engine Simulation

Turboprop Engines

The turboprop engine simulation is really a combination of the turbine engine simulation and the propeller simulation code (see Figure 11). Though there are a few unique calculations required (N1 and N2 are calculated using a different set of normalized tables, and shaft torque and power must be calculated from the turbine engine code), for the most part this simulation routine is as simple as the turbine engine simulation.

Propeller Simulation

As stated earlier, the propeller simulation is used in combination with a piston engine or turboprop engine powered aircraft. Though seemingly a simple task, simulating prop behavior is really quite complicated. Not only must we calculate the resulting thrust output of the prop, but there are numerous other steps required to properly simulate prop behavior (constant speed props, feathered props, prop sync, etc). Figure 12

provides a more detailed component breakdown of the steps used in the propeller simulation routine.

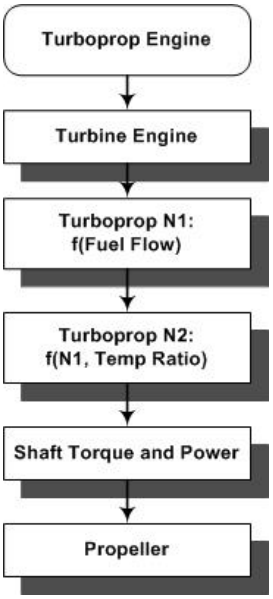

Figure 11. Turboprop Engine Simulation

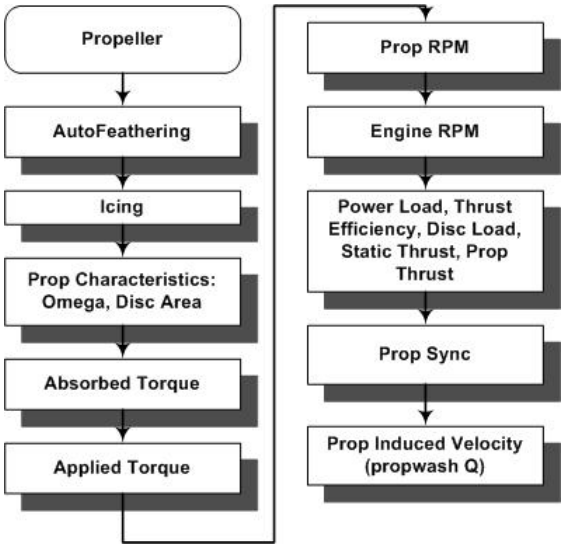

Figure 12. Propeller Simulation

GROUND REACTION

One of the most complicated and difficult parts of aircraft simulation is the interaction of the aircraft with the ground. There are numerous problems associated with this issue, but the main ones are:

- Chicken-or-the-egg
- Collision detection
- Rolling surface behavior
- Steering behavior

Some of these problems are addressed below. Figure 13 provides a more detailed look at the ground reaction simulation routine.

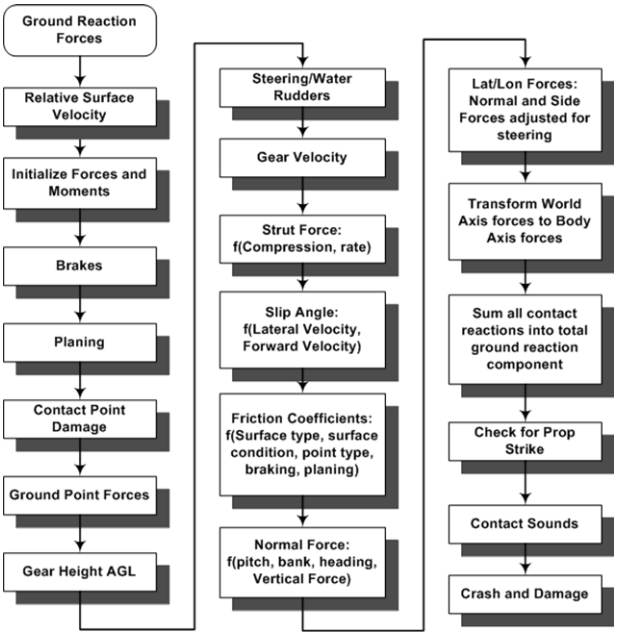

Figure 13. Ground Reaction Simulation

Contact Points

Though somewhat unintuitive, the simulation utilizes a series of defined contact points to represent the "hard points" of an aircraft. These hard points are not to be confused with the *visual model representation* of the aircraft – rather, these points are a subset of the visual model and defined in a completely separate manner.

Each point has 15 parameters associated with it, and can be used to define a landing gear point, a float point, a skid point, or a contact point. The type of point defined can influence how its reaction with the surface is simulated.

- Gear: Modeled as a spring. Optional parameters include retraction times, extension times, max steer angle, static compression, compression ratio, and damping ratio.
- Contact point: "hard point" used to detect aircraft extremity coming in contact with the ground, surface object or vehicle.

Gear As Springs

Each contact point is considered a spring/mass damper system in which reaction with the ground is absorbed according to the static compression and damping ratios given in the contact point's definition:

$$F_{Gear} = K*(compression) + B*(compressionRate) \quad (22)$$

Where:
- K = f(static compression)
- B = f(K, damping ratio)

Typically, the points defined as *scrape points* are not given a static compression or damping ratio, hence they are static hard points that do not deflect when a force is applied.

Steering

The maximum steer angle defined for a given contact point determines not only how much steering capability exists, but a value of 180 indicates that the particular point is a full-caster wheel. We use the location of the point to help determine specifics about the point, such as if the point is a tail wheel or a nose-wheel.

Friction and Braking

The friction coefficient used to determine braking efficiency is based on a combination of factors:

- Gear type
- Surface type
- Surface condition

We also calculate a slip angle based on the steer angle and velocity of the given point. This slip angle is then used to determine the side force friction for each point.

There is a natural rolling resistance friction scalar employed to simulate the friction "hump" observed when initiating movement from rest. The braking coefficient of friction is defined in the flight model, and is then scaled by the percentage brake position on to determine the braking resistance at any given brake position.

Collision Detection

Probably the most difficult part of simulating aircraft is the methods required to detect an aircraft collision. Some of the main challenges are:

- Determining difference between a crash and a hard landing.
- Quickly changing terrain – large slopes or cliffs.
- Detecting collisions with non-ground objects (trees, buildings, other aircraft, etc).

The significance of this problem is exasperated by the fact that we must look for collisions on every point, on every aircraft, in every simulation frame, coupled with the need for collision detection when close to the ground – which is also the time the graphics engine is taxing the computer hardware the most. These facts behoove us to be very conscious of the calculations we are performing, as we must always strive to achieve an efficient solution in this area to maintain acceptable game performance.

SYSTEMS

One of the most intriguing and challenging aspects to the SimEngine is the various aircraft systems simulated. Although many of these systems tie directly into the propulsion simulation (oil pressure, exhaust gas temperature, etc.), many other systems are modeled as completely separate systems. The major systems of note are:

- Electrical system
- Fuel system
- Oil system
- Cooling system
- Hydraulics system
- Vacuum system
- Pneumatic system
- Autopilot
- Radios
- Gyroscopes
- GPS
- Exits
- Tail-hooks
- Folding wings

With each new release of MSFS or MSCFS, we make decisions about which new system to implement based on the capabilities of the new aircraft we are going to simulate. Through extensive research and a thorough design process, we attempt to simulate these systems as accurately as possible.

However, we also must be aware of usability issues that may arise, and in some cases include the design for "automatic" modes if a particular system is very complex or cumbersome to use in a simulated environment. This inevitably makes it easier to implement follow-on systems in future versions of the product, and allows for a more robust and realistic simulation experience for users of all experience levels.

DAMAGE MODELING

Upon the advent of CFS1.0, it became clear that a more robust method of simulating aircraft damage was necessary. The main areas we incorporate damage effects are:

- Flight model
- Propulsion
- Gear
- Systems
- Armament

The overall methodology we use is to keep track of damage levels for specific components of the aircraft (left wing, right wing, tail, engine, fuel system, canopy, etc). We then apply scaling factors to the various systems for systems damage, and modify the aerodynamic coefficients for aircraft structural damage. We also attempt to make the damage effects gradual rather than step-wise.

We have improved this system version to version, but we still find numerical instability problems when trying to simulate aircraft with major structural damage – applying too great a scaling factor to the aerodynamic coefficients can result in un-desirable characteristics (i.e., wingless aircraft *gaining* altitude).

REALISM FACTORS

Though the simulation and the respective components therein are designed with the utmost realism in mind, we have made significant efforts to simplify certain parts of the simulation to ease user workload in the cockpit. These *realism factors* are also included to provide an easier transition for novice users and/or pilots who are new to the world of aviation and flying.

- Flight model
- Control inputs
- Crash tolerance
- Ground handling
- Weapon effectiveness
- Gyroscopic effects
- Stall/spin behavior
- Stress damage effects
- Fuel weight and balance
- Automatic engine controls

Without these factors, the proper operation of some aircraft can be very difficult for even the most experienced MSFS/MSCFS enthusiast or pilot.

CONCLUSIONS

Through over 20 years of consecutive development, Microsoft Flight Simulator and Microsoft Combat Flight Simulator have continuously improved the techniques and methodologies used to simulate aircraft behavior.

By taking every step possible to accurate simulate aircraft flight dynamic behavior, propulsion, ground reaction, systems and damage effects, we not only achieve a flexible and robust coding platform, but we allow ourselves the opportunity to keep striving to be *As Real As It Gets*!

ACKNOWLEDGEMENTS

This work would not have been possible without the contributions in design and development by Mike Schroeter, lead developer for aircraft simulation in MSFS. His many years of involvement have really made the SimEngine what it is today. I would like to thank Dave Denhart for verifying the historical accuracy of the paper, and Bruce Williams for editorial review. I'd also like to thank Carl Edlund, Brian Syme, Leon Rosenshien and the rest of the Microsoft Flight Simulator and Microsoft Combat Flight Simulator teams for all their hard work and dedication to making such great products.

REFERENCES

1. Artwick, Bruce A., A Versatile Computer-Generated Dynamic Flight Display, Aviation Research Laboratory, Institute of Aviation, University of Illinois at Urbana-Champaign, prepared for Engineering Psychology Programs, Office of Naval Research, May 1975.
2. Anon., "Bruce Artwick is still flying: and thanks to his software, many of us can too," Alumni News, Department of Computer Science, University of Illinois at Urbana-Champaign, Spring 1996.
3. Roskam, Jan, Airplane Flight Dynamics and Automatic Flight Controls, Part I, DARCorporation, Lawrence, KS, 1995.
4. Perkins, C. D., and Hage, R. E., Airplane Performance, Stability and Control, Wiley, 1949.
5. Nelson, Robert C., Flight Stability and Automatic Control, McGraw Hill, 1989.
6. Rolfe, J. M. and Staples, K. J., Flight Simulation, Cabridge University Press, 1986.
7. Stevens, Brian L. and Lewis, Frank L., Aircraft Control and Simulation, John Wiley & Sons, 1992.
8. Hanke, R., "The Simulation of a Large Jet Transport Aircraft Volume I: Mathematical Model," NASA CR-1756, March 1971.