Essentials of Aircraft and Missile Simulation

By

M. Clay Harden

A MASTER OF ENGINEERING REPORT

Submitted to the College of
Engineering at Texas Tech
University in Partial Fulfillment of
The Requirements for the Degree of

MASTER OF ENGINEERING

Approved:

_____
Dr. J. Borrelli

_____
Dr. A. Ertas

_____
Dr. T. Maxwell

_____
Dr. M. Tanik

12 October, 2002

TABLE OF CONTENTS

"Aircraft and space-vehicle simulation for engineering design and training is perhaps the most important application for continuous-system simulation; simulation here replaces flight experiments, which are expensive, dangerous, and/or not yet possible"

— Granino A. Korn
John V. Wait

CHAPTER 1. INTRODUCTION

This paper will introduce flight vehicle simulation as a powerful tool to be used in the design, development, and testing of aircraft and missiles. Some tenets applicable to simulation in general are discussed, but the specific application of interest is digital modeling of flying vehicle characteristics and dynamics.

Though the art and science of simulation has some pitfalls and disadvantages, with care these can be overcome. The potential advantages and cost efficiencies to be gained through the use of simulation far outweigh the drawbacks. Detailed, full mission simulation is essential to the development of modern missile and airplane projects, both because of their typical complexity and their tremendous cost. Many details of integrating hardware, software, and electronic components of an aircraft or missile system cannot be fully appreciated until the integration process begins. Likewise, many issues surrounding the flight of such complex systems cannot be accounted for through static analysis and must be studied during and after the first few flights of a new system. Simulation allows for these processes to be better understood in the design and development phases of a program, saving both time and money, and contributing to flight safety during flight test.

The author's experience with simulation and modeling includes programming, maintaining, and using large digital simulations of the C-130J and C-5 military transport airplanes and the Tactical Tomahawk cruise missile. In both cases, simulation tools were invaluable during development and testing of the flight vehicles.

On the C-130J program, simulation was used to prepare for flight test, to familiarize pilots with new features in the aircraft, and in one case to perform testing for certification credit. On the Tactical Tomahawk program, simulation was integral in every aspect to the development program. Simulation was used to design guidance and control algorithms, to test embedded software, to exercise new hardware components, and to design test missions to extract maximum value from each flight.

CHAPTER 2. BACKGROUND

Many articles and books have been written on the topic of simulation of various kinds and for a range of purposes (evidence the references for this paper alone). Digital simulation is now used in all types of fields including social sciences and economics [1]. Certainly firms involved in the physical sciences widely employ simulation to design or predict behavior of complex systems. For example, simulation is used in nuclear power plant design [1], weather prediction, and aircraft and missile design.

Existing work explores simulation from various approaches and industries. Many texts focus on special-purpose programming languages designed for simulation, such as DYNAMO [1], GPSS [2], and SIMSCRIPT [2]. Some works examine specific approaches to simulation, such as Franta's "The Process View of Simulation" [3], and others specifically deal with continuous or discrete system simulation.

This paper discusses the topic of digital simulation of aircraft and missiles and has been prepared with several audiences in mind. One audience is engineering management, to whom the paper will present the benefits of maintaining a detailed, high fidelity simulation of a flight vehicle system under development. The specific benefits depend on the level of program maturity, but include technical risk reduction early in a program [4] and reduced test requirements later in system development.

Another audience is the practicing engineer having indirect or infrequent interaction with simulation and the results thereof. To this audience, the paper seeks to enrich the

understanding of the methods, purposes, advantages, and limitations of simulation. A better awareness of these topics will allow engineers to make more informed decisions when faced with data produced by simulation. Both managers and engineers will benefit from a description of the types of simulation in common use, the structure and components of a simulation program, and the strengths and weaknesses of simulation.

CHAPTER 3. WHY SIMULATION?

In his foreword to a text on simulation, Gerald M. Weinberg complains "Of the thousands of ways computers have been misused, simulation tops the list" [5]. Several texts lament the expense and time required to develop and use simulation to solve problems [4]. Nevertheless, the same texts suggest simulation as an essential design tool that should be among the first activities in the product life cycle [4]. Here, some of the negative and positive aspects of simulation as a design tool and problem solution are discussed.

3.1 Disadvantages of Simulation

Many authors have catalogued some of the disadvantages of simulation:

1) Simulation is expensive when the manpower and computer time required to develop and maintain it are considered [5]. On the C-130J program, roughly ten man-years were devoted to developing the aerodynamic and flight control simulation of the aircraft. This effort was required in spite of the use of existing C-130 simulation models and databases as a starting point. The manpower devoted to simulation development on Tactical Tomahawk was on order of magnitude larger.
2) Satisfactory simulation models are time consuming to develop. A simulation of a system cannot be rapidly assembled to provide a quick answer. Data collection, model development, analysis, and report generation may require years to complete for a complex system [4]. Therefore, when simulation is an important part of a development program, it must be given consideration early and remain adequately supported.

3) Models used in simulation are difficult to validate. Validation is the process of making sure the computer model accurately represents the system being studied. When the system being modeled does not yet exist, this task can become formidable [4]. The opinions of experts and the analysis of similar existing systems and components must be considered.

4) Critical assumptions made in the model development phase may be forgotten or ignored as time passes [5]. Analyses of circumstances in violation of the founding assumptions of a model may be performed. For example, a vehicle aerodynamic model may be developed assuming structural flexibility effects on aerodynamics are negligible. If it is later found that structural flexibility is substantial, the aerodynamic model must be updated. If the model is not updated, results from the simulation are not valid. A danger is that the model may continue to be used in ignorance.

3.2 Advantages of Simulation

In spite of these difficulties, both practitioners and academics now rank simulation as one of the most important quantitative modeling techniques [6]. If simulation is so difficult to use properly, why is it so widely acclaimed and implemented? The answer lies in these significant advantages of simulation, as suggested by several authors:

1) Simulation allows for detection and resolution of unforeseen problems before a system is built, [4] substantially reducing program risk.

2) Development of models for use in simulation increases system knowledge and forces the modeler to bring together diverse sources of information. This process may uncover misunderstandings between various groups working on the project [4].

3) The effort to create a functional simulation may force the development of components that are lagging behind the rest of the system, thereby avoiding component latency issues during later system integration [4].

4) Simulation development during system and component design may enhance creativity in design because it allows the designer to experiment with several solutions to an issue before selecting the best performing approach [4].

5) A simulated environment and system state provides for better control over experimental conditions than real world testing allows [4,7]. For example, in testing the landing performance of an airplane in high winds, it is difficult to make actual landings in the limiting wind conditions, simply because the exact conditions needed do not frequently occur.

   If a particular airplane were designed with the capability to land in 35 knots of crosswind, one would prefer not to attempt landings in 40 knots of crosswind. On the other hand, landing in 30 knots of wind does not validate the design capability. Instead, a series of landing may be performed with 20-35 knots of crosswind. The data can then be used to develop and validate a simulation model, which can in turn confirm the capability. The author was involved in using this approach to validate C-130J crosswind capability.

6) Simulation allows for the assessment of potential system performance before an actual system is built [5]. This advantage is applicable to the design of a new missile, which may cost tens or hundreds of millions of dollars to develop and test. Developers must trust in the eventual effectiveness of the system before the funds will be spent.

7) Time expansion or contraction can be accomplished using simulation [5,7]. In the case of a nine-month spacecraft flight to Mars, time compression is clearly required for a meaningful number of simulations to be possible. For chemists studying a chemical reaction that takes less than a millisecond, time expansion is needed for results to be comprehensible [5].

8) Experimentation can be done without disturbing the real system [1,7]. In some situations, systems failures cannot be safely tested, yet reaction plans to those failures must be developed. An example is a nuclear reactor. Testing actual reactors under emergency conditions would involve excessive risk. For that reason, much of what is known about the likely behavior of nuclear reactors during accidents is derived from simulation.

9) Simulation models of a system are valuable for training users of the system. This is widely and successfully employed in the airline industry and in the military to drastically reduce training costs when placing a pilot in a new aircraft type.

3.3 Appropriate Circumstances for Employing Simulation

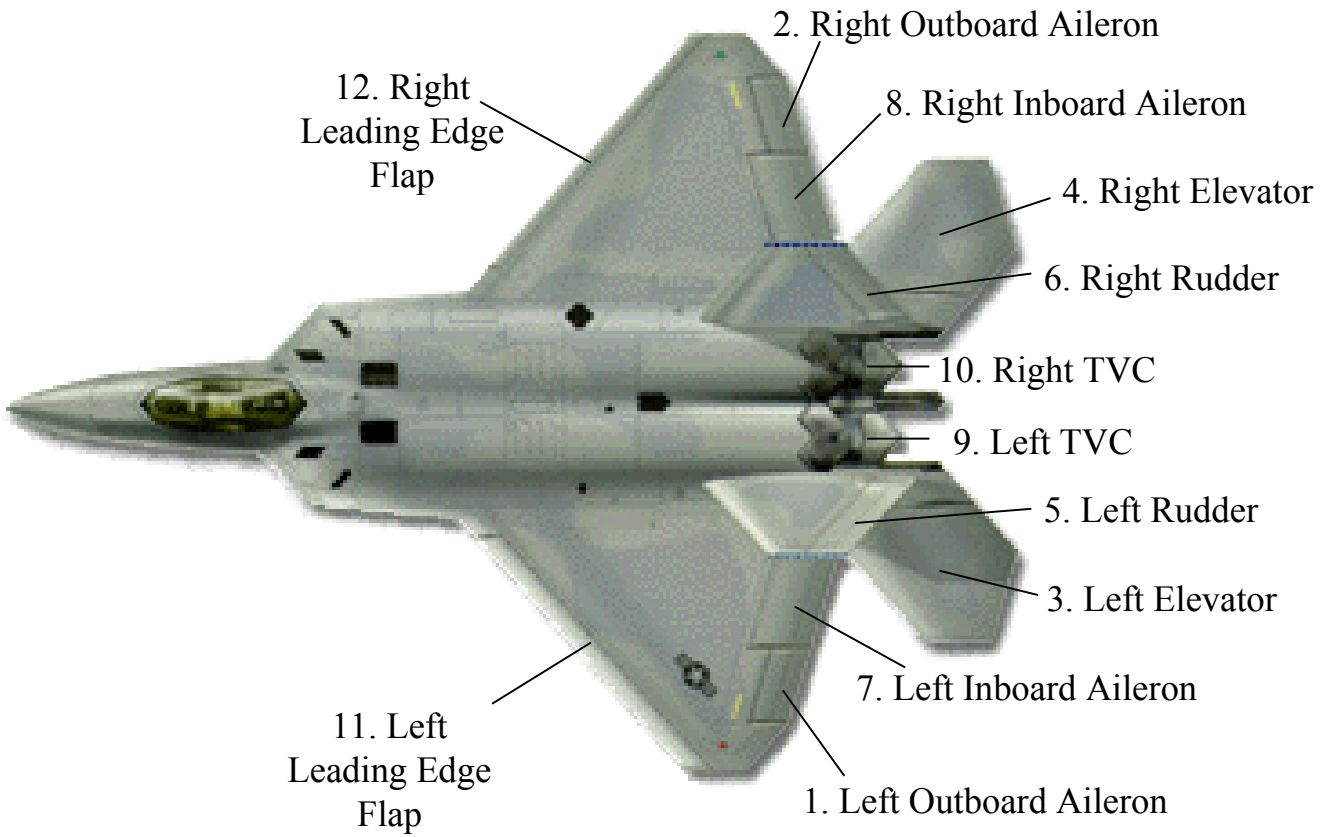
Given these positive and negative attributes of simulation, a number of qualifications have been suggested before simulation is considered as a problem-solving technique.

The real system does not exist and it is expensive, time-consuming, hazardous, or impossible to build and experiment with a prototype [2,4]. This description certainly fits the development of a new aircraft or missile in terms of expense, and time. The hazards of flying an incompletely designed aircraft are also clear. Proper design of a modern aircraft

with its extraordinary complexity might be impossible without modeling and simulation. Flight experimentation of manned aircraft has been supplanted in the last few decades by simulation [8]. Flight tests of new aircraft are now designed largely to validate the simulation models used to design the vehicle.

Mathematical models of a system have no practical analytical or numerical solutions [2]. This qualification applies to systems which are extremely complex. Vehicle dynamics described by nonlinear differential equations also fit this scenario, so this consideration applies to aircraft and missile flight dynamics [9]. As an example of a highly complex, nonlinear system, consider the flight controls of the F-22 fighter aircraft. The airplane's designers studied the flight envelope that would be required. In each flight regime, the designers defined control effectors that would allow for the required level of control authority in each axis (pitch, roll, and yaw). Twelve separate control effectors were eventually included, as illustrated in Figure 1.

Figure 1. F-22 Fighter Aircraft Control Effectors



Once all the control effectors are in place, the question becomes how they should be optimally used in each flight phase and scenario. This question is particularly acute when one considers that deflection of any control effector affects the authority of practically all the others. If the pilot commands a roll, how should the control system enact the command? In normal cruise flight, a rolling moment may be provided by differential deflection of the outboard ailerons (1 and 2), the inboard ailerons (7 and 8), the elevators (3 and 4), or most likely some combination of those inputs.

Similarly in the pitch plane, thrust vector control (9 and 10), symmetric elevator deflection (3 and 4), or even symmetric rudder deflection (5 and 6) can provide some pitching moment. This enormous complexity must be quantified and flight controls optimally scheduled across the Mach number, altitude, and attitude envelope of the aircraft. This can only be accomplished through simulation using highly accurate models of the aerodynamic effects of each of these components and their effects on one another.

Satisfactory validation of simulation models and results is possible [2]. This caveat presents a dilemma for flight vehicle simulation, because the system being simulated is typically not in existence during simulation development. Indeed, the simulation is intended as a design tool. Validation in this case must be done against expectations of designers and against other modeling data. For example, one of the most significant unknowns involved in flight vehicle design is the aerodynamic characteristics of the airframe. Aerodynamic models must be based on wind tunnel testing of reduced scale physical models of the intended vehicle. Computational fluid dynamics (CFD) is also increasingly used to fill in portions of

the aerodynamic database not adequately tested in the wind tunnel. Wind tunnel results are validated based on their repeatability. CFD results are validated based on their level of agreement with wind tunnel data.

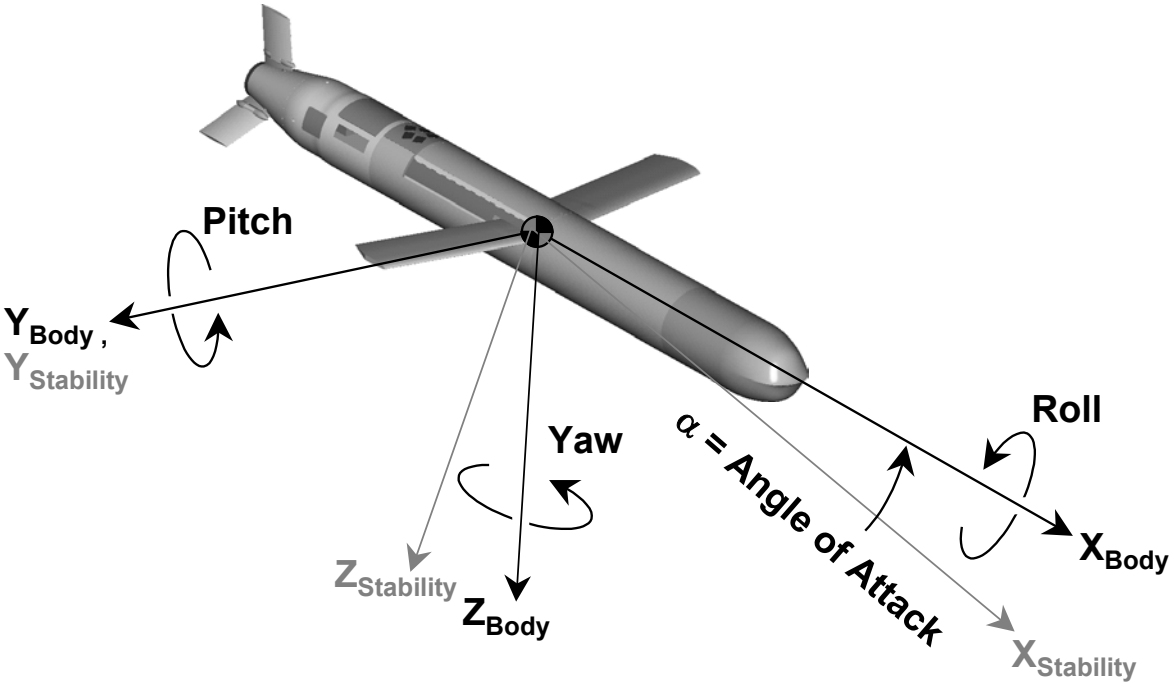
Expected accuracy of simulation results is consistent with the requirements of the particular problem [2]. This again presents a challenge for simulation of non-existent flight vehicles. For simulation to be used as a design tool, a high level of accuracy in results is essential. However, the accuracy of all the models making up the simulation cannot be finally confirmed until after the vehicle has been built and successfully flown a number of times. This leads to the conclusion that only extreme care in modeling every detail directly affecting flightworthiness of the vehicle will ensure eventual success.

CHAPTER 4. TYPES OF SIMULATION

The most common flight vehicle simulation is a time-marching program. Beginning from some default condition (such as takeoff or launch) or some in-flight equilibrium condition (found via a trim routine, discussed later) the simulation marches forward in time, computing the vehicle response to control inputs and changing environmental conditions. Ideally, the simulation will also respond to changing conditions within the vehicle, such as weight decrease due to fuel burn.

A typical aircraft or missile simulation tool may commonly be referred to as a six degree-of-freedom or 6DOF simulation. This refers to the directions of motion allowed of the simulated vehicle. A simulation allowing the full range of motion will consider both translation and rotation about each of three orthogonal or mutually perpendicular axes, as illustrated in Figure 2. The translational motions are forward, lateral, and vertical velocity. The rotations are pitch, roll, and yaw about the X_{body}, Y_{body}, and Z_{body} axes, respectively. Other axis systems may be chosen, and often are for different types of modeling. For example, aerodynamic models generally use the stability axis, which has an X-axis rotated in the pitch plane to point into the relative wind. The angle between the vehicle body and the stability axis is the angle of attack.

Figure 2. Translational and Rotational Motions of a Flight Vehicle



In decades past, it was common to simulate fewer than six degrees of freedom to accommodate computers with slow processors and scarce memory. For example, if roll performance of a fighter aircraft were to be studied, a one-degree-of-freedom simulation might be written to consider only the control inputs and aerodynamic effects contributing to rolling moment. It was also common to build two separate three-degree-of-freedom tools known as a longitudinal model and a lateral-directional model to study the dynamics of a single vehicle. A longitudinal dynamics model considers pitch, forward velocity, and vertical velocity, whereas a lateral-directional model considers roll, yaw, and side force. This convenient division of effects is possible because longitudinal dynamics are almost completely independent of lateral-directional dynamics for the typical aerospace vehicle.

Today, modeling fewer than the full six degrees of freedom is unusual because of the tremendous computing power available. It's more efficient today to model the complete vehicle in a single tool in order to minimize "overhead" portions of the computer program. Modeling the full range of motion also allows full mission simulation, which is increasingly common. For a transport aircraft, a full mission simulator might be capable of simulating electronic system and hardware startup, taxi and takeoff, cruise, landing, and system shutdown. This type of capability is particularly conducive to pilot training.

New types of distinctions are now made among simulations designed for varying purposes. Some of those details are discussed below.

4.1 All-Software Simulation

An "all-software" simulation is comprised of software models of all pertinent vehicle systems, along with the generic simulation components required to execute the time history. System embedded software is simplified or emulated, but not fully included. Some types of embedded software typically emulated for an all-software missile simulation are guidance algorithms, flight control laws, and sensor-related procedures.

All-software simulations are often the first developed on a new program. These tools may consist of tens of thousands of lines of computer code, but are still relatively simple compared to the simulations developed later in the cycle of vehicle design. The primary purpose for an all-software simulation may be for use as a design tool. Simplified emulation of guidance and control software and sensor logic allows engineers to implement notional concepts and

conduct studies to develop and improve the design of the eventual embedded software. Once a vehicle development program has reached an appropriate stage of maturity, writing of the actual embedded software may begin or accelerate. At this point, the next type of simulation becomes important.

4.2 Software-in-the-Loop Simulation

As in an all-software simulation, pertinent hardware systems are modeled with software. The difference is the emulation of embedded software is replaced with some or all of the actual embedded software. When possible, embedded software is unmodified from the intended flight version. The capability to exercise and test embedded software in a simulation environment is absolutely required considering the complexity of software included in modern aircraft and missiles.

Embedded software only understands its environment (such as altitude, velocity, and stage of flight) from information provided to it by other software or sensors. Embedded software should be exercised as if it's in a real vehicle, flying a real mission, though in fact it is running in a lab environment. Sensor inputs are generated based on software models of sensor hardware, and sent to the embedded software just as they would be in flight.

It may be desirable to maintain a software-in-the-loop simulation in parallel with an all-software tool to allow for continued study of the guidance, control, and sensor algorithms. If so, the two tools should have in common all components except the embedded software. In

this way, the amount effort required to maintain and add capabilities to both tools is minimized.

The interface between embedded software and a model of a hardware component should, as closely as possible, emulate the actual software-hardware interface. In this way, hardware will more easily be integrated into the simulation to form a hardware-in-the-loop capability.

4.3 Hardware-in-the-loop Simulation

Also known as HIL simulation, this tool generally begins with a software-in-the-loop simulation. During flight vehicle development, hardware components are gradually added as they become available. For example, software models of static and total pressure sensors used to determine altitude and airspeed may be replaced by the sensors themselves. During simulation runs, pressure lines can be connected to the sensors. Using data from the simulation atmosphere model, the actual hardware sensors can be provided with the pressures expected during the flight. In this way, the entire air data system can be tested before the first flight of the vehicle.

Ultimately, functionality of an entire aircraft or missile may be tested using the HIL flight simulation, exercising most or all hardware and electronics components. In the limit of HIL simulation capability, propulsion systems may even be operated at simulated flight conditions using actual embedded software. The remaining simulated entities are typically those contributing to the physics of airframe dynamics, including environment and aerodynamics. Hardware-in-the-loop simulation is intended as the nearest possible approximation to actual

flight, and may be used to complete component and integration testing of the first few flight vehicles.

Like the software-in-the loop tool, the HIL simulation should retain all possible simulation components of the original all-software tool. The generic components (not specific to any particular vehicle) required to perform flight simulation are discussed in a separate chapter.

4.4 Performance Prediction Simulation

Another type of simulation tool is the performance prediction program. Unlike the other simulation types, performance prediction does not use a time-marching approach. Rather, it uses the software models related to flight performance, including the aerodynamic model and propulsion characteristics, to deterministically predict elements of airframe flight performance. Considering a manned aircraft, some examples of the types of parameters predicted are maximum takeoff weight, and speed/altitude envelope in a given configuration.

Again this tool should share all possible components with other simulations in use for the vehicle. There are enormous advantages to maintaining common models among several levels of simulation from all-software to HIL. Corrections and updates to performance models, for example from new wind tunnel test data, are automatically gathered into all simulations. Disadvantages to sharing models is the logistical difficulty of ensuring model updates are consistent with all simulations using the model.

CHAPTER 5. UTILITY OF SIMULATION

The C-130J is the 21st century version of the venerable military transport. Changes to the "J" model from the earlier "H" model included new engines, more efficient, high-speed six-bladed propellers, Full-Authority Digital Engine Control (FADEC), and modern digital instrumentation, including a Heads Up Display (HUD) as a primary instrument. The effort to create a high fidelity, certifiable engineering simulation of the aircraft began rather late in the program. Nonetheless, as the simulation was transformed from a limited-use, flight segment simulation of an earlier C-130 version, it began to be used extensively. Some of its uses were stability and control analysis, performance prediction, and flight test analysis. The fidelity of the tool became sufficient to meet the Level D requirements of the Federal Aviation Administration Advisory Circular on flight simulator fidelity (FAA AC120-40B) [10] for most flight regimes.

One of the applications for the engineering simulation software was the creation of a full flight simulator for the C-130J, similar to that used by airlines to train pilots. Once this tool became available, it was used extensively by company pilots to prepare for flight tests, to practice airshow routines, and to evaluate new cockpit instrumentation functionality or symbology. During the C-130J flight test program, some required aircraft certification tests (generally of failure scenarios) were determined to be too dangerous to test in the aircraft. In some cases where realistic pilot reaction to a situation was a critical element, the FAA deemed the full flight simulator acceptable for performance of certification tests. The aircraft was eventually certified partially on the basis of those simulated scenarios.

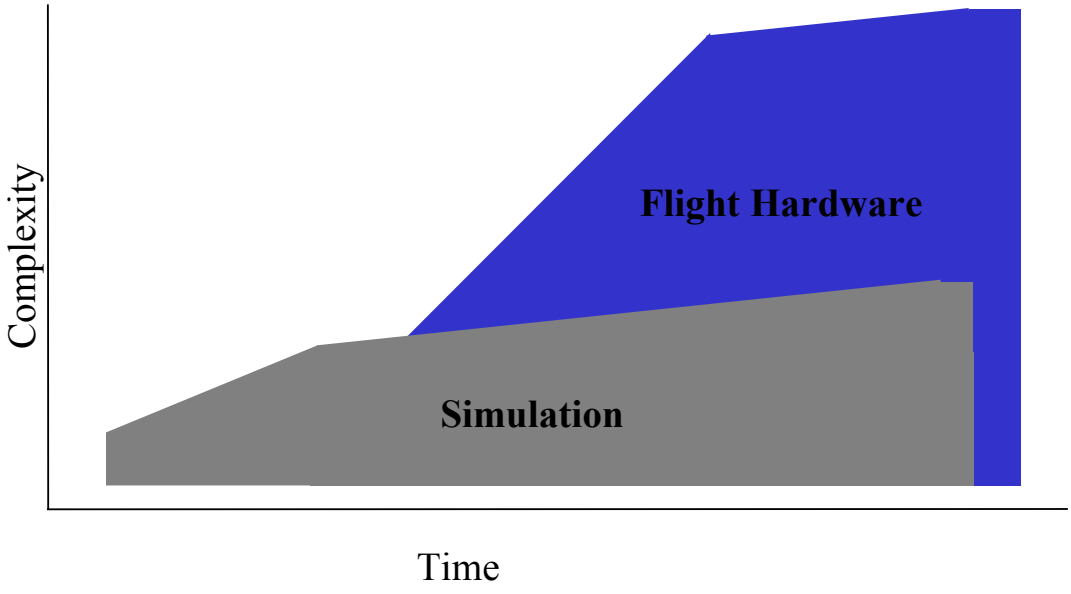
This represents an extraordinary confidence on the part of the FAA that the simulation faithfully represented the behavior of the aircraft, even in a flight situation that the aircraft had never been in. When simulated flight data from across the aircraft flight envelope are compared directly to the respective aircraft data and expected to demonstrate equivalency within predetermined bounds [11], this kind of confidence can be earned.

It is imperative that this kind of trust is built into, and expected of, flight vehicle simulation capabilities. This is particularly true in the missile business where each test flight ends in the destruction of the test vehicle. This confidence in simulation can only be realized when the simulation team is given the resources and support it needs.

5.1 Simulation Availability

Program management must ensure that creation and maintenance of a detailed, high fidelity simulation of the flight vehicle is given priority throughout the life of the vehicle, beginning in initial design stages. Figure 3 illustrates a timeline of vehicle simulation and flight hardware development.

Figure 3. Flight Vehicle Simulation and Hardware Development Timeline



This diagram, while only notional, illustrates several points. First, simulation development should begin well before the design is complete. Thus, simulation complexity and completeness should have made its most rapid growth (along with vehicle design maturity) before hardware development begins.

Ideally, the simulation effort should not start from scratch. Generic simulation components, and even a basic simulator program structure, would ideally be available from existing work. Even better, an engineering organization may independently develop a generic simulation tool to serve as a starting point for new vehicle programs to customize and specialize to their particular application. This topic is often discussed in engineering organizations and in technical writings, but has rarely been successfully implemented.

Another point illustrated by Figure 3 is that simulation complexity continues to grow, though more slowly, as hardware development continues. This is because as hardware components become available, testing of the components yields data useful for developing higher resolution models. This is particularly true of dynamic components including actuators, pumps, and most notably propulsion systems. A turbine engine alone is so complex as to require its own simulation/hardware development plan. The precise performance characteristics and dynamic behavior of a jet engine are not typically known until several prototypes have been built and tested. This, in turn, affects the overall vehicle design and simulation fidelity until relatively late in a program.

A number of uses for simulation throughout the life of a program have been highlighted. Some capabilities that a complete, mature simulation package brings to a vehicle development program are discussed below.

5.2 Subsystem Requirements Development

If a simulation program is available appropriately early in the cycle of vehicle development, it can become a valuable tool for defining and refining subsystem requirements. All-up vehicle capability requirements are often defined by a customer or by program management up front. As a vehicle is designed to meet these requirements, an evolving simulation model can be used to emulate the limiting scenarios which will define subsystem performance needed to support overall vehicle performance.

After subsystem requirements are defined and vehicle design is progressing, simulation may be used to regularly check static and dynamic performance margins against specification values. As vehicle design progresses, various components may begin present challenges in terms of meeting specified performance. Simulation at this point becomes an invaluable tool for testing the impacts of subsystem shortcomings on overall performance. In many cases, negative margin relative to specification on some components may be allowable due to positive margin in other areas. Without a detailed, high fidelity simulation to study such scenarios, program health could be jeopardized.

5.3 Software Development and Test

The avionics software used in the F-22 fighter aircraft is said to have around 1.7 million lines of code [12], and the F-35 Joint Strike Fighter is predicted to require as many as 6.5 times that number! [13] It is hardly practical to write such a tremendous volume of software without a means to test and debug it in as near the intended environment as possible.

Even in civilian aircraft or relatively simple weapon systems, embedded software will include logic related to the environment the vehicle will operate in and the vehicle configuration. Some means must be devised to test the software in all combinations of configuration and environmental conditions, and the most obvious answer is simulation.

5.4 Subsystem Integration and Test

The hardware-in-the-loop simulation often provides the first opportunity to perform an integrated test of newly designed hardware components. Without a HIL capability, isolated testing of individual components would be the only means to predict integrated performance.

Experience indicates that a large fraction of design and manufacture issues arise during the integration phase, where the various components of a vehicle or system are brought together for the first time. HIL simulation allows these problems to be encountered and solved earlier and more efficiently. Efficiency gains may be possible because only a few hardware components at a time are added to a functioning simulation, so problems are easily isolated.

5.5 Pilot Training

To the uninitiated, the term “simulator” often brings to mind the pilot training device so commonly used today. In fact, pilot training is an important use for simulation both in the airline industry and in military flight training. The most complex and realistic training simulator is a motion-based ‘full flight simulator’, which mechanically consists of an aircraft cab mounted on hydraulic struts, a visual representation of the world outside the aircraft windscreen, cockpit instrumentation, and flight controls. A typical motion-based full flight simulator is shown if Figure 4. The simulator shown is a training device for a Boeing 737-700 and is operated by FlightSafety Boeing Training International.

Figure 4. Example of a Full Flight Simulator – Exterior and Interior Views [14]

Though the two may seem worlds apart, a full flight simulator is closely related to a hardware-in-the-loop simulation. Only the hardware directly affecting the pilot-aircraft interaction is installed. The primary difference between the HIL simulation and a training device is in where emphasis is placed and where realism is important. In HIL simulation, considerations external to flight hardware behavior take secondary importance. In the case of a training device, it's less important to use strictly unmodified hardware than to faithfully represent vehicle behavior to the pilot.

"The FAA permits a great deal of credit and benefit to be derived from simulation." As such, the FAA has an interest in the quality, fidelity, and effectiveness of simulators. These training tools are intended not only to transfer training from aircraft to training aids, but also to transfer pilot behavior, that is reactions to situations, back into the cockpit. The only known reference for accomplishing this task is to insist on a high degree of realism, meaning a well-validated simulation of the aircraft [11].

Both the FAA and airlines prefer training with simulation devices to training in actual aircraft for a number of reasons:

1) More effective training can be accomplished in a simulator. Trainees observe immediate response to their actions in a simulator. In an aircraft, many tasks and events have to be thought and talked through, rather than actually carried out. System failures and emergency procedures cannot be fully replicated in an aircraft without risking the safety of the plane and crew [11].

2) Training accidents are eliminated. No major air carrier has had a training accident in decades simply because there are no training flights [11].
3) Operating costs of even the most complex simulators are below the operating costs of a medium to large airplane [11].

At major air carriers, simulation training is so extensive and so highly thought of that pilots are permitted to proceed directly to revenue operations after simulator training in a new aircraft type [11]. No further training or checking is required, only observation and supervision by a more experienced crewmember. Military pilot training takes this a step further. For fighter and attack military aircraft, two-seat trainer versions are no longer required. Pilots of F-22 and F-35 will make their first flights in the aircraft type as pilot-in-command. This extraordinary trend saves millions of development dollars by avoiding costly structural, aerodynamic, and cockpit systems design effort required to build two-seat pilot training versions. This well-earned confidence in simulation fidelity and its ability to properly instill necessary skills and reactions is remarkable.

Pilot and maintainer training are now considered critical elements of an aircraft development program. Part of the selection process that chose Lockheed Martin as prime contractor for the F-35 Joint Strike Fighter was presentation of a notional training system architecture. "Like the plane, the training system is scheduled to evolve during development." [15] Even after Lockheed Martin was awarded the contract, they continue to take training issues seriously. "Training systems are frequently one of the most ignored aspects of any major weapons

program, but for JSF it is a $750-million initiative during the development and demonstration phase alone." [15]

The visual and tactile human interfaces (view of outside world, control column, throttle, etc.) designed into pilot training simulators are critically important for their effective use. However, the engineering capabilities making the training tool possible are the aerodynamic and flight control models, equations of motion, mission control software, and other elements of the aircraft mathematical model being executed within the simulation. These may be the same simulation elements present in the all-software design simulation and the software and hardware-in-the-loop tools.

5.6 Marketing and Public Relations

In July of 1999, the Lockheed Martin F-22 was in the initial stages of a years-long flight test program. At the same time a few powerful individuals in the Congress were determined to kill the program entirely. Congressional sub-committees cut the $1.8 billion funding [12] for the first six production airplanes that were meant to complete the flight test program. Action on the floor of the House of Representatives eventually restored the funding. These types of persistent political challenges to large, expensive military programs point to the need for defense contractors to be well represented before the Congress, military leaders, and the executive branch of the U.S. government.

One way this was done on the F-22 program (in addition to traditional lobbying) was to develop a mobile simulation platform. This piloted simulator could be taken wherever

necessary to allow key decision-makers to gain a more personal knowledge of the system by giving them a chance to "fly" the aircraft. Congressmen and staffers were given a short mission assignment in the simulator and were shown how the mission could be accomplished with the F-22. They were then shown how much more difficult the same mission would be flying an F-15 (the aircraft to be replaced by the F-22). Because the F-15 is much more easily tracked by radar, and thus more susceptible to surface to air missiles and anti aircraft fire, some F-22 missions could be shown to be impossible with the F-15.

Also the prime contractor for three variants of the F-35 Joint Strike Fighter, Lockheed Martin is again employing simulation to help ensure the political future of the aircraft. An *Aviation Week & Space Technology* pilot recently flew a simulation of the F-35 short-takeoff-and-vertical-landing version at a Lockheed Martin facility near Washington, DC. The pilot was given the opportunity to carry out a simulated air to ground strike in the vicinity of surface to air missile sites. Later, an encounter with two enemy aircraft was simulated, resulting in the enemies being destroyed. Finally, a simulated landing was performed [16].

This experience led the pilot to write a very positive article about the F-35 in a magazine widely read within the military aerospace community. Lockheed Martin has found that simulation is not only a good investment from a technical standpoint, but is an excellent way to convey the utility of a proposed aircraft.

CHAPTER 6. THE SIMULATION COMPUTER PROGRAM

Simulation development should be one of the first engineering activities undertaken at the beginning of a new aircraft or missile program. Ideally, the simulation computer program should start as a generic simulation used throughout an engineering organization. Choosing an appropriate organizational structure for a simulation program and creating the communication structure among models and modules is the most sensitive element of simulation programming because of its implications on later efficiency and usability. If such a skeletal structure is readily available, it offers a tremendous advantage to the simulation team because they can immediately focus on customizing models to represent the new vehicle.

Trade studies must be readily supported by the structure of the program. For some parameters such as aerodynamic drag and control surface authority, this may be as simple as providing user scale factors and offsets. For other areas such as jet engine performance, the ability to easily change out performance databases must be supported. A generic simulation capability should incorporate these types of tools so that they are not reinvented for each new development program.

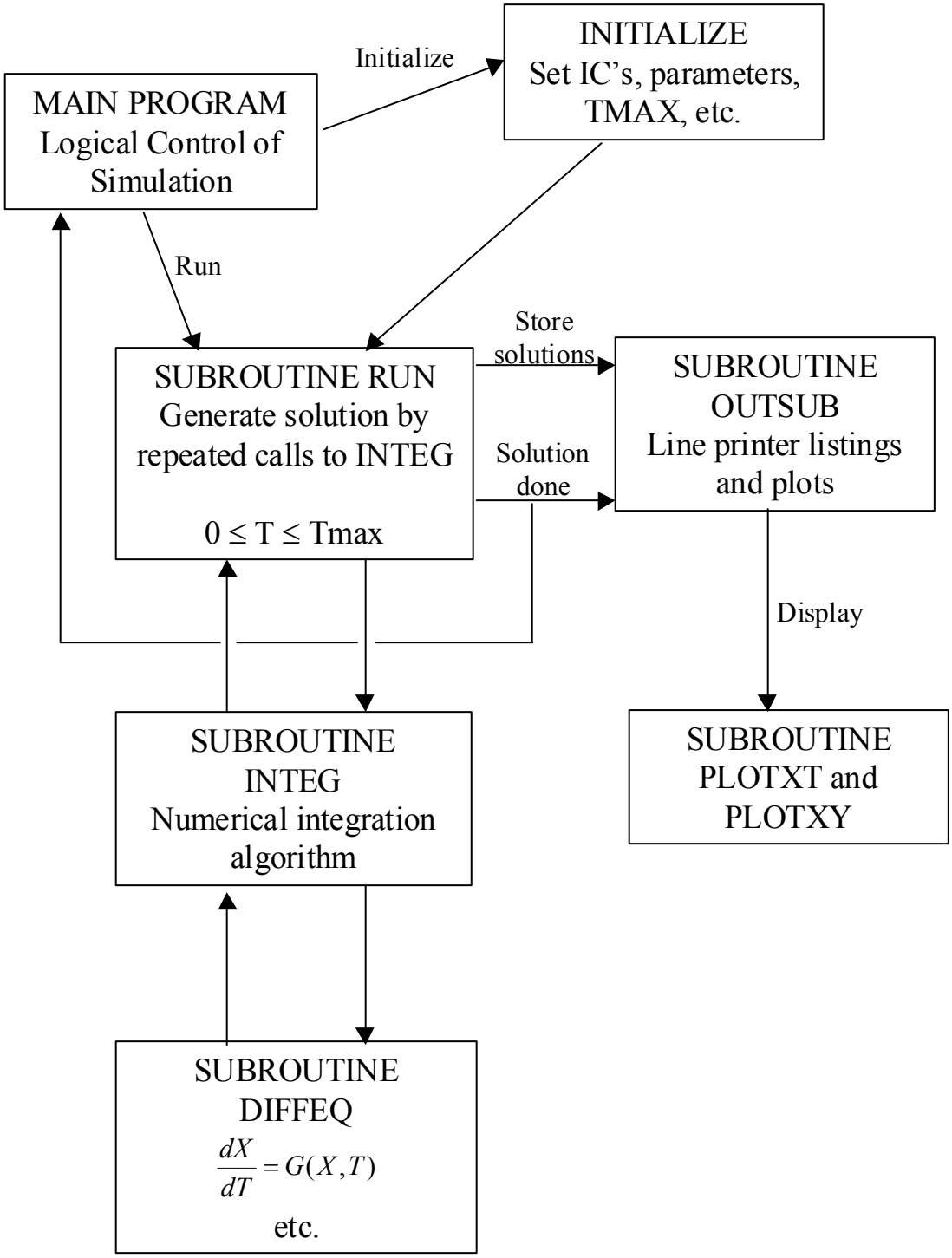
McHaney notes “It is important to know what information the simulation is required to provide at an early stage” [4] so that it can be designed accordingly. In the case of a development simulation for a flight vehicle, the simulation should be set up in such a way that any and all parameters within the simulation program can be output. This is because engineers involved in development of systems and subsystems may at any time require access

to the minute details of the workings of any particular software model or module. Output routine flexibility is one of the more difficult and essential components of a good development simulation.

6.1 Structure and Components

Many authors and engineers have suggested patterns for the organization of a simulation program. A high level structure suggested by Korn and Wait is shown in Figure 5. This structure represents a simulation with simple dynamics, most likely represented by a single differential equation. The main routine would collect user inputs to determine conditions for the run. The 'Initialize' routine would set initial conditions and initialize all simulation variables as necessary. Routine 'Run' would make repeated calls to 'Integ', which would solve the differential equation contained in the 'DiffEq' routine by numerically integrating the results. Outputs would be stored in 'Outsub', and the simulation would be terminated at the appropriate Tmax by 'Outsub'. The simple model framework is reasonable, but needs additional components in order to handle simulation of a more elaborate system.

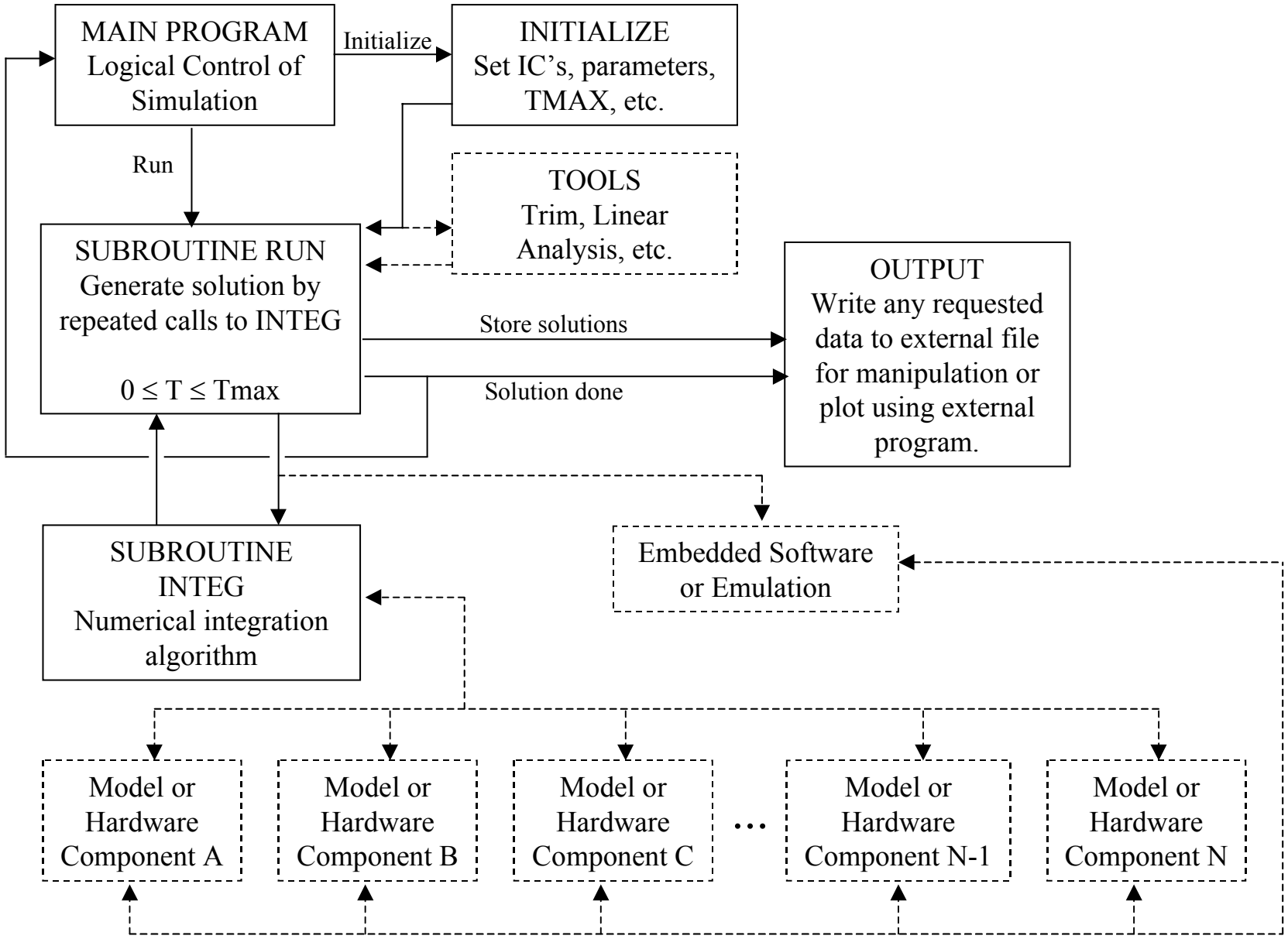
Figure 5. A Suggested Structure For A Simple Simulation Computer Program [8]



A modified version of Korn and Wait's structure is shown in Figure 6. Modifications are shown in dashed lines. This is a notional structure designed simply as a prop to discuss the various components required in a complex simulation. The differential equation in the simple structure of Figure 5 has been replaced by a number of models. Outputs from models that result in a force acting on the vehicle are collected in the 'Integ' routine. Here, the vehicle equations of motion are solved by numerical integration. Thus the 'Integ' routine is commonly referred to as the 'solver'.

The models emulate components and properties of the vehicle as well as the atmosphere in which it flies. In the case of a HIL simulation, some of these models may be replaced by actual hardware. In the more complex structure, the 'Run' routine communicates with the embedded software or emulation thereof. The embedded software in turn communicates with models of the vehicle components to control the behavior of the vehicle. For example, in a missile application, the embedded software manipulates control deflections to steer the weapon to its intended target. The results of the control deflection must also be carried through the aerodynamic model and ultimately through the equations of motion. In this way, the embedded software controls behavior of the model in the same way it will control hardware components in an actual flight. The structure of Figure 6 also contains a module of 'Tools'. These and other components of the more complex model are discussed in the sections below.

Figure 6. A Suggested Structure For A More Complex Simulation Computer Program



6.2 Models

Korn and Wait define simulation as "experimentation with models" [8]. This rightly implies that the models of the vehicle components and of the environment in which the vehicle operates form the heart of a simulation program. The only portions of the computer program that must differ between simulations of different vehicles are the embedded software and some models.

Construction of a simulation model involves writing computer code to accurately represent the system being modeled [4]. For highly complex subsystems, such as a jet engine, the external supplier of the component will often provide a model that can be used directly in the vehicle simulation. When this is not the case, the engineers involved in the development effort must create models of each component to be included in the simulation.

The following scenario demonstrates some examples of components to be modeled. When the pilot of an F-22 fighter jet pulls back on the control stick to command a climb, the flight control computer detects the control movement. The flight control computer then commands some combination of control effectors (shown in Figure 1) to move. The individual control actuators then respond to the command from the flight computer, resulting in control displacement. These displacements affect the air flowing around the aircraft in such a way that the aircraft is pitched up.

Though this entire process only takes a fraction of a second, an accurate simulation must continually consider each component involved. The sensors within the control stick must be modeled to account for any inaccuracies in detecting the movement as well as any time delay in relaying the movement to the flight computer. The flight computer must be modeled (or actual flight software included) to decide which control effectors should be moved and how much control is needed given the current flight condition and the magnitude of the pilot's input. This adds another time delay. Once the command is sent to the control actuators, the mechanical devices have some response characteristic that must be modeled. Finally, the control effectors are displaced and the aerodynamics cause the aircraft to respond. The aerodynamic model, mathematically describing the forces acting on the vehicle, is a major component of the simulation.

Creation of each of these models is a time-consuming and technically challenging task. Graybeal and Pooch suggest four stages of model development [5]:

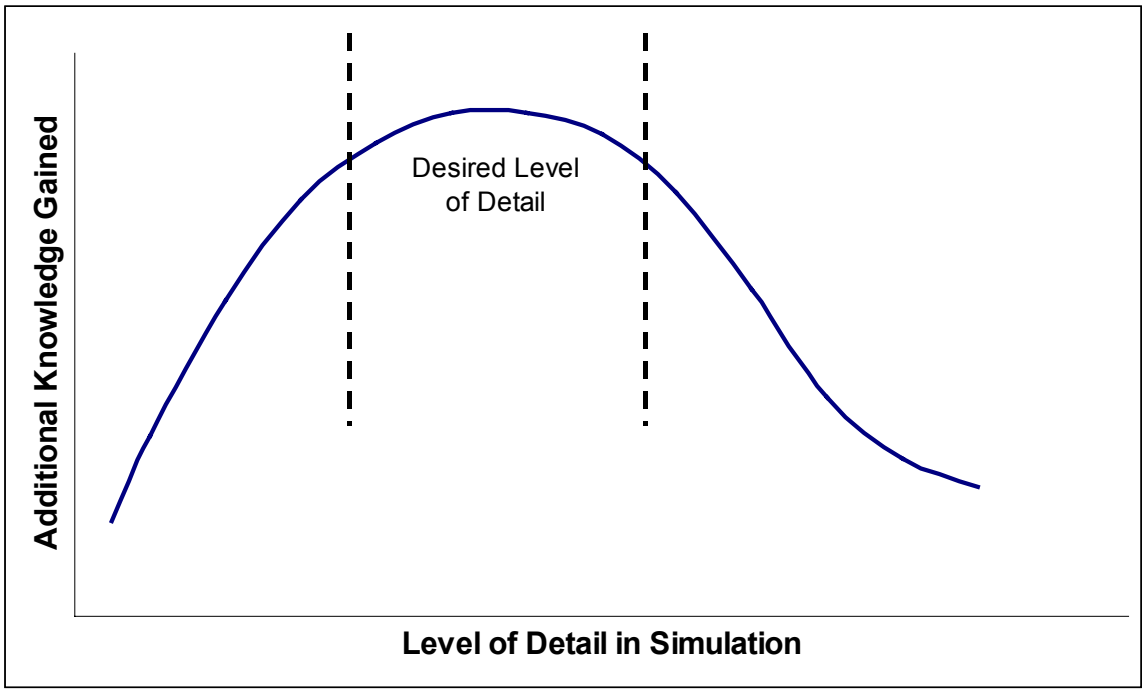
1) Observation of existing system. If the system to be modeled exists, its characteristics can be studied and tested. For systems that do not yet exist, this step is replaced by study of the component design, discussion with its designers, and observation of similar existing components.
2) Formulation of theories. Hypotheses are developed regarding the most correct way to model a component. Several approaches may be attempted in this stage and simple implementations of each taken to the next step.
3) Prediction of behavior. The candidate models are run with a variety of expected environments and stimuli.

4) Comparison of predicted versus actual. To validate the subsystem, the model results from step three are compared to observed behavior. This is challenging for a nonexistent system, in which case predicted behavior is compared to the designer's expectations and to behavior of similar existing systems.

Other authors have suggested similar model development processes [17,1]. Graybeal and Pooch note that these steps are simply an application of the scientific method [5] to model development.

Models cannot recreate all the aspects of the real system [2]. Attempts to add too much complexity will result in more difficult and time-consuming modeling efforts, but not necessarily a more useful model. Figure 7 illustrates the desired level of detail in a computer model of a system. Like in most engineering activities, additional effort eventually enjoys diminishing returns. This is a difficult balance to strike when building a model in the absence of a complete simulation in which to test the model. Therefore, model complexity often increases as overall simulation maturity advances.

Figure 7. Desired Level of Detail in Models [4]



Some particular types of models are discussed in more detail here.

6.2.1 Aerodynamics

Aerodynamic models for missiles and airplanes are derived almost solely from wind tunnel testing. For larger vehicles, the testing is done on sub-scale models either to reduce cost or because available wind tunnels cannot handle a full-scale model. Wind tunnel testing is an expensive endeavor, so efficiencies are sought whenever possible. Combinations of vehicle attitudes, configurations, and Mach numbers of interest are tested in a series of wind tunnel runs. For each run, aerodynamic forces acting on the vehicle are recorded. A database is constructed linking configuration and attitude data with force and moment information. This database becomes the basis for a vehicle aerodynamic model.

Throughout a simulation run, equations of motion (EOM) are integrated to determine attitude and flight speed. Configuration (such as landing gear position and control effector position) is user defined or determined by embedded software. These data are used to look up aerodynamic forces acting on the airframe. These forces are added to thrust and other forces and sent through the EOM, which are integrated to determine the vehicle attitude and speed for the next time step.

6.2.2 Flight Controls

Controls design is one of the major uses for simulation early in a development program. Designing control surfaces or effectors of the correct size and configuration is an iterative task. Available control authority must be sufficient to impart any desired vehicle responses, but control surfaces must also be aerodynamically efficient.

Controls design work must be done even before wind tunnel aerodynamic data are available. This is because wind tunnel tests need to be conducted with the correct vehicle configuration. An early simulation may use aerodynamic parameters estimated based on historical data and various estimation techniques. Several control techniques and configurations may be modeled and tested using this initial simulation. After the flight controls configuration is determined, wind tunnel tests are conducted. At that point, the aerodynamic effects of flight control deflections are modeled with simple table look-ups, similar to other aerodynamic forces.

Modeling of the mechanical portion of the control system is another matter entirely. A brief discussion is presented earlier in this section on how the control stick, flight computer, and control actuators on an airplane interact in response to a pilot's input. A similar chain of events occur in the case of a guided missile, except that the pilot and control stick are replaced by guidance logic.

6.2.3 Propulsion

One or more turbine engines generally supply propulsion for large airplanes. Long-range missiles may also use a jet engine, but shorter-range missiles often use rocket motors. In either case, the propulsion device is almost always supplied by an outside organization. The engine manufacturer normally supplies simulation models for jet engines, along with models of other components of the engine subsystem. Other items necessary in a high fidelity simulation of a jet engine are the fuel control system and engine control logic.

Rocket propulsion is far simpler to simulate. For a solid fuel rocket, the motor manufacturer may supply simple traces of thrust versus time since ignition for a few operating conditions. In this case, the simulation only needs to signal the rocket to ignite, then look up a thrust level based on time since ignition. For a liquid fuel, throttle controlled rocket motor, the manufacturer will likely provide a model for the fuel control system, and the problem is similar to the jet engine situation.

Thus the vehicle manufacturer is often largely relieved of the task of creating a model of the propulsion system. Nonetheless, vehicle development engineers must thoroughly understand

the model to properly integrate it into the overall simulation. This process may also provide insight into issues that may be encountered during actual engine integration.

6.2.4 Atmosphere and Winds

Modeling of air and wind properties is a crucial component of a flight vehicle simulation. Atmospheric models are typically based on variations about the "standard atmosphere" as defined by a central organization such as the FAA or a military customer. The same standard atmosphere characteristics are consistently used around the world. These characteristics are a reasonable set of air pressure, temperature, and density data for all altitudes of interest. A standard atmosphere table is not meant to describe actual conditions at any particular time or place, but rather provide a basis for comparing vehicle performance in a common environment.

Other sets of atmospheric conditions are considered, such as "hot day", "tropical day", "cold day", and "polar day". Properties of these atmospheric conditions are also available from a number of sources, including MIL-STD-210, a U.S. military standard concerned with atmospheric and wind modeling. Atmospheric properties are modeled by a simple table look-up based on vehicle altitude.

Winds are another important consideration of atmospheric modeling. One common wind model in use is the constant wind, where wind magnitude and direction are fixed at all altitudes. Many different altitude-varying wind models are defined in sources such as MIL-STD-210 and MIL-F-8785, a standard for certifying the stability and control characteristics of

military aircraft. Winds are generally modeled using equations or tabular data based on altitude and time.

Finally, atmospheric turbulence must be considered for its effect on airplane and missile stability and controllability. Turbulence is modeled as altitude and time-based random variation in wind direction and magnitude. Characteristics of turbulence and some guidance on modeling methods are presented in a number of military and civil specifications.

6.2.5 Terrain and Earth

Required fidelity of terrain and earth modeling depends a great deal on the type of vehicle and mission being simulated. For a transport aircraft, a flat earth model may be sufficient. The aircraft can takeoff and land using interactions between the landing gear model and a smooth ground surface. The simulation can be programmed to halt and declare a crash if the aircraft altitude falls below zero and the landing gear is not extended. In this scenario, the fact that the earth is somewhat spherical and is revolving about its axis has no bearing on the behavior of the airplane.

A slightly more complex modeling scheme may be adopted for airplanes or missiles that employ terrain following flight. In this scenario, the earth's surface may be modeled as arbitrarily smooth or rough, depending on the situation of interest. Topographical mapping data may also be used to simulate flight over a particular location on the earth. This type of model is also a "flat earth" model, meaning the earth's curvature and revolution is not considered.

The most complex type of earth model includes a detailed description of the earth's shape and rate of revolution. Actual terrain profiles are loaded for the regions in which flight is simulated. This type of model is a great deal more complex than the flat earth model and is used in whole mission simulators where inertial and Global Positioning System (GPS) navigation are included. With a highly accurate inertial navigation system, revolution of the earth is measurable and may be used to help calibrate or initialize the system on power-up. For modeling of GPS navigation, the characteristics of the orbits of GPS satellites becomes important. This must be modeled in concert with the revolving earth model.

6.2.6 Sensors

Flight vehicles use a multitude of sensors to determine conditions such as position, airspeed, Mach number, and angle of attack. For navigation purposes, a GPS receiver and related logic are likely to be in use. Mach number is almost always computed using data from pressure sensors mounted outside the vehicle. Angle of attack is useful for determining aerodynamic performance capability in flight and may be detected using a vane mounted on the vehicle.

Modeling of each sensor type will vary widely depending on the sensor's form and function, but must include potential sources of measurement error. Accurate modeling of sensor behavior is essential to correctly predict guidance and control software response and thus vehicle flight characteristics.

6.3 Tools

Another important element of a simulation computer program is the analysis tools it offers to the user. Various kinds of engineering analyses and studies must be readily supported by the structure of the simulation program. A number of analysis tools and capabilities surrounding the vehicle math model are required for guidance development, control law development, mission planning, and other engineering work. A generic simulation capability should incorporate these tools so that they are not reinvented each time a new vehicle is simulated.

One of the most common tools built into a simulation program is a trim routine. Given vehicle state information such as altitude, airspeed, and weight, a trim routine adjusts parameters such as angle of attack, throttle setting, and control deflections to find a state of static equilibrium. This capability allows a simulation run to begin at most any point within the vehicle flight envelope, rather than just at takeoff or launch. If one wishes to analyze an end-of-cruise maneuver or landing with fuel mostly depleted, it may be time consuming to simulate an entire mission up to the point of interest. Thus for airplanes and long range missiles, the capability to "trim" the vehicle enhances analytical efficiency.

Another important tool allows creation of a linear model of vehicle dynamics at a specific flight condition. A linear model, usually in state-space representation, allows for simulation of vehicle response to a control input or wind gust using only a few small matrices. Computer applications such as Matlab are particularly well suited to conducting linear model analysis. Some linear model analysis is used to conduct the solver study in Chapter VII.

Dynamics of a body can be fully described by a relatively small set of carefully chosen parameters called 'states'. At a given trim condition (altitude, airspeed, angle of attack, etc.), a linear model is created by independently inserting small changes into each state, then observing the effects on the other states. This information is collected for each state and arranged in a matrix known as a Jacobian. This process can be automated and quickly performed within the simulation program. For a small region around the trim condition, a linear model can predict time-history dynamic response to a series of control inputs. This capability is essential for flight control and autopilot development.

6.4 Equations of Motion

Equations of motion (EOM) are the differential equations that define the accelerations imparted to a body in response to forces acting on the body. Flight vehicle EOM can be constructed in any one of several frames of reference, as shown in Ridgely's paper on missile EOM [9]. EOM modeling in a single axis frame is fairly straightforward. A routine may be written to accept forces and moments acting on a body. By plugging the forces and moments, plus vehicle weight and inertia data into the EOM, resulting accelerations are computed. The solver routine (discussed in below) then numerically integrates the accelerations over time to predict velocity and position response to any series of control inputs.

The EOM for airplanes and missiles usually take slightly different forms, but the properties of the equations are similar. EOM for flight vehicles are generally non-linear, ordinary differential equations. That they are nonlinear implies no closed-form solution is available (thus the need for simulation). Some further discussion of EOM is given in Chapter VII.

6.5 Solver

Central to meaningful simulation is the "solver" or "integration method" used in the simulation environment. Inaccurate solutions to accurate force and moment predictions are not particularly useful. Therefore, the process of choosing an integration scheme is an important consideration in developing a simulation, whether it's a simple linear model based tool or a large-scale nonlinear model of a complex system. Chapter VII will explore several solver options and discuss the accuracy and practical value of each option in the context of aircraft and missile simulation.

CHAPTER 7. SOLVER STUDY

A study of several types of solvers is discussed in this chapter. The study is conducted using a simple simulation tool to objectively measure the performance of a number of solvers on an airplane linear model.

7.1 Solver Options

Beginning with Newton's Second Law of Motion, translational and rotational equations of motion (EOM) can be developed for any moving body. For aircraft and missiles, the EOM take the form of nonlinear, first order ordinary differential equations that can be written in the form $x' = f(x,t)$. As an example, see Ridgely's paper entitled "The Full Nonlinear Equations of Motion for a Rigid Missile" [9]. These equations can't be solved explicitly, so solutions are obtained via numerical integration methods.

A number of numerical integration techniques based on approximations of the Taylor expansion [18] are available to solve the initial value problem. The simplest is the Euler method, which predicts the next value based simply on the current value and the slope given a fixed time step: $x_{n+1} = x_n + f(x_n, t_n) \cdot \Delta t$. This first order method has the disadvantage of being relatively inaccurate and, depending on the equation to be solved, may become unstable as Δt grows. The error using Euler is of the order Δt [19]. An important advantage of the Euler method is that it requires only one execution of the EOM (x' is only computed once) per time step. The significance of this detail is discussed in a later section.

An improvement on Euler is the Predictor-Corrector method. An estimated slope and new value are computed just as in Euler, but a second step is taken whereby the estimated value is used to compute a corrected slope and new value:

$$x_{n+1} = x_n + \frac{f(x_n, t_n) + f(x_n + f(x_n, t_n) \cdot \Delta t,\ t_{n+1})}{2} \cdot \Delta t\ . \quad [19]$$

Predictor-Corrector is a second order method, essentially using the average of the current slope and the predicted slope at the next point to compute the next point. The error using this method is of the order $(\Delta t)^2$ [19]. This method requires two executions of EOM per time step.

Another family of solvers is known as the Runge-Kutta method. There are infinitely many Runge-Kutta solvers, from first order to n^{th} order. An n^{th} order Runge-Kutta solver also uses an approximation of a Taylor series expansion and requires at least n EOM evaluations per time step. The slope used to compute the next point in the solution is the weighted average of n slope calculations. The most common Runge-Kutta implementation is the fourth order solver, which has an error of the order Δt^4 [19].

The above solvers all use a fixed time step, which must be carefully selected by the user to provide the required accuracy. An additional action can be taken to allow selection of accuracy without performing a large number of unneeded EOM calls. Methods have been developed which use a variable time step such that a specified accuracy is delivered by modulating Δt. For example, one may run both fourth and fifth order Runge-Kutta at a given time step, compare the results, and require the difference to be smaller than a specified tolerance. If the difference exceeds the tolerance, the step size is decreased and the method is repeated. This is roughly the process used in the Matlab built-in solver ODE45. Matlab also

has a routine, ODE23, which uses second and third order Runge-Kutta. A practical comparison of the usefulness of these routines compared to Euler and Predictor-Corrector is presented in the next section.

7.2 Linear Model Study

A convenient way to compare various solver options is to build a linear model of a system, then run simulations of the system using Matlab. Linear model simulation is straightforward when compared to simulation of large-scale nonlinear models often used in the aerospace industry. Though relatively simple to implement, a linear model still provides dynamics and modal interactions representative of a more complex modeling system.

Consider a standard state-space model of an aircraft of the form:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

where x represents a column vector of the states, u represents a column vector of control inputs, and y represents a column vector containing outputs. If one desires to simulate vehicle response to an initial disturbance, then the control input vector can be set to zero. Also, if the desired outputs are simply the states themselves, then the system can be simplified to:

$$\dot{x} = Ax$$
$$y = x$$

so that only the states and the plant dynamics matrix must be known to run meaningful simulations of any vehicle about a trim point. Nelson [20] provides linear model data for several aircraft, one of which is the Navion general aviation airplane. The trim point about

which the linear model is valid is also given. Linear model simulations of Navion longitudinal dynamics about a particular trim point were run using the following state vector and plant dynamics from Nelson:

$$x = \begin{bmatrix} \Delta u \\ \Delta w \\ q \\ \Delta \theta \end{bmatrix}, \; A = \begin{bmatrix} -0.045 & -0.036 & 0.0 & -32.2 \\ -0.369 & -2.02 & 168.8 & 0.0 \\ 0.0019 & -0.0396 & -2.948 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}.$$

The states are: Δu, change in body axis forward velocity (ft/s), Δw, change in body axis vertical velocity (ft/s), q, body pitch rate (rad/s), and Δθ, change in body pitch attitude (rad). A description of the plant dynamics terms is available in Nelson.

Using this model, simulations were run using four solvers and two rates, as shown in Table 1. The simulated response is to a one ft/s mistrim at the start of the run. The trim speed for the linear model was 176 ft/s.

Table 1. Linear Model Simulation Run Identification

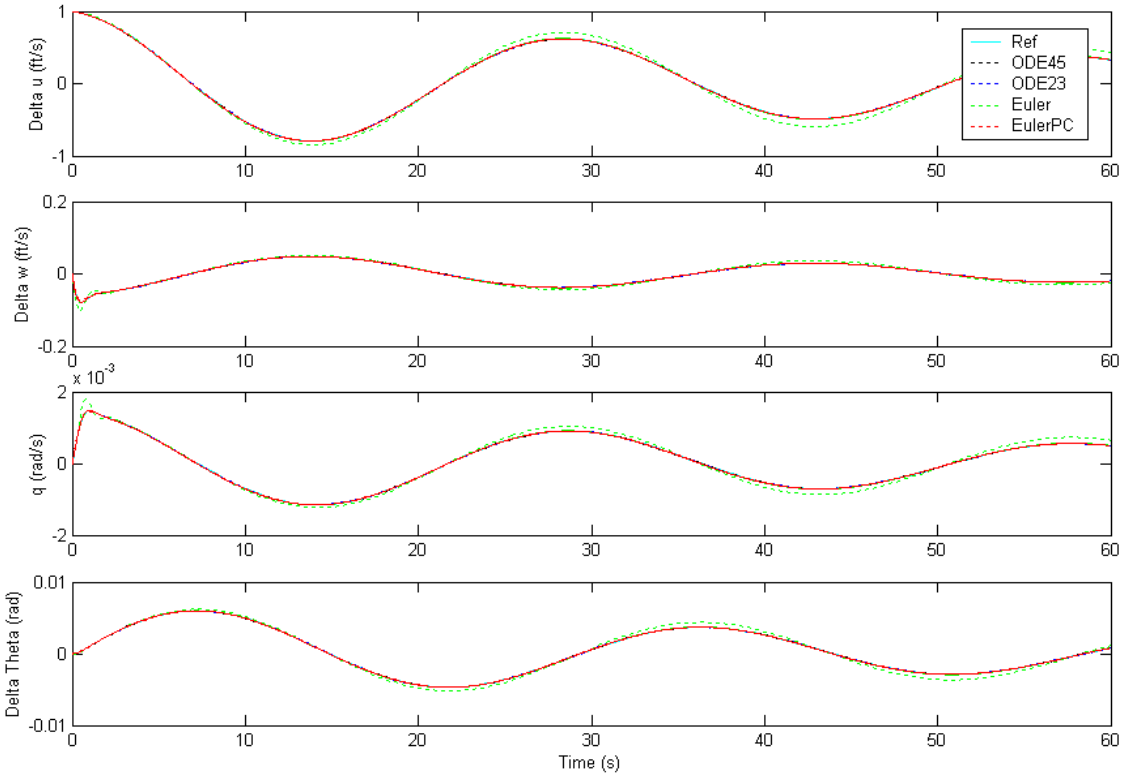
Run ID	Solver	Execution Rate
A1	ODE45	Variable
A2		100 Hz
B1	ODE23	Variable
B2		100 Hz
C1	Euler	~5 Hz
C2		100 Hz
D1	P-C	~5 Hz
D2		100 Hz

The 100 Hz rate shown in Table 1 was chosen because it is a typical rate chosen for fixed step size solvers used in large nonlinear simulations. The 5 Hz rate was chosen for Euler and Predictor-Corrector because it was the average rate used in the ODE45 variable step size run.

In addition to these eight runs, a reference run was made using ODE45 with the solution convergence tolerances set two orders of magnitude below the default values. The reference run is assumed to be sufficiently more accurate than the other runs that it can be taken as truth data. Error in the other runs is measured against the truth data. Several Matlab scripts and functions were written by the author to perform the simulations and compare results. These user-defined .m files are supplied for reference in Appendix A.

Plots of all eight solutions, plus the reference trajectory, are shown in Figure 8. Notice the primary response mode of the simulated aircraft is the phugoid mode, in which excess velocity is traded for excess altitude. This results in a lightly damped, low frequency oscillatory motion. This problem setup provides dynamics representative of a typical simulation case. The vast majority of the simulation time is spent in region where the rates of change are low. There's also a region of rapid change at the beginning of the run as the vehicle responds to the initial mistrim. For the variable step integrators, the problem provides opportunity for variation in Δt through the run. For the fixed step size integrators, integration error will vary substantially between the two regions.

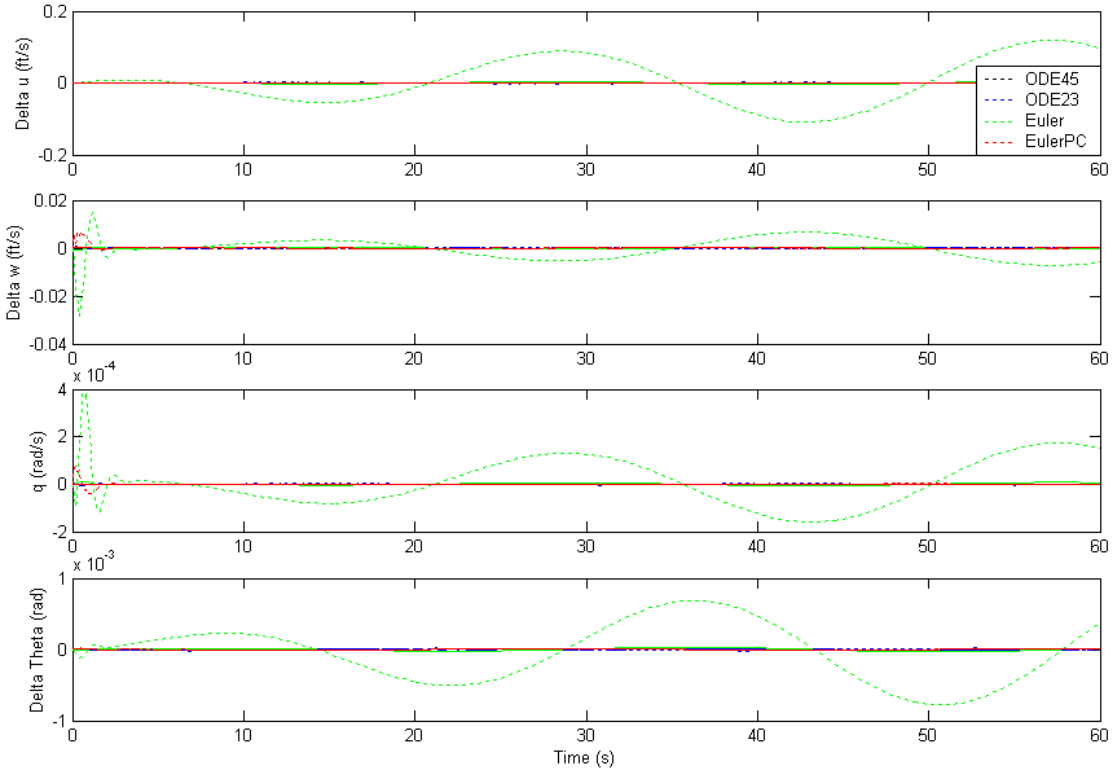
Figure 8. Simulated Response to Initial Airspeed Mistrim



The only trajectory easily discernible from the others is the low rate (5 Hz) Euler solution. Further analysis of the simulated solutions is required to assess and compare accuracy of the integration methods.

Because the independent variable array (Time) does not have consistent length among the several cases, results are not easily compared directly. A Matlab function (Interp.m) was written to interpolate dependent data along a new time vector given the original time vector. This function was used to interpolate all output data sets along the reference time vector. In this way, each data set could be compared directly to the reference output. Error in each state (reference – simulation) is plotted for each data set in Figure 9.

Figure 9. Comparison of State Error for Each Data Set



Again the error in the low rate Euler solution dominates the data. Little can be learned about the relative accuracy and efficiency of the other integration methods from the image because the errors differ by orders of magnitude. To quantify the "goodness" of each method, two terms were computed for each data set.

First, the "error" for each set was defined as the RSS of the error across all four states, providing a single number to compare among integration schemes. Second, the number of EOM executions required to perform each simulation was computed. For the runs using Matlab build in solvers ODE45 and ODE23, the 'statistics' option was turned on, returning several parameters including the number of EOM calls. Because Euler integration always

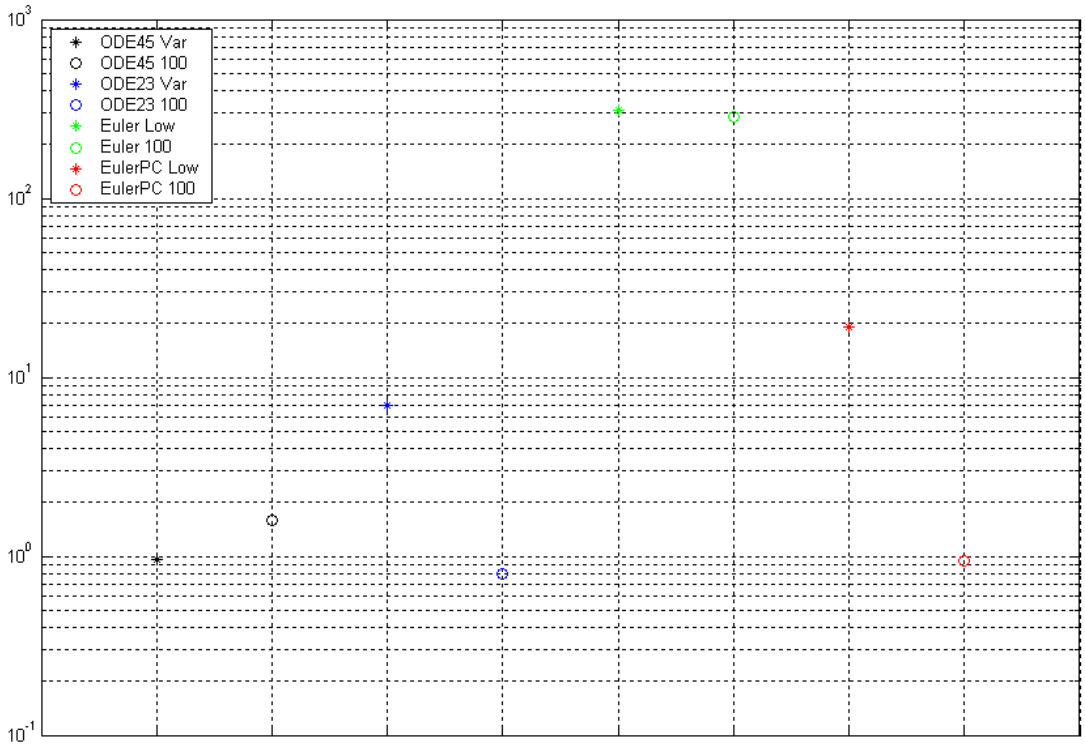
makes one EOM call per time step, the number of calls using Euler was simply computed as one fewer than the length of the time vector. For Predictor-Corrector this is doubled because P-C always makes two calls per time step. A comparison of the error and the number of EOM calls required for each data set is presented in Table 2.

Table 2. Comparison of Error and EOM Calls

Run ID	Solver	Execution Rate	RSS Error	EOM Calls
A1	ODE45	Variable	0.0019	505
A2		100 Hz	0.000044	36013
B1	ODE23	Variable	0.02	352
B2		100 Hz	0.000044	18010
C1	Euler	~5 Hz	1.0	300
C2		100 Hz	0.047	6000
D1	P-C	~5 Hz	0.032	600
D2		100 Hz	0.000079	12000

The minimum error by far is achieved using Matlab's solvers at 100 Hz. However, these two simulations also required the largest number of EOM evaluations. To combine these two measures of goodness into a single parameter, a figure of merit is created. The term is defined as the RSS of the error of each data set multiplied by the number of EOM calls needed to perform the simulation. This formulation produces the desired result: a relatively accurate integration scheme that requires relatively few EOM calls will receive a low figure of merit. A figure of merit of zero is "perfect". A comparison of figures of merit for each integration scheme is given in Figure 10.

Figure 10. Log Plot of Integration Figure of Merit



This comparison is reveals several notable characteristics. Predictor-Corrector at 100 Hz, ODE23 at 100 Hz, ODE45 at 100 Hz, and variable rate ODE45 all have about the same figure of merit, implying they are about equally fit for this type of simulation. It's likely, however, that improving error beyond some threshold is decreasingly valuable. Such a consideration could make these three methods differ widely in usability. If the error present in any of these is considered acceptable, then variable rate ODE45 becomes the clear winner due to the relatively small number of EOM calls.

The next best figure of merit corresponds to variable rate ODE23, roughly an order of magnitude worse than the top four methods. This method uses very few EOM calls, but has an error substantially larger than the top four methods. The next best scheme is Predictor-

Corrector at 5 Hz, with a figure of merit about five times that of variable rate ODE23. Both the error and the number of EOM calls are slightly higher than the comparable ODE23 scheme. Finally, both Euler schemes fared far worse than any other method. Euler integration has about the same figure of merit regardless of integration rate. This is reasonable since integration error with Euler is directly proportional to Δt. Euler at any rate is an order of magnitude worse than any other method tested and has a figure of merit 300 times that of the highest rated method!

This study shows that when choosing an integration scheme from the options presented, several different choices might be reasonable. The relative importance of simulation speed versus solution accuracy can dictate different selections. The author recommends using the data from Table 2 and devising a figure of merit that reflects the values applicable to any particular problem. One fact this study clearly establishes is that Euler integration, at any Δt, is a poor choice when other methods are viable. While the number of EOM calls can be kept to a minimum, errors are gross compared to other methods. The engineer's toolbox must still hold a place for Euler integration, as the next section will illustrate.

7.3 Large Scale Non-Linear Model Considerations

Linear model simulation is used extensively in the aircraft and missile industry for specific areas of autopilot and control design and tuning. Linear models at many points within the flight envelope are generally needed to ensure the controls design is suitable for all regions of flight. These linear models are often produced using the larger general-purpose simulation tool discussed throughout this paper.

In the large-scale, nonlinear simulation environment, the equations of motion remain unchanged from the relatively simple form described by Ridgely. The accelerations of a body must still be computed from the summation of forces. However, the mechanisms that must be executed each time the equations of motion are solved become tremendous. For example, to find the axial force acting on an aircraft, engine thrust must be computed. This is likely determined by a call to a dynamic engine model provided by the engine manufacturer, and will possibly involve communication with mechanical, electrical, and software components of the throttle assembly. Both the engine model and the throttle (whether hardware or simulated) will contain digital filters and feedback control systems to regulate rpm, temperatures, and pressures within various components of the engine.

Such models of complex components, and indeed the components themselves, are designed to function at a fixed execution rate agreed upon early in system development. Even within software elements of a simulation, digital filters will be running and flags will be set based on certain conditions being exceeded. As a result, it becomes very difficult to execute models at rates other than those they're designed to function at.

For these reasons, large-scale simulations in the aerospace industry consistently use fixed-rate Euler integration. The rate will be consistent with all execution rates being used within the components and processors contained within the system. Typical integration rates used in large-scale simulations are 100 to 1000 Hz. System models and interfaces use counters and flags to determine when to perform an execution of the model and when to simply return values from the previous execution.

With great effort, it might be possible to devise model interfaces that would allow multiple calls to EOM per time step and/or varying time steps, which would in turn allow more efficient integration schemes to be utilized. However, with models and components being delivered from many subcontractors and engineering organizations within a company, this task is normally deemed too burdensome. It's a task that would not only be tedious to accomplish, but would require maintenance each time a model was modified or updated.

7.4 Solver Conclusions

Using numerical integration methods such as the second-order Predictor-Corrector and the fourth-fifth order Runge-Kutta solver available in Matlab, great improvements can be made over Euler integration for use as a simulation solver. Use of these methods can provide efficiency gains from one to three orders of magnitude. Whenever practical, higher-order solvers should be used. Nonetheless, simple Euler integration remains the solver of choice for most complex simulation tools in the aerospace industry. This preference is due to the difficulties involved in preparing system and subsystem models capable of accommodating multiple calls per execution step, as would be required by other than a first order method.

CHAPTER 8. SUMMARY

"Neither the computer nor the model can replace human judgement and experience, which play a significant role in determining the validity and usefulness of simulation models for practical applications" [2]. Simulation is a powerful and essential tool for development and testing of flight vehicles. However, the value provided by simulation depends on the care with which the models, the simulation program, and the data inputs were created. The engineers and scientists who design and develop aircraft should view simulation results with skepticism until the tool has proven its value via comparison with real vehicle test data.

This paper has examined some of the essentials of flight vehicle simulation. Advantages and disadvantages of employing simulation have been addressed. Some of the important issues surrounding simulation, including airframe design work, embedded software development and test, performance prediction, and marketing have been discussed. Tenets of simulation programming and modeling have been presented. Finally, a study of solver effectiveness and efficiency has been conducted. The author hopes the paper can be a valuable aid to engineering management and to engineers having casual interaction with simulation and simulation data.

DEFINITION OF TERMS AND ACRONYMS

- **AC** – Advisory Circular, a document issued by the FAA to advise industry on procedures and methods of accomplishing certain tasks. AC's are recommended but not to required under regulation.
- **CFD** – Computational Fluid Dynamics, a method of predicting fluid flow over a body using approximate solutions to the Navier-Stokes equations.
- **Digital Simulation** – Simulation designed to run on a digital computer. Though the system being modeled may be continuous, it is simulated, usually at a high rate, in a digital environment.
- **EOM** – Equations of Motion, the mathematical relationships governing motion of a body. For aircraft, these are usually nonlinear, first order differential equations.
- **FAA** – Federal Aviation Administration
- **GPS** – Global Positioning System. A navigation system comprised of a network of satellites transmitting data to earth. A receiver can use these data to determine position, velocity, etc. via triangulation.
- **HIL** – Hardware-in-the-loop simulation
- **Model** – A simplified representation of a system [2]. In this paper, generally digital computer modeling.
- **Simulation** – The use of models to develop conclusions that provide insight on the behavior of any real world elements [4]. In this paper, the term generally refers to using models to predict aircraft and missile behavior through digital computer simulation.
- **6DOF** – Six Degrees Of Freedom

- **Six-degrees-of-freedom** – Describes the possible motions allowed by a particular simulation. A six-degree-of-freedom simulation considers all possible motions, which are forward, lateral, and vertical velocity plus pitch, roll, and yaw.
- **Solver** – A numerical method of finding solutions to a differential equation, also known as 'integration method'. Many types of solvers are available and in use.
- **Time-marching** – Simulating in a time-dependent fashion. Beginning at a defined start time and moving forward at a fixed or variable rate to a pre-defined or computed stop time.
- **Trim** – A state of static equilibrium, or an algorithm designed to find such a state.
- **TVC** – Thrust Vector Control. A method of creating control moments by deflection engine or rocket motor thrust through a controlled angle.

REFERENCES

1. N. Roberts, et al. *Introduction to Computer Simulation – A System Dynamics Modeling Approach*. Addison-Wesley, 1983. Pages 4, 8, 243.
2. F. Neelamkavil. *Computer Simulation and Modeling*. John Wiley and Sons, 1987. Pages 6-12, 30, 161.
3. W.R. Franta. *The Process View of Simulation*. Elsevier North-Holland, 1977.
4. R. McHaney. *Computer Simulation – A Practical Perspective*. Academic Press, 1991. Pages 2-6, 38, 41-45, 77.
5. W. Graybeal and U. Pooch. *Simulation: Principles and Methods*. Winthrop Publishers, 1980. Preface, Pages 2, 5, 9, 10.
6. J.L. Harpell, M.S. Land, and A.H. Mansour. "Operations Research in Practice: A Longitudinal Study." *Interfaces 19(3)*, May-June 1989, Pages 65-74.
7. G. Adkins and U. Pooch. "Computer Simulation: A Tutorial." *Computer 10, no. 4*, April 1977, Pages 12-17.
8. G. Korn and J. Wait. *Digital Continuous-System Simulation*. Prentice-Hall, 1978. Preface, Pages 20, 119.
9. D.B. Ridgely. "The Full Nonlinear Equations of Motion for a Rigid Missile." *Technical Report, Raytheon Systems Company, Tucson, AZ*, March 2000.
10. "Airplane Simulator Qualification." *Federal Aviation Administration Advisory Circular 120-40B*. 29 July 1991.
11. E.M. Boothe. "Simulation Realism, Cost and Benefits." *ITEC 92*, 8 Apr 1992.
12. *Venik's Web*. "F-22 Raptor Systems." <http://www.aeronautics.ru/nws002/f22/systems.htm>, February 2002.

13. “F-35 Faces Aggressive Flight Test Program.” *Aviation Week & Space Technology*, Aug 19, 2002, Page 56.

14. *FlightSafetyBoeing.* “Simulator Virtual Tours.” <http://www.fsbti.comsimulators/virtual_tours/>, 2 May 2002.

15. “JSF Set to Redefine Training Philosophy.” *Aviation Week & Space Technology*, Aug 19, 2002, Page 53.

16. “F-35 Cockpit Targets Information Integration.” *Aviation Week & Space Technology*, Aug 19, 2002, Page 52.

17. A.G. Mihram. “The Modeling Process.” *IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-2, No. 5*, November 1972, Pages 621-629.

18. D.M. Etter. *Engineering Problem Solving with Matlab*. Prentice Hall, 1993.

19. W.E. Boyce and R.C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. John Wiley and Sons, 1977.

20. R.C. Nelson. *Flight Stability and Automatic Control*. McGraw-Hill, 1989.

APPENDIX A. MATLAB CODE WRITTEN FOR SOLVER STUDY

Test.m

```
% Script to run linear model simulations of Navion general aviation
% airplane.  This model uses data from:
%   Nelson, R.C., Flight Stability and Automatic Control,
%   McGraw-Hill, New York, 1989
% The simulation will be repeated using several solver methods
% to compare suitability of each given several usage scenarios.
%
% The "trim point" about which the simulation is run is described as
% follows:
%   Day Type = Standard Day, Altitude = Sea Level, Velocity = 176 ft/s,
%   A/C Weight = 2750 lb, cg = 29.5% MAC,
%   Ix = 1048, Iy = 3000, Iz = 3530, Ixz = 0 slug-ft-ft
% Aircraft reference geometry is as follows:
%   Wing Area = 184 ft-ft
%   Wing Span = 33.4 ft
%   Wing MAC  = 5.7 ft

% For this model, the states are defined in the body axis as the
% delta from trim of
%   1) x-velocity (u),
%   2) z-velocity (w),
%   3) pitch rate (q),
%   4) pitch attitude (theta)

% Define initial conditions with a disturbance in u
x0=[1 0 0 0];

% Define options for use in calls to build-in solvers.  Initially
% set convergence tolerances to two orders of magnitude below default
% to complete a highly accurate "truth" simulation.  Use Matlab's
% medium order Runge-Kutta solver with variable step size.
disp(' ')
disp('ODE45 Reference')
Opts1=odeset('AbsTol',1e-8,'RelTol',1e-5);
[T0,X0]=ode45('navion',[0 60],x0,Opts1);

%
% Evaluate ODE45

% Use Matlab's medium order Runge-Kutta solver with variable step size
% and default accuracy.
disp(' ')
disp('ODE45 Var')
Opts2=odeset('stats','on'); % Reset options to normal with stats output
[T1,X1]=ode45('navion',[0 60],x0,Opts2);
% Use Matlab's medium order Runge-Kutta solver at 100 Hz
disp(' ')
disp('ODE45 100')
T=[0:.01:60]';
```

```
Opts3=odeset('stats','on','Maxstep',0.01); % Force integ at >= 100 Hz
[T2,X2]=ode45('navion',T,x0,Opts3);

%
% Evaluate ODE23

% Use Matlab's low order Runge-Kutta solver with variable step size
disp(' ')
disp('ODE23 Var')
[T3,X3]=ode23('navion',[0 60],x0,Opts2);
% Use Matlab's low order Runge-Kutta solver at 100 Hz
disp(' ')
disp('ODE23 100')
[T4,X4]=ode23('navion',T,x0,Opts3);

%
% Evaluate Euler

% Call the user-defined Euler method at the average step size used
% by the nominal ODE45
disp(' ')
disp('Euler Low')
dT1=mean(diff(T1));
[T5,X5]=euler('navion',[0 60],x0,dT1);
% Call the user-defined Euler method at 100 Hz
disp(' ')
disp('Euler High')
[T6,X6]=euler('navion',[0 60],x0,.01);

%
% Evaluate Euler Predictor-Corrector

% Call the user-defined P-C method at the average step size used
% by the nominal ODE45
disp(' ')
disp('EulerPC Low')
[T7,X7]=eulerpc('navion',[0 60],x0,dT1);
% Call the user-defined P-C method at 100 Hz
disp(' ')
disp('EulerPC High')
[T8,X8]=eulerpc('navion',[0 60],x0,.01);

% Plot results for comparison
figure(1)
plotsim(T0,X0,0,'c')
plotsim(T1,X1,1,'k:')
plotsim(T3,X3,1,'b:')
plotsim(T5,X5,1,'g:')
plotsim(T7,X7,1,'r:')
plotsim(T2,X2,1,'k')
plotsim(T4,X4,1,'b')
plotsim(T6,X6,1,'g')
plotsim(T8,X8,1,'r')
% Add a legend
subplot(4,1,1)
legend('Ref','ODE45','ODE23','Euler','EulerPC',0)
```

```
% Interpolate all output along the 100 Hz time vector T0 and
% compute the error from the reference array X0
X1_Err=Interp(X1,T1,T0)-X0;
X2_Err=Interp(X2,T2,T0)-X0;
X3_Err=Interp(X3,T3,T0)-X0;
X4_Err=Interp(X4,T4,T0)-X0;
X5_Err=Interp(X5,T5,T0)-X0;
X6_Err=Interp(X6,T6,T0)-X0;
X7_Err=Interp(X7,T7,T0)-X0;
X8_Err=Interp(X8,T8,T0)-X0;

% Plot error arrays for comparison
figure(2)
plotsim(T0,X1_Err,0,'k:')
plotsim(T0,X3_Err,1,'b:')
plotsim(T0,X5_Err,1,'g:')
plotsim(T0,X7_Err,1,'r:')
plotsim(T0,X2_Err,1,'k')
plotsim(T0,X4_Err,1,'b')
plotsim(T0,X6_Err,1,'g')
plotsim(T0,X8_Err,1,'r')
% Add a legend
subplot(4,1,1)
legend('ODE45','ODE23','Euler','EulerPC',0)

% Define an error array
Errs=[norm(X1_Err)
      norm(X2_Err)
      norm(X3_Err)
      norm(X4_Err)
      norm(X5_Err)
      norm(X6_Err)
      norm(X7_Err)
      norm(X8_Err)];

% Compute the number of calls to the function required
Calls=[505               % From printed statistics
       36013             %  "
       352               %  "
       18010             %  "
       length(T5)-1      % Euler uses one call per step
       length(T6)-1      %  "
       2*(length(T7)-1)  % Euler PC uses two calls per step
       2*(length(T8)-1)];%  "

% Compute a measure of efficiency as Err*Calls
ExC=Errs.*Calls

% Plot the normalized error for each run
figure(3)
semilogy(1,ExC(1),'k*'); hold on;
semilogy(2,ExC(2),'ko')
semilogy(3,ExC(3),'b*')
semilogy(4,ExC(4),'bo')
semilogy(5,ExC(5),'g*')
semilogy(6,ExC(6),'go')
```

```
semilogy(7,ExC(7),'r*')
semilogy(8,ExC(8),'ro'); hold off
xlim([0 9])
set(gca,'xticklabel',' ')

Legend ('ODE45 Var  ', ...
        'ODE45 100  ', ...
        'ODE23 Var  ', ...
        'ODE23 100  ', ...
        'Euler Low  ', ...
        'Euler 100  ', ...
        'EulerPC Low', ...
        'EulerPC 100',2);
```

Navion.m

```
function xdot = navion(t,x)
% NAVION(t,x) returns xdot, defined in terms of the state-space
% representation xdot=Ax+Bu.  For this case, the input, u, is
% always [0].  All dynamics will result from initial conditions.

A=[-.045    .036    0       -32.2
   -.369    -2.02   168.8   0
   .0019    -.0396  -2.948  0
    0       0       1       0];

xdot=A*x;
```

Euler.m

```
function [t,y]=euler(f,T,y0,dt)

% function [t,y]=EULER('f',[t0 tmax],y0,dt) uses the Euler finite
% difference equation to numerically solve an ODE.  f is the ODE
% for which a solution is desired.  The form yprime = f(t,y) is
% assumed, where t is independent (possibly time) and y is dependent.
% y0 is the initial conditions.

% Compute the number of iterations necessary
t(1)=T(1);
tmax=T(2);
nmax = (tmax-t(1))/dt+1;

% Store the initial conditions
y=y0;

for n=1:nmax-1
    f1 = feval(f,t(n),y(n,:)'); % Compute the current value of the
function
    y(n+1,:) = y(n,:) + dt*f1'; % Compute the next y
    t(n+1) = t(n)+dt;           % Increment t
end
t=t'; %Output a row vector
```

EulerPC.m

```
function [t,y]=euler(f,T,y0,dt)

% function [t,y]=EULERPC('f',[t0 tmax],y0,dt) uses the Euler
% Predictor-Corrector finite difference equation to numerically
% solve an ODE.  f is the ODE for which a solution is desired.
% The form yprime = f(t,y) is assumed, where t is independent
% (possibly time) and y is dependent. y0 is the initial conditions.

% Compute the number of iterations necessary
t(1)=T(1);
tmax=T(2);
nmax = (tmax-t(1))/dt+1;

% Store the initial conditions
y=y0;

for n=1:nmax-1
    f1 = feval(f,t(n),y(n,:)'); % Compute the current value of the
slope

    yp = y(n,:) + dt*f1';       % Compute the predicted next y
    t(n+1) = t(n)+dt;           % Increment t

    f2 = feval(f,t(n+1),yp');   % Compute the next value of the slope

    y(n+1,:) = y(n,:) + dt/2*(f1'+f2'); % Compute the corrected next y
end
t=t'; %Output a row vector
```

Plotsim.m

```
function plotsim(T,X,mode,color)
% PLOTSIM plots results of the Navion simulation using passed options
% to determine PLOT arguments and formats

if nargin<=2
    mode=0;
end
if nargin<=3
    color='k';
end

if mode==0
    subplot(4,1,1)
        plot(T,X(:,1),color)
        ylabel('Delta u (ft/s)')
        xlim([0 60])
    subplot(4,1,2)
        plot(T,X(:,2),color)
        ylabel('Delta w (ft/s)')
        xlim([0 60])
    subplot(4,1,3)
        plot(T,X(:,3),color)
        ylabel('q (rad/s)')
        xlim([0 60])
    subplot(4,1,4)
        plot(T,X(:,4),color)
        ylabel('Delta Theta (rad)')
        xlabel('Time (s)')
        xlim([0 60])
 else
    subplot(4,1,1)
        hold on
        plot(T,X(:,1),color)
        hold off
    subplot(4,1,2)
        hold on
        plot(T,X(:,2),color)
        hold off
    subplot(4,1,3)
        hold on
        plot(T,X(:,3),color)
        hold off
    subplot(4,1,4)
        hold on
        plot(T,X(:,4),color)
        hold off
end
```

Interp.m

```
function C=Interp(A,TA,TB)
% INTERP(A,TA,TB) Interpolates dependent A with independent TA
% over the new independent TB.

j=length(TB);
for i=1:j-1
    [v,n]=min(abs(TA-TB(i)));  %Find the nearest value in TA to TB(i)
    if TA(n)>TB(i), n=n-1; end %Ensure TB(i) is between TA(n) and
TA(n+1)
    C(i,:)=A(n,:)+(A(n+1,:)-A(n,:))/ ...
                (TA(n+1)-TA(n))*(TB(i)-TA(n)); %Interp A
end
k=length(A);
C(j,:)=A(k,:);                  %Set the final term
```