N90-23068

# Six-Degree-of-Freedom Missile Simulation Using the ADI AD 100 Digital Computer and ADSIM Simulation Language

Koos Zwaanenburg
Applied Dynamics International
Ann Arbor, Michigan

## Abstract

This paper illustrates the use of an AD 100 computer and the ADSIM language in the six-degree-of-freedom digital simulation of an air-to-ground missile. The missile is launched from a moving platform, typically a helicopter and is capable of striking a mobile target up to 10 kilometers away. The missile could be any tactical missile.

## Introduction

Real-time simulation with hardware in the loop is used extensively in the missile development business. It is used primarily to validate designs prior to the actual flight, to avoid experiments with actual flight hardware, and to reduce costs of flight trials by simulating a large variety of scenarios and conditions instead of performing these tests in real life. Missile manufacturing companies are an important section of Applied Dynamics' customer base. Therefore, Applied Dynamics International(ADI) developed this particular missile model so that it can function as a realistic example of the type of models that most of our customers work with.

## Six-Degree-of-Freedom Missile Model

The missile is controlled by four fins mounted in cruciform configuration at the rear of the missile. The fins are independently controlled by pneumatic servos. The servos are activated by commands from the autopilot, which processes the sensor and seeker guidance outputs before issuing these commands. The missile roll, pitch, and yaw attitudes are sensed using two-degree-of-freedom gyros. Two such gyros are required. The target is tracked using an inertially stabilized seeker (laser or RF/IR) mounted at the front end of the missile inside a radome. Imperfect attitude sensing and target tracking are included in the simulation. The user can exclude, include or amplify these extraneous effects by selecting the proper options switches.

The missile is thrusting during part of the flight. Thrust is a prespecified function of time. Burnout occurs about five seconds from launch time. As the missile burns the propellant, the rotational inertias of the vehicle change and the center of gravity moves forward along the missile longitudinal axis. The missile aerodynamic center remains aft of the missile center of gravity at all times. Imperfect c.g. location, thrust offset, and misalignment are also modeled in the simulation. Atmospheric wind velocity is modeled as the sum of a steady state and a gusting component. The gusting component is assumed to be a zero-mean, uniformly distributed random variable in all directions. Vehicle translational equations of motion are expressed in earth axes, while the missile body axes are used for the

rotational dynamic equations.

Aerodynamic coefficients are described as multivariable functions of typically the missile angle of attack, angle of sideslip, control surface displacements and Mach number. Furthermore, rotor downwash (the effects of which are confined to the neighborhood of the helicopter), instantaneous thrust, gyro drift angles, and seeker tracking errors are described as nonlinear functions.

The simulation is terminated when one of the following conditions occurs: when the simulation end time exceeds the user-specified end time, when the missile crashes, when the missile Mach number exceeds a specified limit, or when the relative position component along the relative velocity vector $x_{rv}$ changes sign. When $x_{rv}$ changes sign, the miss distance is computed by linearly interpolating the separation between the missile and the target to the point where $x_{rv}$ is zero.

The complete system of equations has been partitioned into different modules for simplicity. These modules can be looked upon as various "generic" subsystems of the missile model. They can be used separately for other missile simulations.

The missile coordinate conversion module (MCC) describes the various inertial and body axes coordinate transformations required for every tactical missile model, the missile attitude, quaternions, direction cosines, the effect of steady and gusting wind components, the rotor downwash, the velocity components, and angle of attack and sideslip.

The line-of-sight module (LOS) describes the missile-target geometry and velocities in absolute and relative terms and describes the target tracking errors, radome aberrations, and glint.

The seeker module describes the tracking errors in azimuth and elevation and the seeker dynamics. Furthermore, it describes the guidance commands for the autopilot module. This module allows insertion of specifics for laser, RF, or IR seekers.

The autopilot module describes an analog autopilot that accepts guidance commands from the seeker module and body attitude information from the MCC module to implement a proportional navigation scheme that will guide the missile to the target. The autopilot module features separate autopilot channels for roll, pitch, and yaw. Gyro drift is modeled as well. The autopilot module generates four fin position commands for the four actuators. The actuator module will be used four times for this particular missile simulation. The actuators are described as nonlinear fourth-order systems with nonlinearities like running and breakaway friction, torque limiting, and fin angular traveling limiting.

The aerodynamics module describes all aerodynamic force and moment coefficients acting on the missile airframe as multivariable functions. Inputs to the functions are typically the missile angle of attack, angle of sideslip, control surface displacements and Mach number. This module is dependant on the types of wind-tunnel measurements performed on the missile; therefore, this module is the least "generic" of all missile modules. For this particular missile there are 14 functions of one variable, 17 functions of two variables and 9 functions of three variables that describe the aerodynamics. Furthermore, some functions are included to represent thrust and mass variations due to propellant burn.

The missile equations of motion module combines the aerodynamic coefficients with air density, dynamic pressure, and gravity to form the total forces and moments acting on the airframe. Dividing these forces and moments by mass or moments of inertia provides the missile translational and rotational accelerations.

## SYSTEM 100 Architecture

Applied Dynamics International has been involved in solving time-critical simulation of continuous dynamic systems since its founding in 1957. The SYSTEM 100 simulation computer system was introduced by ADI in 1984. It consists of an AD 100, a high-speed, floating-point compute engine; a host controller, a general-purpose digital computer of the VAX family; and ADSIM, a user-friendly simulation language designed specifically for the AD 100. The SYSTEM 100 hardware and software work together to form a complete simulation environment.

### The AD 100

The AD 100, a synchronous, bus-oriented multiprocessor compute engine designed for time-critical digital simulation, is the basic fundamental building block of the SYSTEM 100. The AD 100 is a single-user system without an operating system. It is controlled by a multiprocessing VAX host computer running the VMS operating system. Acting as the controller and user interface, the host computer relieves the AD 100 of these interrupt-based tasks. The compute engine needs to be isolated from the overheads and restrictions associated with an operating system in order to achieve and maintain its optimum computation speed or frame rate.

The AD 100 is capable of 20 million floating-point operations per second. The basic system consists of four processors and a host interface tied to a common bus, the PLUSBUS. The PLUSBUS is 105 bits wide, 65 bits of data and 40 bits of address and control. Emitter-Coupled Logic (ECL) devices are used to obtain high computational speed. The four basic processors are the Communication and Control Processor (COM), the Arithmetic Logic Unit (ALU), the Multiplier (MUL), and the Storage Processor (STO). Each processor has its own program memory, program counter, and instruction decoder. Each processor has a 64-bit instruction. The COM Processor has a 64K program memory, and the other processors have 4K program memories. Timings in the AD 100 are expressed in terms of a master clock period of 25 nanoseconds. The instruction cycle of the AD 100 consists of four of these phases or periods.

Every arithmetic operation performed on the AD 100 is done in floating-point arithmetic. Calculations are performed using either a 53-bit short-word format or a 65-bit long-word format. Both formats contain 1 sign bit, 12 exponent bits, and either 40 or 53 significand bits. The long-word format is used where additional accuracy is needed, such as in the case of numerical integration to minimize roundoff and truncation errors.

The STO Processor provides 64K of 65-bit high-speed data storage. Memory accesses and address arithmetic can take place two per instruction cycle. Some simulation tasks such as function generation and memory buffers require large amounts of data storage. It is for these purposes that an optional processor, the Function Memory Unit (FMU), was introduced, which has data memory of 2 million words by 64 bits.

### ADSIM Environment

With a specific application area in mind, namely real-time simulation, ADI was able to design the AD 100's hardware and ADSIM to handle the necessities of the simulation environment. ADSIM is made up of a high-level simulation language and a run-time interactive control environment.

The ADSIM language is mathematically oriented. Many key elements of a typical simulation are built into the language, such as integration techniques, function generation, and

control-system nonlinearity functions. A control executive consisting of two programs, one that runs on the host controller and one that runs on the AD 100, provides the basis for implementing a model. The control structure built into this executive controls such parameters as system time (simulation time), frame time, and integration step size. A dynamic section is provided for the model's differential equations. ADSIM allows the model to be implemented as a series of scalar and first-order differential equations. If conditional code is included as part of the model, the control executive takes care of padding the frame such that each frame is consistent with real time. Sorting of the dynamic equations and identifying algebraic loops are examples of some of the capabilities of the ADSIM compiler. Integration is handled using Runge-Kutta, Adams-Bashforth, or Adams-Moulton system-level routines. The model development time is much less when a simulation language such as ADSIM is used since many of these standard simulation techniques are built into the language.

ADSIM program development, including editing, compiling, and debugging, is performed on the host computer. There are two ADSIM compilers: one that produces code to be executed on the AD 100 for time-critical work and one that produces code to be executed on the host computer for non-time-critical experimentation and debugging. The same ADSIM source can be processed by either compiler.

Running a program on the AD 100 involves loading the executable code from the host computer into the AD 100 at run time. The user run-time interface consists of a program called INTERACT running on the host computer. INTERACT provides a user-friendly interface for debugging and experimentation, allowing constants to be changed, variables to be displayed, integration methods changed, breakpoints to be set, etc. This environment reduces the time it takes to get the simulation into a state where it can be integrated into the design and testing phase.

## FORTRAN

The AD 100 is also able to run FORTRAN programs. ADI developed FORTRAN/AD, a subset of the FORTRAN 77 standard, to run on the AD 100 computer. In general, FORTRAN programs will not be as computationally efficient as ADSIM programs, but they allow the user to benefit from investments already made in FORTRAN simulations.

## Implementation in ADSIM

The interactive nature of ADSIM, together with the INTERACT utility, make the task of implementation, verification, and validation easier and allows one to develop a feel for the system being simulated. A rich set of INTERACT commands allows the user to change any simulation variable or to change the integration time step and method; an on-line data logger and graphics package allow the capability to verify simulation results. Various researchers have estimated that the verification and validation portion of a simulation can consume 30 to 60 percent of a particular project's schedule and budget. The implementation of the missile model in ADSIM, together with its interactive environment, is very straightforward. It requires about 1150 lines of ADSIM source code. The implementation runs on the AD 100 with a frame time of 444 $\mu$s, allowing the model to run more than four times faster than real time. The same ADSIM model runs on VAXes as well. The frame times on various VAXes allow the model to run between 5 and 20 times slower than real time, depending on the type of VAX. On the AD 100, the entire model requires about 5 percent of the hardware resources.

## Implementation in FORTRAN

To compare the accuracy and performance of the model, ADI also implemented the missile model in FORTRAN. This FORTRAN code runs on the AD 100 as well as on many general-purpose digital computers. The implementation in FORTRAN requires about 3950 lines of source code. On the AD 100, the frame time of the model is about 1100 $\mu$s, allowing the model to run just a bit faster than real time. On a VAX computer, the same FORTRAN code runs from 10 to 50 times slower than real time, depending on the type of VAX.

## Comparing ADSIM and FORTRAN

The implementation in FORTRAN does not support an interactive environment. As a result, the development of the FORTRAN version of the simulation turned out to be orders of magnitude more tedious than the corresponding ADSIM implementation. Such issues as functions, models, sorting of the equations, and optimizing were major hurdles for the FORTRAN implementation, while they were trivial for the ADSIM implementation. The numerical accuracy of the two models is essentially the same. The two implementations provide answers that are similar up to seven or eight decimal places. Both models, when running on the AD 100, can be extended, to hardware-in-the-loop studies.

## Conclusions

The performance numbers of the AD 100 show that it is possible to implement a high-performance missile model in a real-time simulation without the problems associated with an implementation on a general-purpose computer using FORTRAN.

## References


[1] Wright, M. *System 100 Simulation Computer Architecture.* European Simulation Multiconference, June 1989.

[2] *Six-Degree-of-Freedom Missile Simulation.* Applied Dynamics Application Report, March 1989.